

Junos® OS

EVPN User Guide

Published
2025-12-16

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

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

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

Junos® OS EVPN User Guide

Copyright © 2025 Juniper Networks, Inc. All rights reserved.

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

1

About This Guide | xxi

Features Common to EVPN-VXLAN, EVPN-MPLS, and EVPN-VPWS

Configuring Interfaces | 2

VLAN ID Ranges and Lists in an EVPN Environment | 2

Understanding VLAN ID Ranges and Lists in an EVPN Environment | 2

Configuring VLAN ID Lists and Ranges in an EVPN Environment | 5

MAC Address Features with EVPN Networks | 11

Overview of MAC Mobility | 11

Changing Duplicate MAC Address Detection Settings | 14

Configuring Loop Detection for Duplicate MAC Addresses | 16

EVPN Duplicate MAC Detection Exclusion Lists | 20

EVPN MAC Pinning Overview | 23

Configuring EVPN MAC Pinning | 25

Creating an Exclusion List for MAC Pinning | 26

Static Configuration of MAC-IP Bindings | 28

Configuring Routing Instances for EVPN | 31

Configuring EVPN Routing Instances | 31

Configuring EVPN Routing Instances on EX9200 Switches | 36

MAC-VRF Routing Instance Type Overview | 38

EVPN Type 5 Route with VXLAN Encapsulation for EVPN-VXLAN | 47

EVPN Type 5 Route with MPLS encapsulation for EVPN-MPLS | 49

Understanding EVPN Pure Type 5 Routes | 50

Configuring Seamless Type-5 to Type-5 Stitching for EVPN-VXLAN | 55

Seamless VXLAN Stitching with Symmetric EVPN Type 2 Routes using Data Center Interconnect | 58

Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics | 59

- Overview of Symmetric EVPN Routing with Type 2 Routes | 59

- Enable Symmetric IRB with EVPN Type 2 Routes | 63

EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN | 65

Ingress Virtual Machine Traffic Optimization | 75

Tracing EVPN Traffic and Operations | 80

Migrating From BGP VPLS to EVPN Overview | 81

Configuring EVPN over Transport Class Tunnels | 89

Example: Configuring EVPN-VPWS over Transport Class Tunnels | 94

- Overview | 95

- Requirements | 95

- Topology | 95

- Configuration | 96

- Verification | 104

Configuring Route Targets | 108

Auto-derived Route Targets | 108

Example: Configuring VNI Route Targets Automatically | 112

- Requirements | 112

- Overview | 112

- Configuration | 113

Example: Configuring VNI Route Targets Manually | 115

- Requirements | 115

- Overview | 116

- Configuration | 117

Example: Configuring VNI Route Targets Automatically with Manual Override | 120

- Requirements | 121

- Overview | 121

- Configuration | 121

Routing Policies for EVPN | 125

Routing policies for EVPN | 125

Example: Using Policy Filters to Filter EVPN Routes | 132

Requirements | 132

Overview | 133

Base Configuration | 133

Verification | 156

Layer 3 Gateways with Integrated Routing and Bridging for EVPN Overlays | 158

Understanding the MAC Addresses For a Default Virtual Gateway in an EVPN-VXLAN or EVPN-MPLS Overlay Network | 158

EVPN Multihoming | 162

EVPN Multihoming Overview | 162

EVPN Multihoming Designated Forwarder Election | 186

DF Election Overview | 187

DF Election Roles | 188

DF Election Trigger | 189

DF Election Procedure (RFC 7432) | 189

NTP-Based DF Election | 190

Preference-Based DF Election | 192

DF Verification | 198

DF Election for Virtual Switch | 199

Handling Failover | 199

Understanding Automatically Generated ESIs in EVPN Networks | 200

Easy EVPN LAG (EZ-LAG) Configuration | 215

Benefits of Using Easy EVPN LAG Configuration | 216

Easy EVPN LAG Configuration Overview | 217

Simplified CLI Statements and Parameters to Generate the Configuration | 220

Platform-Specific Behavior for EZ-LAG Generated Configurations | 255

Easy EVPN LAG Configuration with Multihomed Servers | 256

Add Configuration for a New Multihomed Server | 268

Add a New VLAN and IRB Interfaces | 270

Add a New Single-homed Server | 273

Use OSPF for the Underlay Configuration | 274

Use IBGP for the Overlay Configuration | 275

Configuring EVPN Active-Standby Multihoming to a Single PE Device | 277

Configuring EVPN-MPLS Active-Standby Multihoming | 280

Example: Configuring Basic EVPN-MPLS Active-Standby Multihoming | 284

- Requirements | 284
- Overview and Topology | 284
- Configuration | 286
- Verification | 297

Example: Configuring EVPN-MPLS Active-Standby Multihoming | 308

- Requirements | 309
- Overview and Topology | 310
- Configuration | 312
- Verification | 330

Example: Configuring EVPN-MPLS Active-Standby Multihoming with ESI | 355

- Example Prerequisites | 356
- Configure ESI per IFD with LACP | 357
- Configure ESI per IFL with CFM | 359
- Verification | 362
- Release Information | 362

Example: Configuring Basic EVPN Active-Active Multihoming | 362

- Requirements | 363
- Overview | 363
- Configuration | 364

Example: Configuring EVPN Active-Active Multihoming | 375

- Requirements | 375
- Overview and Topology | 376
- Configuration | 379
- Verification | 418

Example: Configuring LACP for EVPN Active-Active Multihoming | 442

- Requirements | 442
- Overview | 443
- Configuration | 446
- Verification | 460

Example: Configuring LACP for EVPN VXLAN Active-Active Multihoming | 464

Requirements | 465

Overview | 465

Configuration | 467

Verification | 491

Example: Configuring an ESI on a Logical Interface With EVPN-MPLS Multihoming | 493

Requirements | 494

Overview and Topology | 494

EVPN Multihoming Active-Standby Configuration | 497

Verification | 507

Configuring Dynamic List Next Hop | 510

Link States and Network Isolation Conditions in EVPN Networks | 514

Understanding When to Disable EVPN-VXLAN Core Isolation | 514

Backup Liveness Detection on EVPN Dual Homing | 517

Determine IRB Interface State Changes from Local L2 Interface or Remote Connectivity Status in EVPN Fabrics | 524

Layer 2 Interface Status Tracking and Shutdown Actions for EVPN Core Isolation Conditions | 532

EVPN Maintenance Mode for Multihomed Leaf Isolation | 537

Uplink Protection for Network Isolation | 546

EVPN Proxy ARP and ARP Suppression, and NDP and NDP Suppression | 549

EVPN Proxy ARP and ARP Suppression, and Proxy NDP and NDP Suppression | 549

ARP and NDP Request with a proxy MAC address | 552

Configuring DHCP Relay Agents | 558

DHCP Relay Agent in EVPN-MPLS Network | 558

DHCP Relay Agent over EVPN-VXLAN | 561

High Availability in EVPN | 565

NSR and Unified ISSU Support for EVPN | 565

Preventing Traffic Loss in an EVPN-VXLAN Environment With GRES and NSR | 568

Graceful Restart in EVPN | 569

Monitoring EVPN Networks | 570

Connectivity Fault Management Support for EVPN and Layer 2 VPN Overview | 570

Configure a MEP to Generate and Respond to CFM Protocol Messages | 573

Configure a Maintenance Association End Point (MEP) | 573

Configure a Remote Maintenance Association End Point (MEP) | 575

Layer 2 Control Protocol Transparency | 577

Layer 2 Control Protocol Transparency for EVPN | 577

EVPN-VXLAN

Overview | 588

Understanding EVPN with VXLAN Data Plane Encapsulation | 588

EVPN-over-VXLAN Supported Functionality | 599

Understanding VXLANs | 606

VXLAN Benefits | 607

How Does VXLAN Work? | 607

VXLAN Implementation Methods | 608

Using QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, EX4300-48MP, and EX4600 Switches with VXLANs | 609

Changing the UDP Port on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches | 610

Host Entry Overflow Prevention | 610

Controlling Transit Multicast Traffic on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches | 612

Using an MX Series Router, EX9200 Switch, or QFX10000 Switch as a VTEP | 612

Manual VXLANs Require PIM | 613

Load Balancing VXLAN Traffic | 614

Enabling QFX5120 Switches to Tunnel Traffic on Core-Facing Layer 3 Tagged and IRB Interfaces | 614

Using ping and traceroute with a VXLAN | 615

Supported VXLAN Standards | 615

VXLAN Constraints on EX Series, QFX Series, PTX Series, and ACX Series Devices | 616

EVPN Over VXLAN Encapsulation Configuration Overview for QFX Series and EX4600 Switches | 628

Implementing EVPN-VXLAN for Data Centers | 630

PIM NSR and Unified ISSU Support for VXLAN Overview | 634

Routing IPv6 Data Traffic through an EVPN-VXLAN Network with an IPv4 Underlay | 636

Understanding How to Configure VXLANs and Layer 3 Logical Interfaces to Interoperate | 651

Configuring EVPN-VXLAN Interfaces | 654

Understanding Flexible Ethernet Services Support With EVPN-VXLAN | 654

EVPN-VXLAN Lightweight Leaf to Server Loop Detection | 657

Overlapping VLAN Support Using VLAN Translation in EVPN-VXLAN Networks | 668

Overlapping VLAN Support Using Multiple Forwarding Instances or VLAN Normalization | 674

Layer 2 Protocol Tunneling over VXLAN Tunnels in EVPN-VXLAN Bridged Overlay Networks | 679

L2PT over VXLAN Tunnels | 679

Platform-Specific L2PT over VXLAN Behavior | 684

Configure and Verify L2PT over VXLAN | 685

MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment | 687

GRE over EVPN-VXLAN | 705

DHCP Smart Relay in EVPN-VXLAN | 708

Configuring VLAN-Aware Bundle Services, VLAN-Based Services, and Virtual Switch Support | 710

Understanding VLAN-Aware Bundle and VLAN-Based Service for EVPN | 710

Configuring EVPN with VLAN-Based Service | 711

Configuring EVPN VLAN-Aware Bundle Service | 717

Virtual Switch Support for EVPN Overview | 718

Configuring EVPN with Support for Virtual Switch | 720

Load Balancing with EVPN-VXLAN Multihoming | 723

Dynamic Load Balancing in an EVPN-VXLAN Network | 723

Fast Reroute for Egress Link Protection with EVPN-VXLAN Multihoming | 727

Benefits | 727

How Fast Reroute ELP Works with Multihoming in an Ethernet Segment | 728

Configure Fast Reroute ELP on a PE Device | 732

Verify Fast Reroute ELP Tunnel Creation and Operation | 734

Setting Up a Layer 3 VXLAN Gateway | 741

Using a Default Layer 3 Gateway to Route Traffic in an EVPN-VXLAN Overlay Network | 741

Using a RIOT Loopback Port to Route Traffic in an EVPN-VXLAN Network | 748

Loopback Port Solution for Routing in and out of VXLAN Tunnels (RIOT) for Layer 3 VXLAN Gateway Support | 748

Configure a RIOT Loopback Port on a Layer 3 VXLAN Gateway Leaf Device | 754

Supported Protocols on an IRB Interface in EVPN-VXLAN | 760

VXLAN Layer 3 Gateways Using the Service Provider Style Interface Configuration | 762

Configuring an EVPN-VXLAN Centrally-Routed Bridged Overlay | 771

Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric | 771

Requirements | 772

Overview | 772

Spine 1: Underlay Network Configuration | 776

Spine 1: Overlay Network Configuration | 779

Spine 1: Access Profile Configuration | 781

Spine 2: Full Configuration | 785

Leaf 1: Underlay Network Configuration | 787

Leaf 1: Overlay Network Configuration | 789

Leaf 1: Access Profile Configuration | 791

Leaf 2: Full Configuration | 792

Leaf 3: Full Configuration | 794

Leaf 4: Full Configuration | 795

Verification | 796

Spine 1 and 2: Route Leaking (Optional) | 803

Verification with Route Leaking (Optional) | 805

Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric Using MX Routers as Spines | 808

Requirements | 809

Overview | 810

Topology | 810

Configuration | 811

Verification | 830

Configuring an EVPN-VXLAN Edge-Routed Bridging Overlay | 845

Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway | 845

Requirements | 846

Overview and Topology	847
Configuration For Leaf1	850
Verification	855
Quick Configuration For All Devices	860

Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric With a Virtual Gateway | 866

Requirements	868
Overview and Topology	868
Quick Configuration	871
Underlay Network Configuration	873
EVPN-VXLAN Overlay Network Configuration	875
Customer Profile Configuration	877
Route Leaking Configuration	880
Verification	882
Quick Configuration For All Devices	886

EVPN-VXLAN Pure Type 5 Host-Route Auto-Generated Community | 894

IPv6 Underlay for VXLAN Overlays | 897

EVPN-VXLAN with an IPv6 Underlay | 897

IPv6 Underlay Support in EVPN-VXLAN Fabrics	897
Configure an IPv6 Underlay with EVPN-VXLAN	903

Example: Configure an IPv6 Underlay for Layer 2 VXLAN Gateway Leaf Devices | 905

Overview	906
Requirements	909
Topology	909
Configure Leaf 1	910
Configure Leaf 3	916
Verify the IPv6 Underlay on Leaf 3	920

Multicast Features with EVPN-VXLAN | 925

Multicast Support in EVPN-VXLAN Overlay Networks | 925

Inter-VLAN Multicast Forwarding Modes for EVPN-VXLAN Overlay Networks	925
---	-----

Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment | 934

Example: Preserving Bandwidth with IGMP Snooping in an EVPN-VXLAN Environment | 953

Requirements | **953**

Overview | **954**

Configuration | **955**

Verification | **997**

Overview of Selective Multicast Forwarding | **1005**

Configuring the Number of SMET Nexthops | **1008**

Assisted Replication Multicast Optimization in EVPN Networks | **1010**

Assisted Replication in EVPN Networks | **1010**

How AR Works | **1013**

AR Leaf Device Load Balancing with Multiple Replicators | **1018**

AR Limitations with IGMP Snooping or MLD Snooping | **1028**

AR with Optimized Intersubnet Multicast (OISM) | **1029**

Multicast Forwarding Use Cases in an EVPN Network with AR Devices | **1032**

Configure Assisted Replication | **1038**

Configure IGMP Snooping or MLD Snooping with AR | **1039**

Configure an AR Replicator Device | **1040**

Configure an AR Leaf Device | **1042**

Verify Assisted Replication Setup and Operation | **1043**

Optimized Intersubnet Multicast in EVPN Networks | **1049**

Overview of OISM | **1050**

Overview of Enhanced OISM | **1057**

OISM Components | **1061**

How OISM Works | **1088**

How Enhanced OISM Works | **1112**

Considerations for OISM Configurations | **1117**

Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices | **1158**

Configure Server Leaf Device OISM Elements | **1169**

Configure Border Leaf Device OISM Elements with M-VLAN IRB Method (Symmetric Bridge Domains Model Only) | **1170**

Configure Border Leaf Device OISM Elements with Classic L3 Interface Method | **1175**

Configure Border Leaf Device OISM Elements with Non-EVPN IRB Method | **1177**

CLI Commands to Verify the OISM Configuration | **1180**

EVPN-VXLAN DCI Multicast with Enhanced OISM | 1188

Benefits of DCI Multicast with Enhanced OISM | **1189**

EVPN-VXLAN to EVPN-VXLAN DCI Stitching Overview | **1189**

OISM Overview | **1190**

How Enhanced OISM Works over DCI Stitching | **1192**

More Enhanced OISM over DCI Use Cases | **1196**

Multicast Next Hops Convergence Optimization | **1203**

Configure EVPN-VXLAN DCI with Enhanced OISM | **1204**

Sample Configuration for DCI with Enhanced OISM | **1206**

Show Commands to Verify DCI with Enhanced OISM | **1232**

Microsegmentation Using Group-Based Policies | 1272

Overview | **1272**

Typical Network Use Cases | **1274**

GBP in Large Campus Networks | **1274**

GBP in Branch and Small Campus Networks | **1276**

Supported Platforms | **1278**

GBP Profiles | **1279**

GBP Processing | **1282**

Tag Assignment | **1282**

Assigning Tags Using the CLI | **1284**

Assigning Tags Using RADIUS | **1288**

Tag Assignment Priority | **1292**

Packet Classification | **1294**

Policy Enforcement | **1294**

Enforcement Using GBP Tags | **1295**

Enforcement Using Destination IP Address | **1295**

Enforcement Using L4 Fields | **1296**

Explicit Default Discard | **1297**

Policy Enforcement at the Ingress and Tag Propagation for EVPN-VXLAN | **1298**

Filter-Based Forwarding | **1302**

GBP MAC/IP Inter-tagging | **1307**

Unified Access Policy | **1309**

GBP Messages | **1310**

UAP Access Links | **1311**

UAP Inter-Switch Links | **1312**

- UAP Configuration | 1314
- Using the GBP Pure L2 Profile | 1316

Example: Using GBP to Segment Traffic | 1319

Configuring the Tunneling of Q-in-Q Traffic | 1326

Examples: Tunneling Q-in-Q Traffic in an EVPN-VXLAN Overlay Network | 1326

- Requirements | 1328
- Overview and Topology | 1329
- Configuring Traffic Pattern 1: Popping an S-VLAN Tag | 1333
 - Requirements | 1334
 - Introduction | 1334
 - Ingress VTEP Configuration for Traffic Pattern 1 | 1334
 - Egress VTEP Configuration for Traffic Pattern 1 | 1336

Configuring Traffic Pattern 2: Mapping a Range of C-VLANs to an S-VLAN, and Pushing an S-VLAN Tag | 1338

- Requirements | 1339
- Introduction | 1339
- Ingress VTEP Configuration for Traffic Pattern 2 | 1339
- Egress VTEP Configuration for Traffic Pattern 2 | 1341

Configuring Traffic Pattern 3: Retaining S-VLAN and C-VLAN Tags | 1344

- Requirements | 1344
- Introduction | 1344
- Ingress and Egress VTEP Configuration for Traffic Pattern 3 | 1344

Configuring Traffic Pattern 4: Popping and Later Pushing an S-VLAN Tag | 1347

- Requirements | 1347
- Introduction | 1347
- Configuration for Ingress VTEP for Traffic Pattern 4 | 1347
- Configuration for Egress VTEP for Traffic Pattern 4 | 1350

Tunnel Traffic Inspection on SRX Series Devices | 1354

Tunnel Inspection for EVPN-VXLAN by SRX Series Devices | 1354

- Overview | 1354
- Example - Configure Security Policies for EVPN-VXLAN Tunnel Inspection | 1358
- Configuration for Zone-Level Inspection, IDP, Content Security and Advanced Anti-Malware for Tunnel Inspection | 1371
- Complete Device Configurations | 1381

Fault Detection and Isolation in EVPN-VXLAN Fabrics | 1394

Understanding Overlay ping and traceroute Packet Support | **1394**

Example: Troubleshoot a VXLAN Overlay Network with Overlay Ping and Overlay Traceroute on QFX Series Switches | **1398**

Requirements | **1399**

Overview and Topology | **1400**

Verification | **1403**

Example: Troubleshoot a VXLAN Overlay Network with Overlay Ping and Overlay Traceroute for MX Series Routers | **1413**

Requirements | **1414**

Overview and Topology | **1415**

Configuration | **1418**

Verification | **1418**

EVPN-MPLS

Overview | 1429

EVPN Overview | **1429**

EVPN-MPLS Caveats | **1431**

EVPN Overview for Switches | **1432**

Migrating from FEC128 LDP-VPLS to EVPN Overview | **1434**

Convergence in an EVPN MPLS Network | 1444

Convergence in a Multihomed EVPN-MPLS Network | **1444**

Pseudowire Termination at an EVPN | 1446

Overview of Pseudowire Termination at an EVPN | **1446**

Configuring Pseudowire Termination | **1447**

Support for Redundant Logical Tunnel | **1451**

Configuring the Distribution of Routes | 1453

Configuring an IGP on the PE and P Routers on EX9200 Switches | **1453**

Configuring IBGP Sessions Between PE Routers in VPNs on EX9200 Switches | **1454**

Configuring a Signaling Protocol and LSPs for VPNs on EX9200 Switches | **1455**

Configuring Entropy Labels | 1458

Configuring Control Word for EVPN-MPLS | 1459

Understanding P2MPs LSP for the EVPN Inclusive Provider Tunnel | 1461

Configuring Bud Node Support | 1463

Configuring VLAN Services and Virtual Switch Support | 1465

Overview of VLAN Services for EVPN | 1465

VLAN-Based Service for EVPN | 1467

VLAN Bundle Service for EVPN | 1471

Configuring EVPN VLAN Bundle Services | 1473

Virtual Switch Support for EVPN Overview | 1476

Configuring EVPN with Support for Virtual Switch | 1478

Example: Configuring EVPN with Support for Virtual Switch | 1482

Example: Configuring EVPN with Support for Virtual Switch | 1482

Requirements | 1483

Overview | 1483

Configuration | 1484

Verification | 1495

Configuring Integrated Bridging and Routing | 1499

EVPN with IRB Solution Overview | 1499

An EVPN with IRB Solution on EX9200 Switches Overview | 1505

Anycast Gateways | 1511

Configuring EVPN with IRB Solution | 1515

Configuring an EVPN with IRB Solution on EX9200 Switches | 1519

Example: Configuring EVPN with IRB Solution | 1522

EVPN with IRB Solution Overview | 1522

Example: Configuring EVPN-MPLS with IRB Solution | 1528

Requirements | 1529

Overview | 1529

Configuration | 1530

Verification | 1538

Example: Configuring an EVPN with IRB Solution on EX9200 Switches | 1547

Requirements | 1547

Overview | 1547

Configuration | 1548

Verification | 1557

Configuring IGMP or MLD Snooping with EVPN-MPLS | 1565

Overview of Multicast Forwarding with IGMP or MLD Snooping in an EVPN-MPLS Environment | 1565

Configure Multicast Forwarding with IGMP Snooping in an EVPN-MPLS Environment | 1580

Configure IGMP Snooping for an EVPN or Virtual Switch Routing Instance | 1581

Configure IGMP Snooping with IGMPv3 on ACX Series Routers to Process Source-Specific Multicast Group Membership Reports Only | 1583

Configure Multicast Routing Across Bridge Domains or VLANs with PIM in EVPN-MPLS | 1584

Viewing IGMP Snooping Multicast Information for EVPN-MPLS in the CLI | 1587

Configure Multicast Forwarding with MLD Snooping in an EVPN-MPLS Environment | 1590

Configure MLD Snooping for Default Any-Source Multicast (ASM) Group Membership Processing with MLDv1 or MLDv2 | 1591

Configure MLD Snooping with MLDv2 to Process Source-Specific Multicast Group Membership Reports Only | 1594

Viewing MLD Snooping Multicast Information for EVPN-MPLS in the CLI | 1595

4

EVPN E-LAN

EVPN E-LAN Services | 1598

EVPN E-LAN Overview | 1598

Supported Routing Instance Types and Services for EVPN-ELAN | 1599

EVPN E-LAN over SRv6 | 1602

5

EVPN-VPWS

Configuring VPWS Service with EVPN Mechanisms | 1606

Overview of VPWS with EVPN Signaling Mechanisms | 1606

Control word for EVPN-VPWS | 1610

Overview of Flexible Cross-Connect Support on VPWS with EVPN | 1614

Overview of Headend Termination for EVPN VPWS for Business Services | 1621

Configuring VPWS with EVPN Signaling Mechanisms | 1629

Example: Configuring VPWS with EVPN Signaling Mechanisms | 1632

Requirements | 1632

Overview and Topology | 1632

Configuration | 1634

Verification | 1646

FAT Flow Labels in EVPN-VPWS Routing Instances | 1657

Configuring EVPN-VPWS over SRv6 | 1661

Configuring Micro-SIDs in EVPN-VPWS | 1667

Configuring EVPN-VPWS over SRv6 with Traffic Engineering | 1674

6

EVPN E-Tree

Overview | 1697

EVPN E-Tree Overview | 1697

Configuring EVPN E-Tree | 1701

Example: Configuring EVPN E-Tree Service | 1701

Requirements | 1701

Overview | 1702

Configuration | 1703

Verification | 1712

7

Using EVPN for Interconnection

Interconnecting VXLAN Data Centers With EVPN | 1719

VXLAN Data Center Interconnect Using EVPN Overview | 1719

Example: Configuring VXLAN Data Center Interconnect Using EVPN | 1739

Requirements | 1739

Overview | 1740

Configuration | 1741

Verification | 1753

Interconnecting EVPN-VXLAN Data Centers Through an EVPN-MPLS WAN | 1760

EVPN-VXLAN Data Center Interconnect Through EVPN-MPLS WAN Overview | 1760

Example: Interconnecting EVPN-VXLAN Data Center Networks Through a WAN Running EVPN-based MPLS | **1771**

Requirements | **1771**

Overview | **1772**

Configuration | **1775**

Verification | **1790**

Appendix 1: Set Commands on All Devices | **1796**

Appendix 2: Show Configuration Output on DUT | **1826**

Overview of EVPN-VXLAN Interconnect through EVPN MPLS WAN Using Gateways | **1842**

Extending a Junos Fusion Enterprise Using EVPN-MPLS | 1845

Understanding EVPN-MPLS Interworking with Junos Fusion Enterprise and MC-LAG | **1845**

Example: EVPN-MPLS Interworking With Junos Fusion Enterprise | **1851**

Requirements | **1852**

Overview and Topology | **1852**

Aggregation Device (PE1 and PE2) Configuration | **1855**

PE3 Configuration | **1867**

Example: EVPN-MPLS Interworking With an MC-LAG Topology | **1871**

Requirements | **1872**

Overview and Topology | **1872**

PE1 and PE2 Configuration | **1875**

PE3 Configuration | **1891**

8

PBB-EVPN

Configuring PBB-EVPN Integration | 1897

Provider Backbone Bridging (PBB) and EVPN Integration Overview | **1897**

Example: Configuring PBB with Single-Homed EVPN | **1931**

Requirements | **1932**

Overview and Topology | **1932**

Configuration | **1933**

Verification | **1960**

Example: Configuring PBB with Multihomed EVPN | **1968**

Requirements | **1968**

Overview and Topology | **1969**

Configuration | 1970

Verification | 2002

Configuring MAC Pinning for PBB-EVPNs | 2010

PBB-EVPN MAC Pinning Overview | 2010

Configuring PBB-EVPN MAC Pinning | 2011

9

EVPN Standards

Supported EVPN Standards | 2016

Supported EVPN Standards | 2016

10

VXLAN-Only Features

Flexible VXLAN Tunnels | 2019

Understanding Programmable Flexible VXLAN Tunnels | 2019

Static VXLAN | 2026

Static VXLAN | 2026

Understanding Static VXLAN | 2026

Q-in-Q VLAN Tunnels in a Spine-and-Leaf Network with Static VXLAN | 2031

Configure Static VXLAN at the Global Level | 2033

Configure Static VXLAN at the VLAN or Bridge Domain Level on L2 VXLAN Gateway Devices | 2036

11

Configuration Statements and Operational Commands

Junos CLI Reference Overview | 2043

About This Guide

Use this guide to learn more about, configure, and monitor EVPN-VXLAN, EVPN-MPLS, EVPN-VPWS, EVPN-ETREE, PBB-EVPN, and Static VXLAN on Juniper Network devices.

1

PART

Features Common to EVPN-VXLAN, EVPN-MPLS, and EVPN-VPWS

- [Configuring Interfaces | 2](#)
 - [MAC Address Features with EVPN Networks | 11](#)
 - [Configuring Routing Instances for EVPN | 31](#)
 - [Configuring Route Targets | 108](#)
 - [Routing Policies for EVPN | 125](#)
 - [Layer 3 Gateways with Integrated Routing and Bridging for EVPN Overlays | 158](#)
 - [EVPN Multihoming | 162](#)
 - [Link States and Network Isolation Conditions in EVPN Networks | 514](#)
 - [EVPN Proxy ARP and ARP Suppression, and NDP and NDP Suppression | 549](#)
 - [Configuring DHCP Relay Agents | 558](#)
 - [High Availability in EVPN | 565](#)
 - [Monitoring EVPN Networks | 570](#)
 - [Layer 2 Control Protocol Transparency | 577](#)
-

CHAPTER 1

Configuring Interfaces

IN THIS CHAPTER

- [VLAN ID Ranges and Lists in an EVPN Environment | 2](#)

VLAN ID Ranges and Lists in an EVPN Environment

IN THIS SECTION

- [Understanding VLAN ID Ranges and Lists in an EVPN Environment | 2](#)
- [Configuring VLAN ID Lists and Ranges in an EVPN Environment | 5](#)

You can specify VLAN ID lists and ranges in a service provider style of interface configuration that is referenced in an Ethernet VPN (EVPN) routing instance (a routing instance of type evpn). For more information, see the following topics:

Understanding VLAN ID Ranges and Lists in an EVPN Environment

IN THIS SECTION

- [Benefits of VLAN ID Range and List Support | 3](#)
- [VLAN Bundle Service | 4](#)
- [Sample VLAN ID Range and List Configuration | 4](#)
- [Caveats and Limitations | 5](#)

The service provider style of interface configuration enables you to customize Ethernet-based services at the logical interface level. Service providers typically have multiple customers connected to the same physical interface or aggregated Ethernet interface. Using the service provider style, you can configure multiple logical interfaces on the physical interface or aggregated Ethernet interface and associate each unit with a different VLAN.

Starting in Junos OS Release 19.2R1, you can specify VLAN ID lists and ranges in a service provider style interface configuration that is referenced in an Ethernet VPN (EVPN) routing instance (a routing instance of type evpn). This configuration is supported with the following EVPN environments, services, and features:

- Environments:
 - EVPN with Virtual Extensible LAN (VXLAN) encapsulation
 - EVPN with MPLS encapsulation
- VLAN bundle service:
 - E-LAN
 - E-Tree
 - E-Line
- Features:
 - EVPN multihoming
 - All-active
 - Single-active
 - Single homing

Benefits of VLAN ID Range and List Support

Without the support of VLAN ID ranges and lists, you must configure a dedicated logical interface for each VLAN. VLAN ID range and list support enables you to associate multiple VLANs with a single logical interface, which reduces the overall number of logical interfaces needed. Using fewer logical interfaces provides these benefits:

- Reduces the amount of configuration time
- Reduces the amount of memory consumed
- Reduces the impact to system performance

VLAN Bundle Service

The VLAN bundle service supports the mapping of multiple broadcast domains (VLANs) to a single bridge domain (MAC learning domain). You can associate multiple VLANs with a single EVPN routing instance. As a result, these broadcast domains (VLANs) share the same MAC table in the EVPN routing instance, thereby reducing the utilization of resources—for example, the number of MAC tables, MAC routes, and labels.

Sample VLAN ID Range and List Configuration

The following sample configuration shows a service provider style interface (interface xe-1/0/0 and logical interfaces xe-1/0/0.0 and xe-1/0/0.1) that is configured on a Juniper Networks device in an EVPN-VXLAN topology. The sample configuration also shows the EVPN routing instance (EVPN-VXLAN-3) in which logical interfaces xe-1/0/0.0 and xe-1/0/0.1 are referenced.

```

interfaces {
  xe-1/0/0 {
    unit 0 {
      encapsulation vlan-bridge;
      vlan-id-range 100-102;
      family bridge;
    }
    unit 1 {
      encapsulation vlan-bridge;
      vlan-id-list [ 200-203 213 248 ];
      family bridge;
    }
  }
}

routing-instances {
  EVPN-VXLAN-3 {
    description "EVPN-VXLAN Vlan Bundle service";
    instance-type evpn;
    vtep-source-interface lo0.0;
    interface xe-1/0/0.0;
    interface xe-1/0/0.1;
    route-distinguisher 10.255.235.35:200;
    vrf-target target:123:123;
    protocols {
      evpn {
        encapsulation vxlan;
      }
    }
  }
}

```

```

    }
  }
  vxlan {
    vni 551;
    encapsulate-inner-vlan;
    decapsulate-accept-inner-vlan;
  }
}
}

```

In this configuration, logical interface xe-1/0/0.0 includes a VLAN ID range and logical interface xe-1/0/0.1 includes a VLAN ID list, which is composed of a VLAN ID range and individual VLAN IDs. EVPN routing instance EVPN-VXLAN-3 references both logical interfaces.

Caveats and Limitations

When specifying VLAN ID ranges and lists in a service provider style interface configuration in an EVPN environment, keep these caveats and limitations in mind:

- When specifying a range in either a VLAN ID range or list, you must use an ascending range—for example, 100-102. If you specify a descending range—for example, 102-100—the system considers the range to be invalid, and a commit error occurs.

Configuring VLAN ID Lists and Ranges in an EVPN Environment

Starting in Junos OS Release 19.2R1, you can specify VLAN ID lists and ranges in a service provider style of interface configuration that is referenced in an Ethernet VPN (EVPN) routing instance (a routing instance of type evpn).

This feature enables you to associate multiple VLANs with a single logical interface, thereby freeing you from having to configure a dedicated logical interface for each VLAN.

This feature works with the VLAN bundle service.

This procedure shows you how to specify multiple VLANs using VLAN ID ranges and lists in a service provider style interface configuration and to associate the interface with an EVPN routing instance.

The sample configurations that follow the procedure provide more comprehensive configurations of service provider style interfaces in an EVPN environment.

1. Configure the physical interface or aggregated Ethernet interface with the encapsulation type of flexible Ethernet services, which enables you to specify Ethernet encapsulations at the logical interface level.

```
[edit]
user@switch# set interfaces interface-name encapsulation flexible-ethernet-services
```

2. Configure a service provider style logical interface.
 - a. Specify the encapsulation type of `vlan-bridge` to enable bridging on the logical interface:

```
[edit]
user@switch# set interfaces interface-name unit logical-unit-number encapsulation vlan-bridge
```

- b. Associate the logical interface with multiple VLANs using either a VLAN ID range or list:

```
[edit]
user@switch# set interfaces interface-name unit logical-unit-number vlan-id-range vlan-idA-vlan-idB
```

OR

```
[edit]
user@switch# set interfaces interface-name unit logical-unit-number vlan-id-list [ vlan-id1 vlan-id2 vlan-idA-vlan-idB ...]
```

3. Repeat Step 2 for each additional service provider style logical interface that you need to configure.
4. Create an EVPN routing instance.
 - a. Specify that the routing instance is of type `evpn` or other supported instance type for EVPN instances. For example:

```
[edit]
user@switch# set routing-instances routing-instance-name instance-type evpn
```

- b. Associate the logical interfaces that you configured earlier with the EVPN routing instance.

```
[edit]
user@switch# set routing-instances routing-instance-name interface interface-name.logical-unit-number
```

Sample Configuration: Multiple Logical Interfaces

This sample configuration shows aggregated Ethernet interface ae0, which is divided into logical interfaces ae0.100 and ae0.150. Logical interface ae0.100 is associated with VLANs ranging from 100 through 102. Logical interface ae0.150 is associated with a list of VLANs, which includes 150 through 152, 200, 213, and 248. EVPN routing instance EVPN-1 references both logical interfaces.

```
interfaces {
  ae0 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 100 {
      encapsulation vlan-bridge;
      vlan-id-range 100-102;
      family bridge;
    }
    unit 150 {
      encapsulation vlan-bridge;
      vlan-id-list [ 150-152 200 213 248 ];
      family bridge;
    }
  }
}

routing-instances {
  EVPN-1 {
    instance-type evpn;
    interface ae0.100;
    interface ae0.150;
    route-distinguisher 192.160.0.1:111;
    vrf-target target:65000:111;
    protocols {
      evpn;
    }
  }
}
```



```
    }
}
```

Sample Configuration: Single Logical Interface

This sample configuration is similar to the multiple logical interface sample configuration except that aggregated Ethernet interface ae0 includes only one logical interface (ae0.150) with which all VLANs (100 through 102, 150 through 152, 200, 213, and 248) are associated. EVPN routing instance EVPN-1 references logical interface ae0.150.

```
interfaces {
  ae0 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 150 {
      encapsulation vlan-bridge;
      vlan-id-list [ 100-102 150-152 200 213 248 ];
      family bridge;
    }
  }
}

routing-instances {
  EVPN-1 {
    instance-type evpn;
    interface ae0.150;
    route-distinguisher 192.160.0.1:111;
    vrf-target target:65000:111;
    protocols {
      evpn;
    }
  }
}
```

Sample Configuration: E-Tree

This sample E-Tree configuration is similar to the other sample configurations except for some information specific to E-Tree use (for example, specifying each logical interface as either root or leaf, and enabling the EVPN-ETREE service).

```
interfaces {
  ae0 {
```

```

flexible-vlan-tagging;
encapsulation flexible-ethernet-services;
unit 100 {
    encapsulation vlan-bridge;
    vlan-id-range 100-102;
    family bridge;
    etree-ac-role leaf;
}
unit 200 {
    encapsulation vlan-bridge;
    vlan-id-list [ 200 213 248 ];
    family bridge;
    etree-ac-role root;
}
}

routing-instances {
    ETREE-1 {
        instance-type evpn;
        interface ae0.100;
        interface ae0.200;
        route-distinguisher 192.160.0.1:111;
        vrf-target target:65000:111;
        protocols {
            evpn {
                evpn-etree;
            }
        }
    }
}

```

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 in Junos OS Release 19.2R1, you can specify VLAN ID lists and ranges in a service provider style interface configuration that is referenced in an Ethernet VPN (EVPN) routing instance (a routing instance of type evpn).

RELATED DOCUMENTATION

| *Flexible Ethernet Services Encapsulation*

MAC Address Features with EVPN Networks

IN THIS CHAPTER

- Overview of MAC Mobility | 11
- Changing Duplicate MAC Address Detection Settings | 14
- Configuring Loop Detection for Duplicate MAC Addresses | 16
- EVPN Duplicate MAC Detection Exclusion Lists | 20
- EVPN MAC Pinning Overview | 23
- Configuring EVPN MAC Pinning | 25
- Creating an Exclusion List for MAC Pinning | 26
- Static Configuration of MAC-IP Bindings | 28

Overview of MAC Mobility

MAC mobility describes the scenario where a host moves from one Ethernet segment to another segment in the EVPN network. Provider Edge (PE) devices discover the host MAC address from its local interfaces or from remote PE devices. When a PE device learns of a new local MAC address, it sends a MAC advertisement route message to other devices in the network. During this time, there are two advertised routes and the PE devices in the EVPN network must decide which of the MAC advertisement messages to use.

To determine the correct MAC address location, PE devices use the MAC mobility extended community field, as defined in RFC 7432, in the MAC advertisement route message. The MAC mobility extended community includes a static flag and a sequence number. The static flag identifies pinned MAC addresses that should not be relocated. The sequence number identifies newer MAC advertisement messages. Starting at 0, the sequence number is incremented for every MAC address mobility event. PE devices running Junos OS apply the following precedence order in determining the MAC advertisement route to use:

1. Advertisement routes with a local pinned MAC address (static MAC address).
2. Advertisement routes with a remote pinned MAC address (static MAC address).

3. Advertisement routes with a higher sequence number.



NOTE: When there are two advertisement route messages for pinned MAC addresses with different routes or two advertisement route messages with the same sequence number, the local device chooses the advertisement route message from the PE device with the lower IP address.

Figure 1 on page 12 illustrates a network where a MAC address is relocated from PE1 to PE2. Before the move, a MAC advertisement route message sent by PE1 has the active route for all PE devices in the network. After the relocation, PE2 learns of the new local MAC address and sends an updated MAC advertisement route message. Table 1 on page 12 lists the action taken by each PE device based on the two MAC advertisements. The PE device generates a syslog message when it encounters conflicts with a pinned MAC address.



NOTE: Table 1 on page 12 includes use cases with pinned MAC addresses. These use cases do not apply to PE devices that do not support MAC pinning. To determine whether or not MAC pinning is supported by a particular Juniper Networks device or Junos OS release, see [Feature Explorer](#).

Figure 1: MAC Mobility in an EVPN Network

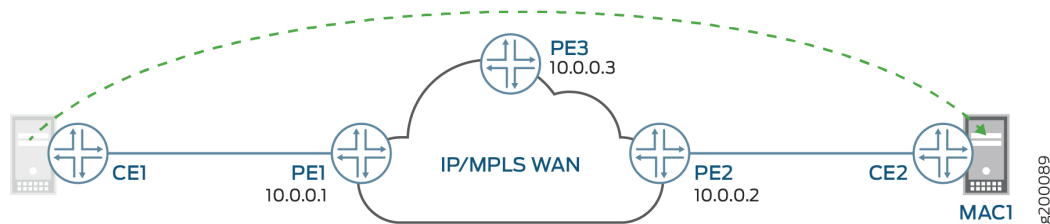


Table 1: MAC Advertisement Routes on PE Devices

MAC Advertisement	PE1	PE2	PE3
PE1: MAC address with a sequence number (n).	Install the remote MAC advertisement route from PE2 because it has a higher sequence number (n+1).	Advertise the local MAC route because it has a higher sequence number (n+1).	Install the remote MAC advertisement route from PE2 because it has a higher sequence number (n+1).
PE2: MAC address with the sequence number incremented by one (n+1).			

Table 1: MAC Advertisement Routes on PE Devices (Continued)

MAC Advertisement	PE1	PE2	PE3
PE1: MAC address with a sequence number (n). PE2: MAC address with the same sequence number (n).	Advertise the local MAC route because PE1 has the lower IP address (10.0.0.1).	Install the remote MAC advertisement route from PE1 because PE1 has the lower IP address (10.0.0.1).	Use the MAC advertisement route from PE1 because PE1 has the lower IP address (10.0.0.1).
PE1: Pinned MAC address with the static bit set. PE2: MAC address and a sequence number (n).	Advertise the local MAC route because it is a pinned MAC address. Generate a syslog message.	Install the remote MAC advertisement route from PE1 because it is a pinned MAC address.	Use the MAC advertisement route from PE1 because it is a pinned MAC address. Generate a syslog message.
PE1: MAC address with a sequence number (n). PE2: Pinned MAC address with the static bit set.	Install the remote the MAC advertisement route from PE2 because it is a pinned MAC address.	Advertise local MAC route because it is a pinned MAC address. Generate a syslog message.	Install the remote MAC advertisement route from PE2 because it is a pinned MAC address.
PE1: Pinned MAC address with static bit set. PE2: Pinned MAC address with static bit set.	Advertise the local MAC route because it is a local pinned MAC address. Generate a syslog message.	Advertise the local MAC route because it is a local pinned MAC address. Generate a syslog message.	Use the MAC advertisement route from PE1 because PE1 has the lower IP address (10.0.0.1). Generate a syslog message.

Junos supports MAC mobility automatically by default. To disable MAC mobility, use the set protocols evpn mac-mobility no-sequence-numbers statement.

RELATED DOCUMENTATION

[EVPN MAC Pinning Overview](#) | 23

clear evpn duplicate-mac-suppression

Changing Duplicate MAC Address Detection Settings

When a host is physically moved or when a host is reconfigured on a different Ethernet segment, the PE device sends an updated MAC advertisement route to other PE devices to update their route table. If there is a misconfiguration in the network, MAC advertisement messages oscillate between the different routes causing MAC address flapping. This makes the network more vulnerable and wastes network resources. Junos supports MAC mobility automatically by default. To disable MAC mobility, use the `set protocols evpn mac-mobility no-sequence-numbers` statement.

Junos OS also automatically detects and suppresses duplicate MAC addresses. Optionally, you can also configure the length of time that the duplicate MAC address is suppressed. When the PE device encounters duplicate MAC addresses, Junos OS generates a syslog message.

To change the duplicate MAC address detection settings, include the `duplicate-mac-detection` statement at either the `[edit routing-instances routing-instance-name protocols]` hierarchy level or the `[edit logical-systems logical-system-name routing-instances routing-instance-name protocols]` hierarchy level:

```
evpn
  duplicate-mac-detection {
    detection-threshold detection-threshold;
    detection-window seconds;
    auto-recovery-time minutes;
  }
```

You can modify the following options under the `duplicate-mac-detection` statement:

- `detection-window`—The time interval used in detecting a duplicate MAC address. The value can be from 5 through 600 seconds. The default is 180 seconds
- `detection-threshold`—The number of MAC mobility events that are detected for a given MAC address within the `detection-window` before it is identified as a duplicate MAC address. Once the detection threshold is reached, updates for the MAC address are suppressed. The value can be from 2 through 20. The default is 5.
- `auto-recovery-time`—(Optional) The length of time a device suppresses a duplicate MAC address. At the end of this duration, MAC address updates will resume. The value can be from 1 through 360 minutes. If a value is not specified, then the MAC address continues to be suppressed.



NOTE: To ensure that the mobility advertisements have sufficient time to age out, set an auto-recovery-time greater than the detection-window.

To manually clear the suppression of duplicate MAC addresses, use the `clear evpn duplicate-mac-suppression` command.

To view MAC duplicate addresses in the EVPN MAC database, use the `show evpn database` command. The following example displays a sample output. The output fields related to duplicate MAC detections are State, Mobility history, and MAC advertisement route status:

```
user@PE1> show evpn database mac-address 00:00:00:00:00:02
extensive

Instance: ALPHA

VLAN ID: 100, MAC address: 00:00:00:00:00:02
State: 0x1 <Duplicate-Detected>
Mobility history
  Mobility event time      Type      Source                      Seq num
  Aug 03 17:22:28.585619  Local     ge-0/0/2.0                  31
  Aug 03 17:22:30.307198  Remote    10.255.0.3                   32
  Aug 03 17:22:37.611786  Local     ge-0/0/2.0                  33
  Aug 03 17:22:39.289357  Remote    10.255.0.3                   34
  Aug 03 17:22:45.609449  Local     ge-0/0/2.0                  35
Source: ge-0/0/2.0, Rank: 1, Status: Active
  Mobility sequence number: 35 (minimum origin address 10.255.0.2)
  Timestamp: Aug 03 17:22:44 (0x5983be54)
  State: <Local-MAC-Only Local-To-Remote-Adv-Allowed>
  MAC advertisement route status: Not created (duplicate MAC suppression)
  IP address: 10.0.0.2
Source: 10.255.0.3, Rank: 2, Status: Inactive
  MAC label: 300176
  Mobility sequence number: 34 (minimum origin address 10.255.0.3)
  Timestamp: Aug 03 17:22:39 (0x5983be4f)
  State: <>
  MAC advertisement route status: Not created (inactive source)
  IP address: 10.0.0.3
```


RELATED DOCUMENTATION

clear evpn duplicate-mac-suppression

duplicate-mac-detection

Configuring Loop Detection for Duplicate MAC Addresses

IN THIS SECTION

- [Understanding Duplicate MAC Address Loop Detection | 16](#)
- [Sample Configurations | 19](#)

Understanding Duplicate MAC Address Loop Detection

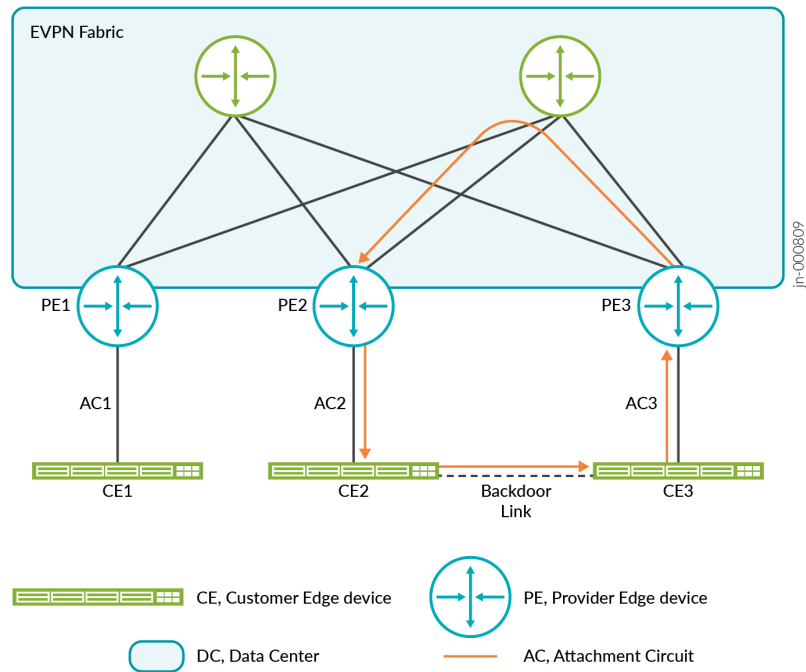


NOTE: For information on how to change settings for duplicate MAC address detection, see *duplicate-mac-detection*.

You can use duplicate MAC address loop detection to detect and resolve loops within the same broadcast domain in an EVPN fabric or between EVPN fabrics. A loop can occur when there is a backdoor path between two provider edge (PE) devices. Because of the backdoor path, PEs could forward a frame back and forth continuously.

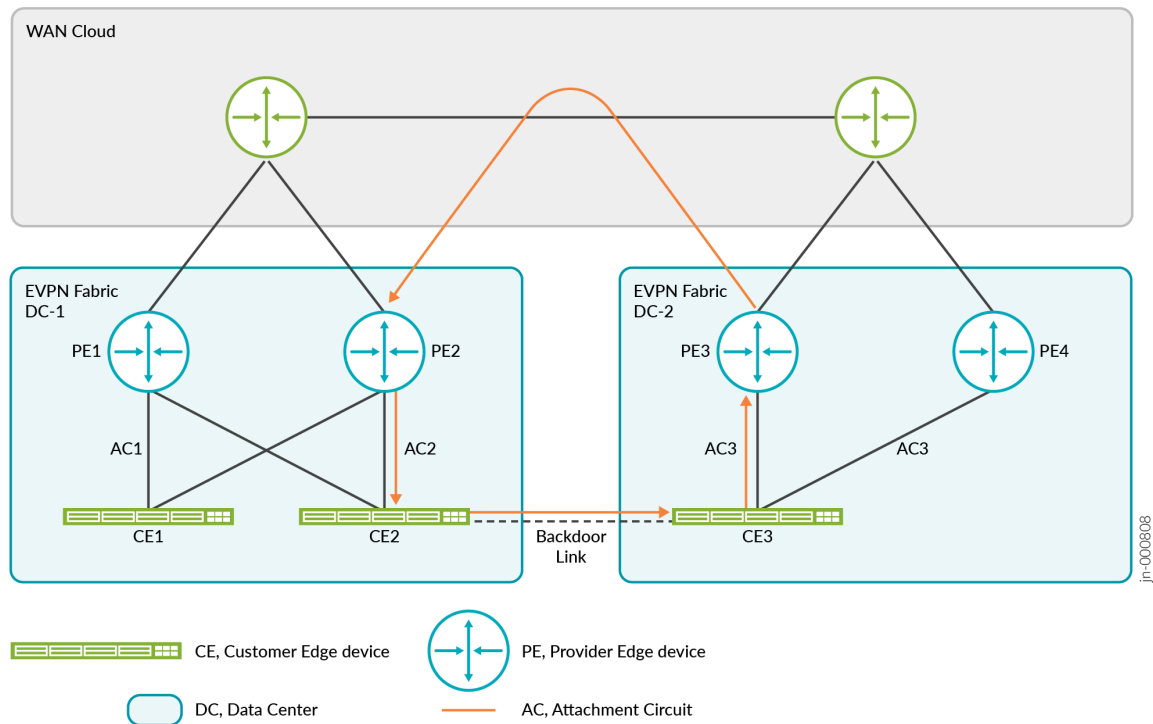
There are two types of loops: local and global. A local loop occurs when there is a backdoor path within the same physical interface or between two attachment circuits (ACs) in the same network virtual interface (NVE). A backdoor path can occur when there is a Layer 2 connection between NVEs within an EVPN instance (EVI).

Figure 2: Backdoor Link in an EVPN Fabric



A global loop occurs when there is a backdoor link between two ACs in the same EVI, but the EVIs are located in different NVEs.

Figure 3: Backdoor Link Between Two EVPN Fabrics



We have enhanced duplicate MAC detection to detect and resolve loops. You can resolve the loops by either blocking duplicate MAC addresses or shutting down the local interfaces associated with the duplicate MAC addresses. For duplicate MAC resolution to work, you also need to configure duplicate MAC address detection.

When a MAC address is marked as a duplicate MAC address, a PE device drops any packet that has a source address or destination address of the duplicate MAC address. Optionally, instead of dropping packets, you could configure a PE device to bring down the attachment circuit on which the frame was last seen.

To block duplicate MAC addresses and shut down their associated local interfaces, enable the action `<block | shutdown>` statement at the `[edit routing-instances name protocols evpn duplicate-mac-detection]` hierarchy. To track local MAC address mobility movements, enable the `include-local-moves` statement at the `[edit routing-instances name protocols evpn duplicate-mac-detection]` hierarchy.

Sample Configurations

Blocking Duplicate MAC Addresses

Here is a sample configuration that shows you how to block duplicate MAC addresses.

```
set routing-instances rtt1 protocols evpn duplicate-mac-detection detection-threshold 3
set routing-instances rtt1 protocols evpn duplicate-mac-detection detection-window 5
set routing-instances rtt1 protocols evpn duplicate-mac-detection auto-recovery-time 10
set routing-instances rtt1 protocols evpn duplicate-mac-detection action block
set routing-instances rtt1 protocols evpn duplicate-mac-detection include-local-moves
```

Shutting Down Local Interfaces

Here is a sample configuration that shows you how to shut down the local interfaces that are associated with the duplicate MAC addresses.

```
set routing-instances rtt1 protocols evpn duplicate-mac-detection detection-threshold 3
set routing-instances rtt1 protocols evpn duplicate-mac-detection detection-window 5
set routing-instances rtt1 protocols evpn duplicate-mac-detection action shutdown
set interfaces et-0/0/0 unit 0 family ethernet-switching recovery-timeout 10
```

Manually Clearing Duplicate MAC Addresses

To manually clear the duplicate MAC addresses, issue the `clear evpn duplicate-mac-suppression` command.

You can also clear duplicate MAC addresses individually or per Layer 2 domain by issuing the `clear evpn duplicate-mac-suppression l2-domain-id` or `clear evpn duplicate-mac-suppression mac-address` commands.

Manually Recovering Interfaces that were Shut Down

To manually recover the interface that was shut down, issue the `clear ethernet-switching recovery-timeout` command.

RELATED DOCUMENTATION

| *duplicate-mac-detection*

EVPN Duplicate MAC Detection Exclusion Lists

SUMMARY

Use the EVPN duplicate detection MAC exclusion lists to bypass the duplicate detection process in scenarios where legitimate MAC movements are frequent.

IN THIS SECTION

- [Benefits of MAC Exclusion List for Duplicate Detection | 20](#)
- [Overview | 21](#)
- [Configuration and Management | 21](#)
- [Usage Scenarios | 22](#)

The MAC Exclusion List for Duplicate Detection in EVPN environments enables you to configure specific MAC addresses that should be excluded from duplicate MAC detection. This feature is particularly beneficial for scenarios involving legitimate MAC address movements, such as VRRP virtual MAC configurations in redundant setups. You can bind exclusion lists to specific EVPN instances, providing granular control over MAC address monitoring. The feature also supports prefix length for MAC addresses, facilitating the management of larger address sets with fewer commands. Comprehensive CLI commands are available for configuring and displaying exclusion lists, ensuring clear visibility and management of network configurations.



NOTE: Support for MAC address with prefix length was added in Junos 25.2R1. This changed the MAC address CLI format for the EVPN *mac-list* and will cause a validation error when upgrading from an image that doesn't support the MAC address with prefix length format, if the previous image has EVPN *mac-list* configured.

Use the following steps to prevent a validation error during the upgrade.

- Before upgrading, deactivate the *mac-list* configuration to prevent validation errors.
- After the upgrade, reconfigure the *mac-list* using the new format.

Benefits of MAC Exclusion List for Duplicate Detection

- Prevents legitimate MAC address movements from being flagged as duplicates, ensuring smooth network operations without unnecessary traffic disruptions.
- Enhances network reliability and robustness by accurately distinguishing between genuine duplicate MAC situations and legitimate MAC mobility scenarios.

- Enables precise control over MAC address exclusion by supporting configuration with prefix lengths, facilitating the management of MAC address ranges rather than individual addresses.
- Provides granular control over MAC address monitoring by enabling the binding of exclusion lists to specific EVPN instances, facilitating precise network management.

Overview

The MAC Exclusion List for Duplicate Detection feature in EVPN environments gives you the ability to exclude specific MAC addresses from being flagged as duplicates. This is particularly useful in scenarios where frequent and legitimate MAC address movements occur, such as in redundant configurations using VRRP virtual MACs. By configuring an exclusion list, you ensure that necessary traffic movements are not mistakenly identified as duplicate MAC addresses, thus preventing unnecessary network disruptions and maintaining smooth operations.

To configure the MAC Exclusion List, use the CLI command `set protocols evpn mac-list list_name mac-address mac_address_with_prefix_len`. With this command you define a list of MAC addresses, with prefix lengths for more granular control. For example, to exclude a range of MAC addresses, you might enter `set protocols evpn mac-list test mac-address 00:00:00:00:00:01/24`. Once the MAC list is defined, you can bind it to a specific EVPN instance using the command `set routing-instances instance_name protocols evpn duplicate-mac-detection exclude-list list_name`. This binding ensures that the exclusion logic is applied only where necessary, preventing unintended impacts on unrelated instances.

Action commands such as `set routing-instances instance_name protocols evpn duplicate-mac-detection action block` or `shutdown` allow you to dictate specific actions when a MAC address is marked as a duplicate. These commands offer flexibility in handling duplicate MAC addresses, letting you choose to block the MAC or shut down the port. Show commands like `show evpn mac-lists` and `show evpn instance` extensive help in diagnosing and managing the MAC lists and their bindings.

Please refer to [Feature Explorer](#) for a complete list of the products that support this feature.

Configuration and Management

Use the `set protocols evpn mac-list` statement to create one or more MAC Exclusion Lists. A single *mac-list* can be bound to different EVPN instances. But each EVPN instance can only use one *mac-list*.

```
[edit]
set protocols evpn mac-list test description "Mac list description"
set protocols evpn mac-list test mac-address 00:00:00:00:00:01/24
set protocols evpn mac-list test mac-address 11:00:00:00:00:02/48
set protocols evpn mac-list test mac-address 12:00:00:00:00:03/16
```

Use the `set protocols evpn duplicate-mac-detection exclude-list` statement to bind the list to an EVPN Instance.

QFX5K platforms supporting the default-switch-instance:

```
[edit]
set protocols evpn duplicate-mac-detection exclude-list test
```

Platforms supporting routing instances:

```
[edit]
set routing-instances vs1 protocol evpn duplicate-mac-detection exclude-list test
```

You can view the configured MAC lists and their instance bindings with the following commands:

```
show evpn mac-lists
```

```
show evpn instance extensive
```

Additionally, a commit check in the system prevents configuration conflicts. It ensures you don't configure `duplicate-mac-detection no-mac-suppression` and `duplicate-mac-detection exclude-list` statements simultaneously. These statements serve similar functions but differ in logging. This check maintains your network configuration's integrity.

Usage Scenarios

When using the MAC Exclusion List for Duplicate Detection, it is essential to understand the behavior of the system in different scenarios. For example, if a MAC address is added to the exclusion list after being marked as a duplicate, the duplicate state will be cleared automatically. Conversely, if a MAC address is removed from the exclusion list, it will be subject to duplicate detection again if it moves. This dynamic handling ensures that the MAC Exclusion List remains effective in both preventing false positives and maintaining accurate duplicate detection.

By understanding and utilizing these configurations, you can effectively manage MAC address movements in your network, ensuring stability and performance in environments with frequent legitimate MAC movements.

[Table 1 on page 23](#) lists the expected behaviors for each scenario .

Table 2: MAC Exclusion List Scenarios

Scenario	Marked as Duplicate	Expected Behavior
MAC address added to an exclude list after the MAC is marked as a duplicate.	Yes	Clear the MAC mobility record and the duplicate state on the system.
MAC address removed from an exclude list.	Yes	Clear the MAC mobility history.
MAC address added to an exclude list before the MAC is marked as duplicate.	No	Clear the MAC mobility history
MAC address removed from an exclude list before it reaches the duplicate detect threshold limit.	No	Clear the MAC mobility history

RELATED DOCUMENTATION*duplicate-mac-detection**mac-list (protocols evpn)**show evpn mac-lists***EVPN MAC Pinning Overview**

MAC address pinning for Ethernet VPN (EVPN), including customer edge (CE) interfaces and EVPN over MPLS core interfaces, is supported in all-active mode or in active-standby mode.

When you configure an interface with MAC pinning, the I2ald process adds an 8-octet extension to the address (which implements aspects of Section 7.7 of RFC-7432: MAC Mobility Extended Community). The low-order bit in the flag octet of this structure is the Sticky/static flag. Configuring MAC pinning sets the flag to 1 and designates the address is static.

If you configure MAC pinning on a CE interface, that MAC address cannot be moved to any other CE interface. Similarly, if you configure MAC pinning on an MPLS core interface on a PE device, that MAC address cannot be moved to a different interface on the MPLS core.

CE devices advertise MAC pinned addresses through the I2ald process to remote PE devices. When a PE device learns a MAC-pinned interface address from a CE device, the PE device synchronizes the address, through the control plane, with remote peer PE devices in the EVPN network. Thereafter, if the PE device receives traffic, or an advertised route, from any CE device in the EVPN network that bears the same source MAC address, the receiving device drops the traffic.

A PE device that learns a MAC pinned address via its control plane prefers that address over an address bearing the identical MAC address that is learned from a remote peer PE device, or locally by way of its I2ald from a CE interface.

Before the introduction of MAC address pinning for EVPN, a MAC address on a remote PE device that was learned locally could be aged out. For EVPN MAC pinning, a pinned MAC address does not age out on the remote PE device unless the routing protocol process removes it from the routing table. Similarly, a pinned MAC address persists for the control plane of the remote PE device.

This static interface address cannot be moved unless it is deleted from the routing table of the device on which it is configured.

A MAC address might be considered moved as a result of:

- Misconfiguration
- Configuration on a different Ethernet segment
- Physical movement of a device within a network topology



NOTE: If EVPN is configured in all-active multihoming mode, you must either enable or disable MAC pinning on the multihoming PE device interfaces in the broadcast domain. Also, either enable or disable MAC pinning on all CE device interfaces to avoid inconsistent MAC learning in the EVPN broadcast domain.

If EVPN is configured in active-standby multihoming mode, a MAC pinned address received by the active PE device can be moved to a CE interface on the standby PE device in response to a switchover, if traffic has been running continuously on the CE interface.



CAUTION: Do not enable or disable MAC pinning on an interface while traffic is running.

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
16.2	Starting in Release 16.2, Junos OS enables MAC address pinning for Ethernet VPN (EVPN), including customer edge (CE) interfaces and EVPN over MPLS core interfaces, in both all-active mode or active-standby mode.

RELATED DOCUMENTATION

[EVPN Overview | 1429](#)

[Configuring EVPN Routing Instances | 31](#)

Understanding MAC Pinning

mac-pinning

Configuring EVPN MAC Pinning

The EVPN MAC pinning feature enables you to apply MAC pinning to CE/VPLS interfaces in a bridge domain and to PE interfaces in the core of a service-provider network.

The EVPN MAC pinning feature enables you to make the MAC address associated with a specific interface static. That is, the MAC-pinned address cannot be moved to any other interface in the bridge domain. The MAC-pinned addresses on the interface ages out at the same interval as the default MAC table timeout interval for an MX Series router.

Traffic transmitted with pinned MAC address as source MAC, from any interface in the EVPN domain other than the one on which the MAC address is pinned, will be discarded.

Pinning a MAC address to an interface for EVPN does not require new CLI. Existing CLI enables pinning MAC addresses to CE-device and PE-device interfaces. You can enable MAC Pinning on an interface in an EVPN network with the following command:

```
User@CE1# set switch-options interface interface-name mac-pinning
```

RELATED DOCUMENTATION

[EVPN Overview | 1429](#)

[Configuring EVPN Routing Instances | 31](#)

Creating an Exclusion List for MAC Pinning

SUMMARY

Use an exclusion list to exclude MAC addresses from being pinned in an EVPN network.

IN THIS SECTION

- [Benefits of Using an Exclusion List for MAC pinning | 28](#)

MAC pinning allows you to control the movement of MAC addresses and prevents the creation of network loops by pinning a virtual machine's MAC address to an interface. When you enable MAC pinning on an interface in an EVPN network, the MAC addresses that are learned on the interface are identified as pinned MAC addresses on that interface and in the MAC advertisement message. This prevents virtual machines (MAC addresses) from being moved to another interface in the EVPN network. However, in some cases, you might not want to pin *all* the MAC addresses for an interface; instead, you might want to exclude a few MAC addresses. For example, the Virtual Router Redundancy Protocol (VRRP) provides redundancy with the primary and backup router sharing a virtual MAC address. The network needs to know when the VRRP virtual MAC address has moved from the primary VRRP router to the backup VRRP router, so in this case, you would want to exclude the VRRP virtual MAC address from being pinned.

While MAC pinning is enabled separately on individual interfaces, the exclusion list is configured globally on the device. When you configure an exclusion list, the `I2ald` process verifies the newly learned addresses on the interface against the MAC addresses on the exclusion list. Addresses that are not on the exclusion list are identified as pinned MAC addresses. Addresses that are in the exclusion list are identified as dynamically learned MAC addresses. When the device sends the MAC IP advertisement route message to other devices, the pinned MAC addresses will be identified with the static flag in the extended community set to 1.

When you add a MAC address to the exclusion list that was previously identified as a pinned address, the `I2ald` process removes the pinned MAC address from the MAC address tables, adds it back in as a dynamic nonpinned MAC address, and sends an updated MAC route advertisement messages to the other devices. A similar process happens when you remove a MAC address from the exclusion list.

To configure an exclusion list, include a list of MAC addresses with the `exclusive-mac` parameter at the `[edit protocols l2-learning global-mac-move]` hierarchy level.

For example, if you want to set an exclusion list for MAC addresses 00:00:5E:00:01:01 and 00:00:5E:00:01:20, you include the following configuration. The output for `show bridge mac-table` displays the following

```
User@PE1# set protocols l2-learning global-mac-move exclusive-mac
00:00:5E:00:01:01
User@PE1# set protocols l2-learning global-mac-move exclusive-mac
00:00:5E:00:01:20
```

To remove a MAC address from the exclusion list, use the `delete` configuration mode command at the `[edit protocols l2-learning global-mac-move]` hierarchy. For example, `delete protocols l2-learning global-mac-move exclusive-mac 00:00:5E:00:01:01`.

The following `show bridge mac-table` output shows how MAC addresses are learned by other PE devices and identifies the excluded MAC addresses and pinned MAC addresses.

```
User@PE1> show bridge 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 : EVPN
Bridging domain : bd100, VLAN : 100
  MAC          MAC      Logical      NH      MAC      active
  address      flags    interface  Index  property  source
  00:00:5E:00:01:01  DL      ge-0/0/1.0                #Excluded MAC
Address
  00:50:56:93:f8:ff  DC                1048575      10.1.1.3
  00:50:56:93:c2:60  DC                1048575
10.1.1.3
  00:50:56:93:d9:fb  DP      ge-0/0/1.0                #Pinned MAC Address
```

The following features supports exclusion lists for EVPN MAC pinning:

- EVPN-MPLS, EVPN-VXLAN, EVPN, ELAN, and EVPN E-tree.
- EVPN routing instances and virtual-switch routing instances.
- All-active and single-active EVPN routing instances.
- MAC mobility extended community support for EVPN Type 5 routes.

- Static MAC addresses.
- MC-LAG.

Benefits of Using an Exclusion List for MAC pinning

Exclusion lists allows you to have more flexibility and more control in configuring devices and interfaces on your network.

RELATED DOCUMENTATION

[EVPN MAC Pinning Overview | 23](#)

[Configuring EVPN MAC Pinning | 25](#)

exclusive-mac

global-mac-move

mac-pinning

Static Configuration of MAC-IP Bindings

IN THIS SECTION

- [Configure static MAC-IP bindings | 29](#)

You can statically configure MAC-IP bindings on a logical device. With this feature, you can bind an IP address to one or more MAC address entries to enhance network management and improve communication between infrastructure hosts. This feature is helpful particularly in internet exchange point (IXP) networks, where participant customer edge (CE) devices are:

- Widely recognized
- Static, without transitioning to other provider edge (PE) devices

You can also establish a static link between an IP address and a MAC address for a logical interface inside a bridge domain or VLAN. When you provision a static MAC-IP entry on a PE device, the PE device sends three probes in an exponential backoff pattern. The PE device sends these probes

approximately every 30 seconds until the entry is learned. The probe uses an all-zero sender IP address on the configured PE device.

When the PE device configured with the MAC-IP binding responds to the probe:

1. The PE device learns the MAC-IP binding as static.
2. The probe uses an EVPN Type 2 MAC advertisement route to propagate the MAC-IP bindings to remote PE devices.
3. The remote PE device learns the MAC address of the CE device as dynamic.

If you want to deactivate the probing mechanism for learning MAC-IP bindings, you must configure the `arp-nd-probe-disable` statement at the `[edit protocols l2-learning]` hierarchy. Without probing, the PE device learns both the MAC address and the MAC-IP bindings from network traffic and uses EVPN to propagate the bindings.



NOTE: If you've deployed a multihomed network, you must configure the same static MAC-IP binding on all PE devices in the corresponding ethernet segment identifier (ESI).

Configure static MAC-IP bindings

You can configure static MAC-IP bindings on your logical device with the following command and statements.

- Statitically configure MAC-IP bindings.

- QFX Series:

```
set vlans vlan-name switch-options interface interface-name static-mac-ip ip-address [MAC1 MAC2 ... MACn]
```

- MX Series (Virtual-Switch):

```
set routing-instances routing-instance-name bridge-domains bridge-domain-name bridge-options interface interface-name static-mac-ip ip-address [MAC1 MAC2 ... MACn]
```

- MX Series (EVPN):

```
set routing-instances routing-instance-name protocols evpn interface interface-name static-mac-ip ip-address [MAC1 MAC2 ... MACn]
```



NOTE: You can configure a maximum of eight MAC addresses for each static IP address.

You can also configure the **router** and **override** bits for IPv6 entries. For example:

QFX Series:

```
set vlans vlan-name switch-options interface interface-name static-mac-ip ip-address [MAC1 MAC2 ... MACn]  
<router | override>
```

- Disable default probing when provisioning static MAC-IP entries.

```
set protocols l2-learning arp-nd-probe-disable
```

- Enable logging failed probing of static MAC-IP entries.

```
set protocols l2-learning arp-nd-probe-failed-log
```

You receive notifications via syslog if the probe fails to receive a response, providing useful feedback for troubleshooting.

- Enable GARP/Unsolicited-NA for local and remote static entries.

```
set protocols l2-learning garp-na-enable
```

When learning a new MAC-IP binding on a local PE device, or learning a MAC-IP binding on a remote PE device via EVPN, an unsolicited GARP or NA message is sent to all connected access CE devices. This message enables the connected CE devices to update their ARP or ND caches.

- Disable dynamic learning of MAC-IP entries.

- QFX Series:

```
set vlans vlan-name switch-options drop-unknown-mac-ip
```

- MX Series (Virtual-Switch):

```
set routing-instances routing-instance-name bridge-domains bridge-domain-name bridge-options drop-unknown-mac-ip
```

- MX Series (EVPN):

```
set routing-instances routing-instance-name protocols evpn drop-unknown-mac-ip
```

Configuring Routing Instances for EVPN

IN THIS CHAPTER

- [Configuring EVPN Routing Instances | 31](#)
- [Configuring EVPN Routing Instances on EX9200 Switches | 36](#)
- [MAC-VRF Routing Instance Type Overview | 38](#)
- [EVPN Type 5 Route with VXLAN Encapsulation for EVPN-VXLAN | 47](#)
- [EVPN Type 5 Route with MPLS encapsulation for EVPN-MPLS | 49](#)
- [Understanding EVPN Pure Type 5 Routes | 50](#)
- [Configuring Seamless Type-5 to Type-5 Stitching for EVPN-VXLAN | 55](#)
- [Seamless VXLAN Stitching with Symmetric EVPN Type 2 Routes using Data Center Interconnect | 58](#)
- [Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics | 59](#)
- [EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN | 65](#)
- [Ingress Virtual Machine Traffic Optimization | 75](#)
- [Tracing EVPN Traffic and Operations | 80](#)
- [Migrating From BGP VPLS to EVPN Overview | 81](#)
- [Configuring EVPN over Transport Class Tunnels | 89](#)
- [Example: Configuring EVPN-VPWS over Transport Class Tunnels | 94](#)

Configuring EVPN Routing Instances

You can configure an EVPN instance using a supported Layer 2 (L2) instance type (see *instance-type*) in which you enable the EVPN protocol with other parameters such as an encapsulation type, a route distinguisher, and a route-target. We support several *instance-type* options for EVPN instances, including:

- `evpn` instance type
- `virtual-switch` instance type
- `mac-vrf` instance type

- The default switch instance (in this case, you don't configure a named EVPN instance, and you configure EVPN protocol options at a global level)

Support for different EVPN instance types is platform-specific, so not all platforms support all of these instance-type values. This procedure uses instance-type `evpn`.



CAUTION: Some configuration changes can be what we call *catastrophic* if you perform the changes in an operating network, which means you might see significant disruption in network operation and services. The network disruption can include loss of connectivity among the network devices, and traffic loss while the devices reconverge on changes to network information. When you want to change options in an EVPN routing instance, the change has the potential to impact traffic flow for EVPN as well as other services in the network. As a result, when you change EVPN instance parameters, make sure you use the following procedure to avoid network disruption and traffic loss:

1. Deactivate the routing instance configuration.
2. Change the traffic impacting option.
3. Reactivate the updated routing instance configuration.

For example, you must follow this procedure if you need to change settings such as:

- EVPN protocol settings in a virtual-switch instance—You must configure EVPN protocol settings in a virtual switch instance at the same time you configure the virtual-switch instance type, or deactivate the instance before changing these settings in an existing instance. Otherwise, the device has problems adding EVPN Type 2 (MAC-IP) route entries in the EVPN routing tables.
- The service-type in a MAC-VRF routing instance—When you change the service type of a running instance, the device might incorrectly change the VLAN ID if it is not deactivated before making the change.
- The vlan-id in an EVPN routing instance—Changing the vlan-id without first deactivating the associated EVPN routing instance would be catastrophic.

To configure an EVPN routing instance, complete the following configuration on the PE router (or on the MPLS edge switch [MES] or QFX Series switch) within the EVPN service provider's network:

1. Configure the EVPN routing instance name using the `routing-instances` statement at the [edit] hierarchy level:

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

2. Configure the `evpn` option for the `instance-type` statement at the `[edit routing-instances routing-instance-name]` hierarchy level:

```
instance-type evpn;
```

Alternatively, configure one of the other supported `instance-type` options for EVPN instances, such as `virtual-switch` or `mac-vrf`.



NOTE: For MX Series devices, EX Series, and QFX Series switches, you can include multiple logical interfaces of an Ethernet segment identifier (ESI) across different bridge-domains or VLANs of an EVPN routing instance in all-active mode. However, you cannot include multiple logical interfaces of the same ESI within the same bridge-domain or VLAN.

3. Configure the interfaces for handling EVPN traffic between the MES or PEs and the CE device using the `interface` statement at the `[edit routing-instances routing-instance-name]` hierarchy level:

```
interface interface-name;
```

4. Configure a VLAN identifier for the EVPN routing instance using the `vlan-id` statement at the `[edit routing-instances routing-instance-name]` hierarchy level:



NOTE: For QFX Series, set the VLAN ID to `none`.

```
vlan-id (vlan-id | all | none);
```

5. Configure a route distinguisher on a PE router by including the `route-distinguisher` statement:

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

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 a list of the hierarchy levels at which you can include this statement, see the statement summary 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 providers (ISPs) own or the customer's own AS number.



NOTE: The automatic derivation of the BGP route target (auto-RT) for advertised prefixes is supported on a 2-byte AS number only.

- *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.
6. Configure either import and export policies for the EVPN routing table, or configure the default policies using the *vrf-target* statement configured at the [edit routing-instances *routing-instance-name*] hierarchy level.

See *Configuring Policies for the VRF Table on PE Routers in VPNs*.

7. Configure each EVPN interface for the EVPN routing instance:
 - Configure each interface using the *interface* statement at the [edit routing-instances *routing-instance-name* protocols *evpn*] hierarchy level.
 - Configure interface encapsulation for the CE facing interfaces at the [edit interfaces *interface-name* encapsulation] hierarchy level. Supported encapsulations, except for EX9200 switches and QFX Series switches, are: (ethernet-bridge | vlan-bridge | extended-vlan-bridge). Supported encapsulations for EX9200 switches are: (extended-vlan-bridge | flexible-ethernet-services). Supported encapsulation for QFX Series switches is vxlan.
 - (Optional) Allow the EVPN to establish a connection to the CE device even if the CE device interface encapsulation and the EVPN interface encapsulations do not match by including the *ignore-encapsulation-mismatch* statement at the [edit routing-instances *routing-instance-name* protocols *evpn* interface *interface-name*] hierarchy level.
 - (Optional) (Not available on EX9200 switches) Specify a static MAC address for a logical interface in a bridge domain using the *static-mac* statement at the [edit routing-instances *routing-instance-name* protocols *evpn* interface *interface-name*] hierarchy level.
8. Specify the maximum number of media access control (MAC) addresses that can be learned by the EVPN routing instance by including the *interface-mac-limit* statement.

You can configure the same limit for all interfaces configured for a routing instance by including this statement at the [edit routing-instances *routing-instance-name* protocols *evpn*] hierarchy level. You can

also configure a limit for a specific interface by including this statement at the [edit routing-instances *routing-instance-name* protocols evpn interface *interface-name*] hierarchy level.

By default, packets with new source MAC addresses are forwarded after the MAC address limit is reached. You can alter this behavior by including the `packet-action drop` statement at either the [edit routing-instances *routing-instance-name* protocols evpn interface-mac-limit] or the [edit routing-instances *routing-instance-name* protocols evpn interface *interface-name*] hierarchy level. If you configure this statement, packets from new source MAC addresses are dropped once the configured MAC address limit is reached.

9. Specify the MPLS label allocation setting for the EVPN by including the *label-allocation* statement with the per-instance option at the [edit routing-instances *routing-instance-name* protocols evpn] hierarchy level.

If you configure this statement, one MPLS label is allocated for the specified EVPN routing instance.

10. Enable MAC accounting for the EVPN by including the *mac-statistics* statement at the [edit routing-instances *routing-instance-name* protocols evpn] hierarchy level.
11. Specify the number of addresses that can be stored in the MAC routing table using the *mac-table-size* statement at the [edit routing-instances *routing-instance-name* protocols evpn] hierarchy level.

You can optionally configure the *packet-action drop* option to specify that packets for new source MAC addresses be dropped once the MAC address limit is reached. If you do not configure this option, packets for new source MAC addresses are forwarded.

12. Disable MAC learning by including the *no-mac-learning* statement at either the [edit routing-instances *routing-instance-name* protocols evpn] hierarchy level to apply this behavior to all of the devices configured for an EVPN routing instance or at the [edit routing-instances *routing-instance-name* protocols evpn interface *interface-name*] hierarchy level to apply this behavior to just one of the CE devices.
13. On MX Series and ACX5448 devices, while configuring EVPN-MPLS, you need to include the `network-services enhanced-ip` CLI statement at the [edit chassis] hierarchy level. On ACX Series devices running on Junos Evolved OS, the `network-services enhanced-ip` CLI statement is enabled by default.

RELATED DOCUMENTATION

[Configuring Policies for the VRF Table on PE Routers in VPNs](#)

[Configuring Routing Instances on PE Routers in VPNs](#)

[Tracing EVPN Traffic and Operations](#) | 80

Configuring EVPN Routing Instances on EX9200 Switches

To configure an EVPN routing instance, complete the following configuration on the PE router (or on the MPLS edge switch) within the EVPN service provider's network:

1. Configure the EVPN routing instance name using the `routing-instances` statement at the [edit] hierarchy level:

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

2. Configure the `evpn` option for the `instance-type` statement at the [edit `routing-instances routing-instance-name`] hierarchy level:

```
instance-type evpn;
```

3. Configure the interfaces for handling EVPN traffic between the MES and the CE device using the `interface` statement at the [edit `routing-instances routing-instance-name`] hierarchy level:

```
interface interface-name;
```

4. Configure a VLAN identifier for the EVPN routing instance using the `vlan-id` statement at the [edit `routing-instances routing-instance-name`] hierarchy level:

```
vlan-id (vlan-id | all | none);
```

5. Configure a route distinguisher on a PE router by including the `route-distinguisher` statement:

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

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 a list of the hierarchy levels at which you can include this statement, see the statement summary 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.
6. Configure either import and export policies for the EVPN routing table, or configure the default policies using the *vrf-target* statement configured at the [edit routing-instances *routing-instance-name*] hierarchy level.

See *Configuring Policies for the VRF Table on PE Routers in VPNs*.

7. Configure each EVPN interface for the EVPN routing instance:
- Configure interface encapsulation for the CE facing interfaces at the [edit interfaces *interface-name* encapsulation] hierarchy level. . Supported encapsulations for EX9200 switches are: (extended-vlan-bridge | flexible-ethernet-services | vlan-bridge).
 - Configure *vlan-bridge* encapsulation on the logical interface at the [edit interfaces *interface-name* flexible-vlan-tagging encapsulation flexible-ethernet-services unit 0 encapsulation] hierarchy level.
 - (Optional) Allow the EVPN to establish a connection to the CE device even if the CE device interface encapsulation and the EVPN interface encapsulations do not match by including the *ignore-encapsulation-mismatch* statement at the [edit routing-instances *routing-instance-name* protocols evpn interface *interface-name*] hierarchy level.
8. Specify the maximum number of media access control (MAC) addresses that can be learned by the EVPN routing instance by including the *interface-mac-limit* statement.

You can configure the same limit for all interfaces configured for a routing instance by including this statement at the [edit routing-instances *routing-instance-name* protocols evpn] hierarchy level. You can also configure a limit for a specific interface by including this statement at the [edit routing-instances *routing-instance-name* protocols evpn interface *interface-name*] hierarchy level.

By default, packets with new source MAC addresses are forwarded after the MAC address limit is reached. You can alter this behavior by including the *packet-action drop* statement at either the [edit routing-instances *routing-instance-name* protocols evpn interface-mac-limit] or the [edit routing-instances *routing-instance-name* protocols evpn interface *interface-name*] hierarchy level. If you configure this statement, packets from new source MAC addresses are dropped once the configured MAC address limit is reached.

9. Enable MAC accounting for the EVPN by including the *mac-statistics* statement at the [edit routing-instances *routing-instance-name* protocols evpn] hierarchy level.
10. Specify the number of addresses that can be stored in the MAC routing table using the *mac-table-size* statement at the [edit routing-instances *routing-instance-name* protocols evpn] hierarchy level.

You can optionally configure the *packet-action* drop option to specify that packets for new source MAC addresses be dropped once the MAC address limit is reached. If you do not configure this option, packets for new source MAC addresses are forwarded.

11. Disable MAC learning by including the *no-mac-learning* statement at either the [edit routing-instances *routing-instance-name* protocols evpn] hierarchy level to apply this behavior to all of the devices configured for an EVPN routing instance or at the [edit routing-instances *routing-instance-name* protocols evpn interface *interface-name*] hierarchy level to apply this behavior to just one of the CE devices.

RELATED DOCUMENTATION

Configuring Policies for the VRF Table on PE Routers in VPNs

Configuring Routing Instances on PE Routers in VPNs

[Tracing EVPN Traffic and Operations | 80](#)

MAC-VRF Routing Instance Type Overview

IN THIS SECTION

- [Benefits of the MAC-VRF Routing Instance Type | 38](#)
- [MAC-VRF Instances Enable Customer-Specific VRF Tables | 39](#)
- [MAC-VRF Instances Enable Common EVPN Configuration across Platforms | 41](#)
- [MAC-VRF Instance Show Commands and Aliases | 42](#)
- [Usage and Behavior Notes | 45](#)

Use the MAC-VRF routing instance type to configure multiple customer-specific EVPN instances (EVIs), each of which can support a different EVPN service type. You configure a MAC-VRF instance with the *mac-vrf* statement at the [edit routing-instances *mac-vrf-instance-name* instance-type] hierarchy. With this configuration, you can create customer-specific virtual routing and forwarding (VRF) tables.

Benefits of the MAC-VRF Routing Instance Type

- Customer-specific VRF tables

- Consistent configuration across supported router and switch platforms within an EVPN-VXLAN network or an EVPN-MPLS network.
- Configuration alignment with [RFC 7432](#)

MAC-VRF Instances Enable Customer-Specific VRF Tables

When you configure a MAC-VRF routing instance, you can isolate or group routing and forwarding traffic by customer. In fact, you can manage MAC-VRF instances around multiple schemes within an organization, including by department, division, or geographic location. The traffic belonging to any one MAC-VRF instance cannot interact with traffic from any other MAC-VRF instances.

MAC-VRF Instance Service Types

MAC-VRF instances follow the EVPN instance design in [RFC 7432](#), which includes the three service types in [Table 3 on page 40](#). When you create a MAC-VRF instance, you must configure one of the supported service types using the service-type statement at the `[edit routing-instances mac-vrf-instance-name]` hierarchy level.



CAUTION: Some configuration changes can be what we call *catastrophic* if you perform the changes in an operating network, which means you might see significant disruption in network operation and services. The network disruption can include loss of connectivity among the network devices, and traffic loss while the devices reconverge on changes to network information. When you want to change options in an EVPN routing instance, the change has the potential to impact traffic flow for EVPN as well as other services in the network. As a result, when you change EVPN instance (EVI) parameters, make sure you use the following procedure to avoid network disruption and traffic loss:

When you want to change a traffic impacting option under a routing instance, use the following procedure.

1. Deactivate the routing instance configuration.
2. Change the traffic impacting option.
3. Reactivate the updated routing instance configuration.

For example, follow this procedure if you need to change settings such as:

- The `service-type` in a MAC-VRF routing instance—When you change the service type of a running instance, the device might incorrectly change the VLAN ID if it is not deactivated before making the change.
- The `vlan-id` in an EVPN routing instance—Changing the `vlan-id` without first deactivating the associated EVPN routing instance would be catastrophic.

Table 3: MAC-VRF Instance Service Type Options

Service Type Option	Description
vlan-aware	<p>You can configure the MAC-VRF EVPN instance to correspond to one or more VLANs. The MAC-VRF instance maintains one bridging table per VLAN.</p> <p>NOTE: By default in EVPN-VXLAN MAC-VRF instances with this service type, the device extends all VXLAN network identifiers (VNIs) in the instance. You don't need to explicitly configure the <i>extended-vni-all</i> statement. You can configure the <i>extended-vni-list</i> statement if you want to extend only a subset of the VNIs in the instance.</p>
vlan-based	<p>You can configure the MAC-VRF EVPN instance to correspond to a single VLAN and corresponding bridging table.</p> <p>NOTE: If the VLAN maps to different VLAN IDs per Ethernet segment, then you must configure each device in the EVPN fabric to perform VLAN ID translation on packets destined for the Ethernet segment.</p>
vlan-bundle	<p>You can configure the MAC-VRF EVPN instance to correspond to multiple VLANs that share the same bridging table. MAC addresses must be unique across all VLANs in the instance. This service type also doesn't support VLAN translation (you can configure each VLAN with one unique VLAN ID).</p> <p>NOTE: If all VLANs for a port are part of the same VLAN bundle service, the service is called a <i>port-based service</i>.</p>

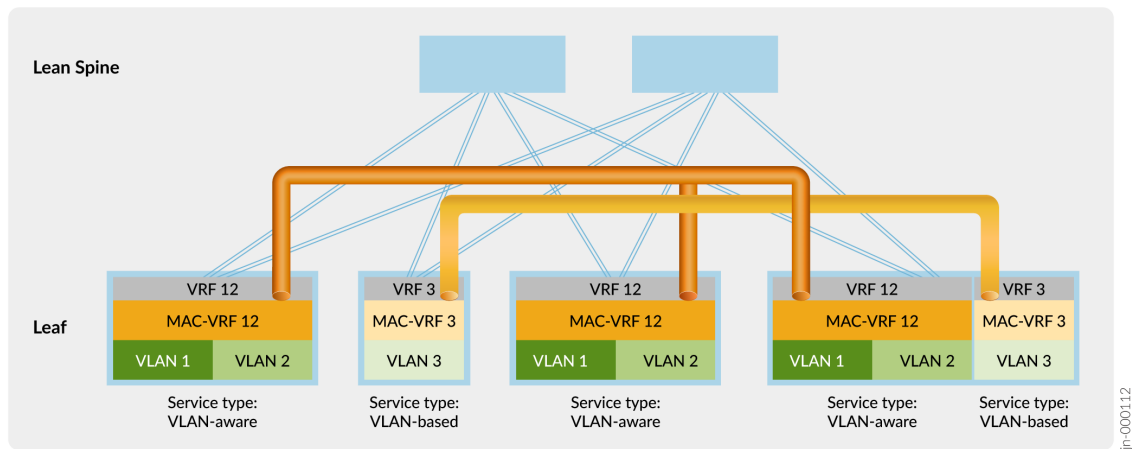
Flexible Configuration Options at Layer 2 and Layer 3

With MAC-VRF instances, you have more flexible configuration options for customers at both Layer 2 (L2) and Layer 3 (L3):

- At L2: You can configure different service types in different MAC-VRF instances on the same device.
- At L3: You can configure a VRF instance that corresponds to the VLAN or VLANs in a single MAC-VRF instance. You can also configure a VRF instance that spans the VLANs in multiple MAC-VRF instances.

For example, the following figure shows an edge-routed bridging (ERB) EVPN-VXLAN fabric. The leaf devices establish VXLAN tunnels that maintain L2 and L3 separation between a customer on VLAN 1 and VLAN 2, and a customer on VLAN 3. MAC-VRF 12 and MAC-VRF 3 separate the customers at L2. VRF 12 and VRF 3 separate the customers at L3. The figure also shows that you can configure multiple MAC-VRF instances on the same device with different service types.

Figure 4: EVPN-VXLAN Fabric with MAC-VRF and VRF Instances for L2 and L3 Customer Separation



MAC-VRF Instances Enable Common EVPN Configuration across Platforms

On devices that support EVPN configurations, the configuration methods vary from platform to platform. For example:

- On MX Series routers, before MAC-VRF instance support, you could configure EVPN instances using instances of type `virtual-switch` with statements in the following hierarchies:
 - `bridge-domains`

- `routing-instances`
- On QFX Series switches, before MAC-VRF instance support, you could configure EVPN instances using the default switch instance with statements in the following hierarchies:
 - `ethernet-switching`
 - `routing-instances`
- On ACX Series, QFX Series, and PTX series platforms running Junos OS Evolved that support EVPN, on which EVPN feature support was introduced with MAC-VRF EVPN instances, you configure EVPN features using only MAC-VRF instances.

This disparity can be confusing and lead to errors configuring EVPN across different platforms.

We introduced the `mac-vrf` routing instance type in Junos OS Release 20.4R1 on some Junos OS platforms. Other platforms, including Junos OS Evolved platforms, have gained MAC-VRF instance support in subsequent releases. You can use `mac-vrf` instances to create a common EVPN configuration on all supported platforms. The common configuration hierarchy across routing and switching platforms also brings our implementation of MAC-VRF into compliance with [RFC 7432](#). RFC compliance enables our MAC-VRF implementation to work well in data center, service provider, and public cloud environments.



NOTE: Platforms that support MAC-VRF instances as well as other instance types to configure EVPN instances can have different instance configurations coexist in an active network. EVPN fabrics can comprise a combination of devices that support MAC-VRF instances and devices that use other instance types for EVPN.

MAC-VRF Instance Show Commands and Aliases

Some `mac-vrf` show commands apply only to specific platforms. For example, the MAC-VRF command `show mac-vrf mac-table age` (and the corresponding `show bridge mac-table age` command) applies only to MX Series routers. If you issue the `show mac-vrf mac-table age` command on a QFX Series switch, the output is blank without any error indication.

See [Table 4 on page 43](#) and [Table 5 on page 44](#) for references to the commands that provide information about EVPN instances. The common MAC-VRF forms of these commands are aliases for the platform-specific commands. Use the listed commands as follows:

- You can use the MAC-VRF versions of these commands in the first column to display EVPN instance information on any platforms with MAC-VRF EVPN configurations.
- Use the MAC-VRF commands in the first column on ACX Series, QFX Series, and PTX series platforms running Junos OS Evolved, which support EVPN only with MAC-VRF instance configurations.

- Use the commands in the second column on MX Series routers and the EX9200 line of switches running Junos OS, for example, when you configure EVPN instances using instance type virtual-switch.
- Use the commands in the third column on QFX Series and EX Series switches running Junos OS, for example, when you configure EVPN instances using the default switch instance.

Table 4: List of MAC-VRF Forwarding Commands by Platform

MAC-VRF Command	MX Series Routers and the EX9200 Line of Switches	QFX Series and EX Series Switches
show mac-vrf forwarding flood	show bridge flood	show ethernet-switching flood
show mac-vrf forwarding flood-group	show l2-learning flood-group	show ethernet-switching flood-group
show mac-vrf forwarding global-information	show l2-learning global-information	show ethernet-switching global-information
show mac-vrf forwarding global-mac-count	show l2-learning global-mac-count	show ethernet-switching global-mac-count
show mac-vrf forwarding global-mac-ip-count	show l2-learning global-mac-ip-count	show ethernet-switching global-mac-ip-count
show mac-vrf forwarding instance	show l2-learning instance	show ethernet-switching instance
show mac-vrf forwarding instance-mapping	–	–
show mac-vrf forwarding interface	show l2-learning interface	show ethernet-switching interface
show mac-vrf forwarding mac-ip-table	show bridge mac-ip-table	show ethernet-switching mac-ip-table
show mac-vrf forwarding mac-table	show bridge mac-table	show ethernet-switching table
show mac-vrf forwarding mgrp-policy	show l2-learning mgrp-policy	show ethernet-switching mgrp-policy

Table 4: List of MAC-VRF Forwarding Commands by Platform *(Continued)*

MAC-VRF Command	MX Series Routers and the EX9200 Line of Switches	QFX Series and EX Series Switches
show mac-vrf forwarding statistics	show bridge statistics show evpn statistics	show ethernet-switching
show mac-vrf forwarding vlans	show bridge domains	show ethernet-switching vlans
show mac-vrf forwarding vxlan-tunnel-endpoint esi	show l2-learning vxlan-tunnel-endpoint esi	show ethernet-switching vxlan-tunnel-end-point esi
show mac-vrf forwarding vxlan-tunnel-endpoint remote	show l2-learning vxlan-tunnel-endpoint remote	show ethernet-switching vxlan-tunnel-end-point remote
show mac-vrf forwarding vxlan-tunnel-endpoint svlnh	show l2-learning vxlan-tunnel-endpoint svlnh	show ethernet-switching vxlan-tunnel-end-point svlnh

Table 5: List of MAC-VRF Routing Commands

MAC-VRF Command	Equivalent show evpn Commands
show mac-vrf routing database	show evpn database
show mac-vrf routing igmp-snooping database	show evpn igmp-snooping database
show mac-vrf routing instance	show evpn instance
show mac-vrf routing mld-snooping database	show evpn mld-snooping database
show mac-vrf routing multicast-snooping status	show evpn multicast-snooping status
show mac-vrf routing p2mp	show evpn p2mp



NOTE: We have integrated the syntax of the commands from the show mac-vrf routing hierarchy into the existing show evpn command documentation. The links in [Table 5 on page 44](#) lead to the existing show evpn commands.

Usage and Behavior Notes

Read the following notes to know more about using MAC-VRF routing instances and the observed behaviors of those MAC-VRF routing instances.

MAC-VRF Instance Type Support Limitations

Note the following support limitations for MAC-VRF instances:

- We don't support the MAC-VRF instance type for configuring EVPN instances with EVPN-MPLS on MX Series routers.
- We don't support changing the encapsulation setting at the `[edit routing-instances <instance-name> protocols evpn]` hierarchy on ACX Series routers. If you need to configure a different encapsulation for your EVPN routing instance, then you must delete the current routing instance and create a new routing instance with the desired encapsulation.

Shared VTEP Tunnels in EVPN-VXLAN Fabrics with Multiple MAC-VRF Routing Instances

Lower capacity devices might have problems with VTEP scaling when the configuration uses multiple MAC-VRF instances. To prevent this problem, some platforms, such as QFX5130 switches, QFX5700 switches, and ACX7100 routers, enable the shared VTEP tunnels feature in the default configuration for MAC-VRF instance VXLAN tunnels. With this feature enabled for VXLAN routing, the device minimizes the number of next-hop entries to reach remote VTEPs.

Switches in the QFX5000 line running Junos OS are lower capacity devices that don't enable shared tunnels by default. As a result, when you configure MAC-VRF instances on those devices, we require that you configure the shared tunnels feature. To configure shared tunnels, set the `shared-tunnels` option at the global `[edit forwarding-options evpn-vxlan]` hierarchy.



NOTE: When you configure the `shared-tunnels` option, you must reboot the device for the setting to take effect.

This statement is optional on devices that can usually handle higher VTEP scaling, such as the QFX10000 line of switches. This statement is not available on other devices that don't need it in scaled EVPN-VXLAN fabrics, such as MX series routers. The shared VTEP tunnel feature operates locally on the device, so different platforms in the same network can interoperate whether the devices use this option or not.

MAC-VRF show commands display shared tunnel VTEP interfaces as `vtep-index.shared-tunnel-unit`, where:

- *index* is the VTEP index associated with the MAC-VRF routing instance.

- *shared-tunnel-unit* is the unit number associated with the shared tunnel remote VTEP logical interface.

For example:

```
vtep-7.32772
```

VLANs, Forwarding Instances, and Overlapping VLANs with MAC-VRF Routing Instances

Each supported platform has its own support framework for VLANs and forwarding instances, and how VLANs can overlap.

- EX4400, QFX5100, QFX5110, QFX5120, QFX5200, QFX5130-32CD, and QFX5700 switches, and PTX10001-36MR, PTX10004, PTX10008, PTX10016 routers

These devices support only one forwarding instance (*default-switch*). You can't configure overlapping VLANs within a single MAC-VRF routing instance or across different MAC-VRF routing instances.

However, on some of these platforms, you can alternatively use VLAN translation to support overlapping VLANs. See *vlan-rewrite* and ["Overlapping VLAN Support Using VLAN Translation in EVPN-VXLAN Networks" on page 668](#).

- ACX7100-32C and ACX7100-48L routers, and the QFX10000 line of switches

The QFX10000 line of switches supports multiple forwarding instances using the *forwarding-instance* option at the [edit routing-instances *mac-vrf-instance-name*] hierarchy. This statement maps a MAC-VRF instance to a forwarding instance corresponding to the configured identifier. Starting in Junos OS Evolved Release 22.3R1, ACX7100-32C and ACX7100-48L routers also support multiple routing instances using the *forwarding-instance identifier* option.

On these platforms, you can configure overlapping VLANs across multiple MAC-VRF routing instances if you've mapped each routing instance to a unique forwarding instance. You can also configure multiple MAC-VRF routing instances to use one forwarding instance. You can't configure overlapping VLANs within a single MAC-VRF routing instance or across routing instances each of which maps to the same forwarding instance. If you don't configure the forwarding instance, the MAC-VRF routing instance uses the default forwarding instance (*default-switch*).

ACX7100-32C and ACX7100-48L routers alternatively support overlapping VLANs using explicit or implicit VLAN normalization. You can also use VLAN normalization to support overlapping VLANs on these routers in releases before Junos OS Evolved Release 22.3R1. See ["Overlapping VLAN Support Using Multiple Forwarding Instances or VLAN Normalization" on page 674](#).

- MX Series routers and the EX9200 line of switches

You can configure overlapping VLANs across multiple MAC-VRF routing instances. You cannot configure overlapping VLANs within a single MAC-VRF routing instance.

On these platforms, each MAC-VRF routing instance that you configure automatically maps to its own forwarding instance. We don't support the `forwarding-instance` option.



NOTE: In the default configuration, devices include a default VLAN with a VLAN ID value of 1 associated with the default forwarding instance. Because VLAN IDs must be unique in a forwarding instance, if you want to configure a VLAN with VLAN ID 1 in a MAC-VRF instance that uses the default forwarding instance, you must reassign the VLAN ID of the default VLAN to a value other than 1. For example:

```
set vlans default vlan-id 4094
set routing-instances mac-vrf-instance-name vlans vlan-name vlan-id 1
```

Extended VNI List Behavior

You can configure extended virtual network identifier (VNI) lists within any MAC-VRF routing instance at the `edit routing-instances mac-vrf routing instance name protocols evpn hierarchy` level. The `extended-vni-list` keyword is an optional configuration element. By default, the device extends all VNI (whether in a VNI list or not) within the MAC-VRF routing instance. If you configure a specific VNI list, then you can extend only those VNIs that are in the list.

RELATED DOCUMENTATION

show mac-vrf forwarding mac-table

show mac-vrf forwarding vxlan-tunnel-end-point remote

show evpn mac-ip-table

EVPN Type 5 Route with VXLAN Encapsulation for EVPN-VXLAN

IN THIS SECTION

- [Blocking Asymmetric EVPN Type 5 Routes](#) | 48

EVPN is a flexible solution that uses Layer 2 (L2) overlays to interconnect multiple edges (virtual machines) within a data center. Traditionally, the data center is built as a flat L2 network with issues such as flooding, limitations in redundancy and provisioning, and high volumes of MAC addresses learned, which cause churn at node failures. EVPN is designed to address these issues without disturbing flat MAC connectivity.

Virtual extensible local area network (VXLAN) overlay technology encapsulates MAC frames into a UDP header at L2. Communication is established between two virtual tunnel endpoints (VTEPs). VTEPs encapsulate the virtual machine traffic into a VXLAN header, as well as strip off the encapsulation. Virtual machines can only communicate with each other when they belong to the same VXLAN segment. A 24-bit virtual network identifier (VNID) uniquely identifies the VXLAN segment. This enables having the same MAC frames across multiple VXLAN segments without traffic crossover. Multicast in VXLAN is implemented as Layer 3 (L3) multicast, in which endpoints subscribe to groups.

When a bridge domain (BD) is not L2 extended across data centers (DCs), the IP subnet belonging to the BD is confined within a single DC. If all BDs within each DC network satisfy this requirement, there is no longer a need to advertise MAC+IP routes for each tenant between data centers, because host routes for the tenants can be aggregated. As a result, the L2 inter-DC connectivity issue can be simply transformed to an inter-DC L3 IP prefix reachability issue.

Starting with Junos OS Release 17.1, the EVPN Type 5 IP prefix route advertises the IP prefixes between the DCs. Unlike the Type-2 EVPN MAC advertisement route, the EVPN Type 5 IP prefix route separates the host MAC address from its IP address and cleanly advertises an IP prefix for the BD.

We also support:

- Inter-DC connectivity with VXLAN encapsulation for EVPN-VXLAN using the EVPN Type 5 IP prefix route.

Each BD within a DC is not L2 extended. If EVPN-VXLAN is enabled between the DC GW (Data Center Gateway) routers and top-of-rack devices (ToRs) while providing inter-DC connectivity, the spine, which acts as a DC GW router, performs L3 routing and IRB functions.

- Inter-pod connectivity with VXLAN encapsulation by using the EVPN Type 5 IP prefix route.

The IP prefix route VXLAN inter-pod connectivity does not address the L2 extension problem when a BD is stretched across different pods. The spines that provide the inter-pod connectivity perform L3 routing and integrated routing and bridging (IRB) functions.

Blocking Asymmetric EVPN Type 5 Routes

While Juniper devices support asymmetric routes in EVPN Type 5 routes, processing asymmetric EVPN Type 5 routes consume Packet Forwarding Engine (PFE) resources. In some cases, you may want to conserve PFE resources and block asymmetric EVPN Type 5 routes. When you block asymmetric EVPN Type 5 routes, the local device examines the incoming EVPN Type 5 route and rejects the route when

the VNI in the ingress route differs from the locally configured VNI. The device still installs the route in the `bgp.evpn.0` table, but rejects the routes (doesn't install them) in the `routing-instance-name.inet.0` table.

To block asymmetric EVPN Type 5 routes in a virtual routing and forwarding (VRF) instance where EVPN Type 5 routes are enabled, include following statement:

```
user@device1# set routing-instances routing-instance-name protocols evpn ip-prefix-routes reject-
asymmetric-vni;
```



NOTE:

ACX7100 routers running Junos OS Evolved Release 24.4R1 and earlier do not support asymmetric EVPN Type 5 routes. When you configure EVPN Type 5 routes on an ACX7100 router running Junos OS Evolved Release 24.4R1 and earlier, you must also configure the `reject-asymmetric-vni` option at the `[edit routing-instances routing-instance-name protocols evpn ip-prefix-routes]` hierarchy level in the same routing instance.

RELATED DOCUMENTATION

[EVPN Type 5 Route with MPLS encapsulation for EVPN-MPLS](#) | 49

ip-prefix-routes

EVPN Type 5 Route with MPLS encapsulation for EVPN-MPLS

EVPN is a flexible solution that uses Layer 2 overlays to interconnect multiple edges (virtual machines) within a data center. Traditionally, the data center is built as a flat Layer 2 network with issues such as flooding, limitations in redundancy and provisioning, and high volumes of MAC addresses learned, which cause churn at node failures. EVPN is designed to address these issues without disturbing flat MAC connectivity.

The MPLS infrastructure allows you to take advantage of the MPLS functionality provided by the Junos operating system (Junos OS), including fast reroute, node and link protection, and standby secondary paths.

Starting with Junos OS Release 17.1, to support the interconnection through EVPN Type 5 route in the metro service application using EVPN/MPLS, an MPLS tunnel is required instead of a VXLAN tunnel.



NOTE: L3VPN forwarding based on pure Type 5 without overlay next-hop is only supported.

In the control plane EVPN Type 5 is used to advertise IP prefix for inter-subnet connectivity across metro peering points. To reach the end host through the connectivity provided by the EVPN Type 5 prefix route, data packets are sent out as an IP packet encapsulated in the MPLS across the metro peering points.

RELATED DOCUMENTATION

[EVPN Type 5 Route with VXLAN Encapsulation for EVPN-VXLAN](#) | 47

Understanding EVPN Pure Type 5 Routes

IN THIS SECTION

- [Defining EVPN-VXLAN Route Types](#) | 51
- [Implementing Pure Type 5 Routes in an EVPN-VXLAN Environment](#) | 52
- [Best Practices and Caveats](#) | 54
- [Release Information](#) | 54

Ethernet VPN (EVPN) offers an end-to-end solution for data center Virtual Extensible LAN (VXLAN) networks. A main application of EVPN is Data Center Interconnect (DCI), which provides the ability to extend Layer 2 connectivity between different data centers. EVPN uses the concept of route types to establish sessions between the provider edge and the customer edge. There are many route types. A Type 5 route, also called the IP prefix route, is used to communicate between data centers (DC) when the Layer 2 connection does not extend across DCs and the IP subnet in a Layer 2 domain is confined within a single DC. In this scenario, the Type 5 route enables connectivity across DCs by advertising the IP prefixes assigned to the VXLANs confined within a single DC. Data packets are sent as Layer 2 Ethernet frames encapsulated in the VXLAN header. Additionally, the gateway device for the DC must be able to perform Layer 3 routing and provide IRB functionality.

A pure Type 5 route operates without an overlay next hop or a Type 2 route for recursive route resolution. With pure Type 5 routing, the Type 5 route is advertised with the MAC extended community

so that the Type 5 route provides all necessary forwarding information required for sending VXLAN packets in the data plane to the egress network virtual endpoint. There is no need to use an IP address as an overlay next hop to interconnect Layer 3 virtual routing and forwarding (VRF) routes sitting in different data centers. Because no Type 2 routes are used for route recursive resolution, this provisioning model is also called the IP-VRF-to-IP-VRF model without a core-facing IRB interface.

Defining EVPN-VXLAN Route Types

The EVPN-VXLAN route types are:

- **Type 1 route, Ethernet autodiscovery route**—Type 1 routes are for networkwide messages. Ethernet autodiscovery routes are advertised on a per end virtual identifier (EVI) and per Ethernet segment identifier (ESI) basis. The Ethernet autodiscovery routes are required when a customer edge (CE) device is multihomed. When a CE device is single-homed, the ESI is zero. This route type is supported by all EVPN switches and routers.

An ESI can participate in more than one broadcast domain; for example, when a port is trunked. An ingress provider edge (PE) device that reaches the MAC on that ESI must have Type 1 routes to perform split horizon and fast withdraw. Therefore, a Type 1 route for an ESI must reach all ingress PE devices importing a virtual network identifier (VNI) or tag (broadcast domains) in which that ESI is a member. The Junos OS supports this by exporting a separate route target for the Type 1 route.

- **Type 2 route, MAC with IP advertisement route**—Type 2 routes are per-VLAN routes, so only PEs that are part of a VNI need these routes. EVPN allows an end host's IP and MAC addresses to be advertised within the EVPN Network Layer reachability information (NLRI). This allows for control plane learning of ESI MAC addresses. Because there are many Type 2 routes, a separate route-target auto-derived per VNI helps to confine their propagation. This route type is supported by all EVPN switches and routers.
- **Type 3 route, inclusive multicast Ethernet tag route**—Type 3 routes are per-VLAN routes; therefore, only PE devices that are part of a VNI need these routes. An inclusive multicast Ethernet tag route sets up a path for broadcast, unknown unicast, and multicast (BUM) traffic from a PE device to the remote PE device on a per-VLAN, per-EVI basis. Because there are many Type 3 routes, a separate route-target auto-derived per VNI helps in confining their propagation. This route type is supported by all EVPN switches and routers.
- **Type 4 route, Ethernet segment Route**—An Ethernet segment identifier (ESI) allows a CE device to be multihomed to two or more PE devices—in single/active or active/active mode. PE devices that are connected to the same Ethernet segment discover each other through the ESI. This route type is supported by all EVPN switches and routers.
- **Type 5 route, IP prefix Route**—An IP prefix route provides encoding for inter-subnet forwarding. In the control plane, EVPN Type 5 routes are used to advertise IP prefixes for inter-subnet connectivity across data centers. To reach a tenant using connectivity provided by the EVPN Type 5 IP prefix

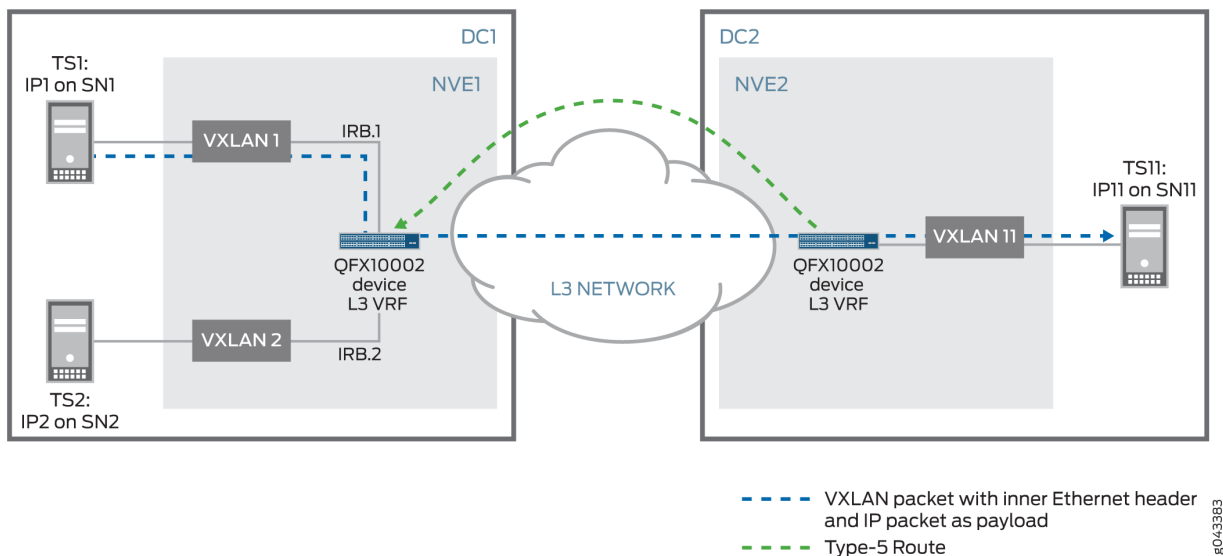
route, data packets are sent as Layer 2 Ethernet frames encapsulated in the VXLAN header over the IP network across the data centers.

- Type 6 route, selective multicast Ethernet tag routes.
- Type 7 route, network layer reachability information (NLRI) to sync IGMP joins.
- Type 8 route, NLRI to sync IGMP leaves.

Implementing Pure Type 5 Routes in an EVPN-VXLAN Environment

You can use EVPN pure Type 5 routes to communicate between data centers through a Layer 3 network. See [Figure 5 on page 52](#). A unified EVPN control plane accomplishes L3 route advertisement between multiple data center locations so that you do not have to rely on an additional L3 VPN protocol family. On the customer edge (CE), hosts such as servers, storage devices, or any bare-metal devices are attached to leaf switches on the provider edge. Between those leaf devices, an MP-BGP session is established for EVPN routes to be used in the overlay control protocol. .

Figure 5: EVPN-VXLAN Connection with Pure Type 5 Route Between Two Data Centers



A global unique virtual network identifier (VNI) is provisioned for each customer L3 VRF and identifies the customer L3 VRF at the egress. A chassis MAC is used as the inner destination MAC (DMAC) for the VXLAN packet. The chassis MAC is shared among different customer L3 VRF instances



NOTE: When a virtual machine (VM) moves from one QFX10000 data center to another, a Type 5 route no longer works. This is because both the VXLAN and IP subnet that belong to the VM are no longer confined within a single data center.



NOTE: For an example of communicating within a single data center without Type 5 routing, see ["Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric" on page 771.](#)

Understanding Pure Type 5-Route Forwarding

Pure Type 5 route forwarding is also called the IP-VRF-to-IP-VRF (virtual routing and forwarding) model. In IP-based computer networks, Layer 3 VRF allows multiple instances of a routing table to coexist within the same router at the same time. Because the routing instances are independent, the same or overlapping IP addresses can be used without conflicting with each other. In this scenario, for a given tenant, such as an IP VPN service, a network virtualization edge (NVE) has one MAC VRF, which consists of multiple VXLANs (one VXLAN per VLAN). The MAC VRFs on an NVE for a given tenant are associated with an IP VRF corresponding to that tenant (or IP VPN service) through their IRB interfaces. A global unique VNI is provisioned for each customer Layer 3 VRF. The VNI is used to identify the Layer 3 VRF for the customer on each data center.

Understanding EVPN Pure Type 5 Routes and Local Preferences


On QFX10000 switches running Junos OS Release 15.1X53-D65 or later, the local preference setting for an Ethernet VPN (EVPN) pure Type 5 route is inherited by IP routes that are derived from the EVPN type 5 route. Further, when selecting an IP route for incoming traffic, the QFX10000 switches consider the local preference of the route. A benefit of the QFX10000 switches including local preference in their route selection criteria is that you can set up a policy to manipulate the local preference, thereby controlling which route the switch selects.


Advantages of Using EVPN Pure Type 5 Routing


There are two main advantages to using EVPN pure Type 5 routing:

- There is no need to exchange all host routes between data center locations. This results in smaller requirements for the routing information base (RIB), also known as the routing table, and the forwarding information base (FIB), also known as the forwarding table, on DCI equipment.
- There is no need to use multiple protocol families, such as both EVPN and an L3 VPN, to advertise L2 and L3 reachability information.

Best Practices and Caveats

**BEST PRACTICE:** You can use pure Type 5 route within a single data center to interconnect points of delivery (pods) as long as the IP prefix can be confined within the pod.

**BEST PRACTICE:** Note that there are differences between EVPN VXLAN and EVPN MPLS. EVPN VXLAN exports a separate route target for Type 1 routes. EVPN-MPLS exports the Type 1 route with the collective set of route-targets of the VNI or tags (broadcast domains) in which the Ethernet segment identifier is participating.

**NOTE:** You cannot use Contrail with pure Type 5 route.

Release Information

Support for SRX and vSRX Virtual Firewall Series firewalls added in Junos OS Release 22.4R1.

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
17.4R1	Starting with Junos OS Release 17.4R1, pure Type 5 routes are supported on standalone QFX5110 switches only.
15.1X53D60	Starting with Junos OS Release 15.1X53-D60, pure Type 5 routes are also supported on QFX10008 and QFX10016 switches.
15.1X53-D30	Only pure Type 5 routes are supported. Support was added in Junos OS Release 15.1X53-D30 for QFX10002 switches only.

RELATED DOCUMENTATION

Understanding EVPN with VXLAN Data Plane Encapsulation 588
ip-prefix-routes
ip-prefix-support

Configuring Seamless Type-5 to Type-5 Stitching for EVPN-VXLAN

SUMMARY

Seamless stitching of EVPN Type 5 routes is beneficial in multi-POD data center architectures, or Data Center Interconnect (DCI) using pure Type 5 routing. It reduces the number of Type 5 tunnels between border-leaf or border-spine devices.

IN THIS SECTION

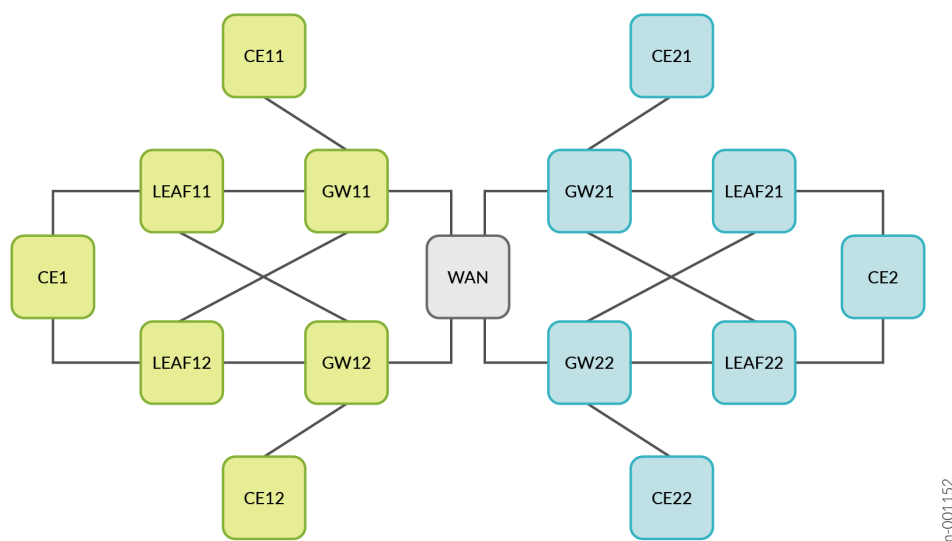
- [Topology | 55](#)
- [Configuration Overview | 56](#)
- [Configure a MAC-VRF Instance | 56](#)
- [Configure a Layer 3 VRF Instance | 57](#)
- [Domain Path ID | 58](#)
- [Release Information | 58](#)

This article contains the configuration necessary for implementing seamless stitching of Type 5 routes between data center PODs. We support a 1:1 VRF instance mapping, meaning that a single VRF instance on the local data center gateway maps to another VRF instance on the remote gateway.

Topology

The following diagram shows two data center PODs connected over a WAN. The two data centers are configured using edge-routed bridging (ERB).

Figure 6: EVPN-VXLAN data center PODs connected over a WAN



Configuration Overview

The following steps show the relevant configuration to implement pure EVPN Type 5 routing between stitched data centers. Each data center POD will have a local EVPN-VXLAN overlay fabric, and utilize EVPN DCI across the WAN to the remote data center POD.

Devices within the same POD will use MAC-VRF instances for the EVPN-VXLAN overlay fabric. Gateway devices will additionally use a Layer 3 VRF instance for the Type 5 DCI across the WAN. Our example focuses on the gateway device overlay configuration. The underlay can use multiple options. See [Understanding EVPN with VXLAN Data Plane Encapsulation](#) for details.

Configure a MAC-VRF Instance

1. Configure the MAC-VRF for GW11. Here we're using the `vlan-aware` service. You might alternately configure `vlan-based` service or `vlan-bundle` service. See [Overview of VLAN Services for EVPN](#) for additional information about these service types.

```
user@GW11#set routing-instances <instance-name> instance-type mac-vrf
```

2. Configure the EVPN protocol with VXLAN encapsulation.

```
user@GW11#set routing-instances <instance-name> protocols evpn encapsulation vxlan
set routing-instances <instance-name> protocols evpn default-gateway do-not-advertise
```

3. Configure a VXLAN tunnel endpoint (VTEP) interface (such as the lo0 interface), the aforementioned service type, one or more VLAN-tagged interfaces, and a router distinguisher and VRF target unique to this MAC-VRF instance.

```
user@GW11#set routing-instances <instance-name> vtep-source-interface interface
set routing-instances <instance-name> service-type vlan-aware
set routing-instances <instance-name> interface <interface>
set routing-instances <instance-name> route-distinguisher <value>
set routing-instances <instance-name> vrf-target target:<value>
```

4. Configure the bridge domain used for the EVPN interconnection. See [Configuring a Bridge Domain](#) for more details. That reference page also includes important information regarding the use of the routing-instances <instance-name> bridge-domains hierarchy, or the routing-instances <instance-name> vlans hierarchy.

```
user@GW11#set routing-instances <instance-name> bridge-domains <name> vlan-id <vlan-id>
set routing-instances <instance-name> bridge-domains <name> interface <interface>
set routing-instances <instance-name> bridge-domains <name> routing-interface <irb interface>
set routing-instances <instance-name> bridge-domains <name> vxlan vni <value>
```

Configure a Layer 3 VRF Instance

1. Configure the Layer 3 VRF instance for GW11, including a route distinguisher and VRF target unique to this VRF instance.

```
user@GW11#set routing-instances <instance-name> instance-type vrf
set routing-instances <instance-name> routing-options multipath
set routing-instances <instance-name> interface interface
set routing-instances <instance-name> route-distinguisher <value>
set routing-instances <instance-name> vrf-target target:<value>
set routing-instances <instance-name> vrf-table-label
```

2. Configure the EVPN interconnect, including a route distinguisher and VRF target unique to the interconnect configuration.

```
user@GW11#set routing-instances <instance-name> protocols evpn interconnect vrf-target
target:<value>
set routing-instances <instance-name> protocols evpn interconnect route-distinguisher <value>
```

3. Configure Type 5 advertisements using the ip-prefix-routes statement. The vni statement maps VLAN's across the interconnect. The VNI here must be unique to the ip-prefix-routes configuration.

```
user@GW11#set routing-instances <instance-name> protocols evpn ip-prefix-routes advertise
direct-nexthop
set routing-instances <instance-name> protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances <instance-name> protocols evpn ip-prefix-routes vni <vni>
```

Domain Path ID

You can optionally configure the domain-path-id statement as part of your DCI Type 5 stitching implementation. This configuration helps to prevent Type 5 routing loops. The domain-path-id statement is configured in the protocols evpn interconnect stanza. See [domain-path-id](#) for more details.

```
set protocols evpn interconnect domain-path-id <domain-id-string>
```

Release Information

Support for Advanced Forwarding Toolkit (AFT)-based linecards on supported MX Series routers introduced in Junos OS Release 25.2R1.

Seamless VXLAN Stitching with Symmetric EVPN Type 2 Routes using Data Center Interconnect

Data Center Interconnect (DCI) enables you to segment the DC fabric into multiple points of delivery (PODs). Seamless stitching of Virtual Extensible LAN (VXLAN) Virtual Network Identifiers (VNIs) allows you to selectively stretch your Layer 2 network between PODs. Each POD will follow a spine and leaf design. The leaf nodes in a given POD will only establish VXLAN tunnels with leaves and spines in their POD. Spines will transport traffic to other spines of different PODs. VXLAN stitching with DCI is ideal

for larger networks since it reduces MAC flooding by reducing the required number of VXLAN tunnels between PODs.

Symmetric integrated routing and bridging (IRB) in an EVPN-VXLAN environment occurs when the ingress and egress VXLAN tunnel end points (VTEPs) perform both routing and bridging on each side of the VXLAN tunnel. The ingress provider edge (PE) device performs a MAC lookup followed by an IP lookup, and the egress PE performs the opposite, an IP lookup followed by a MAC lookup.

RELATED DOCUMENTATION

[Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics](#)
[Configure VXLAN Stitching for Layer 2 Data Center Interconnect](#)

Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics

SUMMARY

This page provides an overview of symmetric integrated routing and bridging (IRB) with EVPN over Virtual Extensible LAN (VXLAN) tunnels. We also introduce the elements you configure to enable symmetric EVPN Type 2 routing.

IN THIS SECTION

- [Overview of Symmetric EVPN Routing with Type 2 Routes | 59](#)
- [Enable Symmetric IRB with EVPN Type 2 Routes | 63](#)

Overview of Symmetric EVPN Routing with Type 2 Routes

IN THIS SECTION

- [Benefits of the Symmetric Model | 60](#)
- [Asymmetric and Symmetric IRB Models | 60](#)
- [Asymmetric Model | 61](#)
- [Symmetric Model | 61](#)
- [EVPN Type 2 Route Enhancements to Support the Symmetric Routing Model | 63](#)

The Internet Engineering Task Force (IETF) open standard document RFC 9135, [Integrated Routing and Bridging in EVPN](#), defines two operational models for inter-subnet forwarding in EVPN:

- An asymmetric model.
- A symmetric model.

By default in EVPN-VXLAN networks, Junos OS devices use the asymmetric IRB model with EVPN Type 2 routes to send traffic between subnets across the VXLAN tunnels. On supporting devices, you can alternatively enable the devices to use a symmetric model with EVPN Type 2 routes for inter-subnet routing. We support symmetric EVPN Type 2 routing in an EVPN-VXLAN fabric with an edge-routed bridging (ERB) overlay.

These models can also apply to EVPN Type 5 (IP prefix) routes. We support EVPN Type 5 routing on Junos OS devices using only the symmetric IRB model. This is the default behavior when you configure a routing instance to use Type 5 routes with the `ip-prefix-routes` statement. See ["Understanding EVPN Pure Type 5 Routes" on page 50](#) for an overview of EVPN Type 5 routes and other EVPN route types. See ["EVPN Type 5 Route with VXLAN Encapsulation for EVPN-VXLAN" on page 47](#) for details on how Type 5 routes work.

Benefits of the Symmetric Model

- Avoids scaling issues inherent in the asymmetric model when your network has a large number of VLANs. On each device, you only need to configure the VLANs that serve the connected hosts on that device. With the asymmetric model, you must configure the device with all destination VLANs in the network.
- Simplifies traffic monitoring by using the same tunnel identifier (VXLAN network identifier [VNI]) for inter-subnet routing in both directions for a particular tenant. The asymmetric routing model requires different VNIs in each direction in that case.

Asymmetric and Symmetric IRB Models

For intra-subnet forwarding in ERB overlay fabrics, leaf devices serving as VXLAN tunnel end points (VTEPs) forward VXLAN traffic the same way in both the asymmetric and symmetric models. The source and destination VLAN and VNI are the same on both sides of the tunnel. The VTEPs bridge the traffic to and from the tunnel.

For inter-subnet routing, both models use IRB interfaces for routing, but the two models differ in configuration and benefits.

The next sections describe more about how the two models work, with focus on the symmetric model. We also cover tradeoffs for using either model.

Asymmetric Model

With the asymmetric model, leaf devices serving as VXLAN tunnel end points (VTEPs) both route and bridge to initiate the VXLAN tunnel (tunnel ingress). However, when exiting the VXLAN tunnel (tunnel egress), the VTEPs can only bridge the traffic to the destination VLAN.

With this model, VXLAN traffic must use the destination VNI in each direction. The source VTEP always routes the traffic to the destination VLAN and sends it using the destination VNI. When the traffic arrives at the destination VTEP, that device forwards the traffic on the destination VLAN.

This model requires you to configure all source and destination VLANs and their corresponding VNIs on each leaf device, even if a leaf doesn't host traffic for some of those VLANs. As a result, this model can have scaling issues when the network has a large number of VLANs. However, when you have fewer VLANs, this model can have lower latency over the symmetric model. Configuration is also simpler than with the symmetric model.

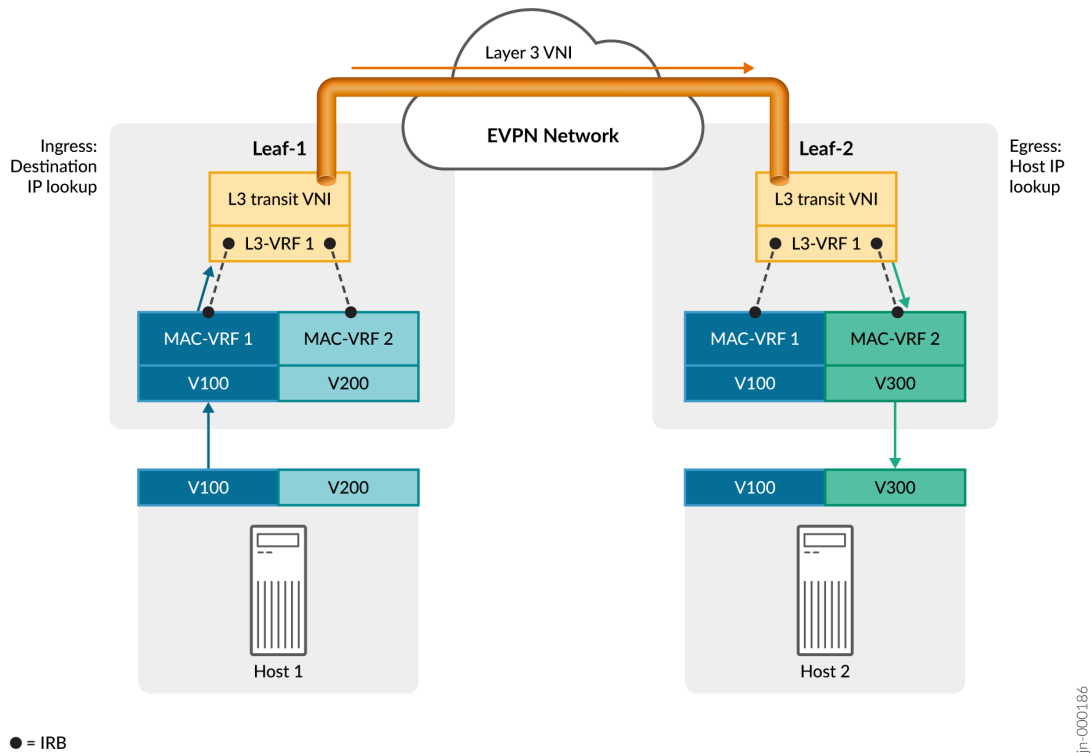
Symmetric Model

With the symmetric IRB routing model, the VTEPs do routing and bridging on both the ingress and egress sides of the VXLAN tunnel. As a result, VTEPs can do inter-subnet routing for the same tenant virtual routing and forwarding (VRF) instance with the same VNI in both directions. We implement this model for EVPN Type 2 routes the same way as for EVPN Type 5 routes (which we support using only the symmetric model). The VTEPs use a dedicated Layer 3 traffic VNI in both directions for each tenant VRF instance.

[Figure 7 on page 62](#) illustrates the symmetric model with switches serving as leaf devices in an ERB overlay configuration. The EVPN instances on the leaf devices use the MAC-VRF instance type at Layer 2. You configure each MAC-VRF instance (with one or more VLANs) with IRB interfaces to route traffic to a associated tenant VRF instance at Layer 3 (L3 VRF).

You configure an extra VLAN with an IRB interface, mapped to a VNI, for each tenant L3 VRF instance. That VNI is the Layer 3 transit VNI between VTEPs for the tenant VXLAN traffic. The tenant L3 VRF instance routes the traffic onto the Layer 3 transit VNI. The symmetric model uses the Layer 3 transit VNI in both directions regardless of the destination VLAN and its corresponding VNI.

Figure 7: Symmetric IRB Model on Leaf Devices in an ERB Overlay Fabric



This model requires that the network has established Layer 3 connectivity between all source and destination VTEPs for EVPN type 2 routing. You configure EVPN Type 5 routing in the tenant VRF instance to provide the Layer 3 connectivity.

Figure 7 on page 62 shows how a leaf device on one VLAN symmetrically routes tenant traffic to another leaf device on a different VLAN, as follows:

1. A tenant host sends traffic on the source VLAN to the remote tenant host in the EVPN-VXLAN network on a different VLAN.
2. The source (ingress) leaf device routes the source VLAN traffic through the tenant L3 VRF onto the VXLAN tunnel. The tunnel VNI is the Layer 3 transit VNI.
3. The Layer 3 network infrastructure tunnels the traffic to the destination VTEP using the Layer 3 transit VNI.
4. The destination (egress) leaf device routes the traffic from the Layer 3 transit VNI onto the destination VLAN.
5. The destination leaf device bridges the traffic on the destination VLAN to the destination host.



NOTE: [Figure 7 on page 62](#) shows MAC-VRF instances with the VLAN-based service type (one instance serves one VLAN). However, we support either VLAN-based or VLAN-aware bundle service types with symmetric Type 2 routing.

EVPN Type 2 Route Enhancements to Support the Symmetric Routing Model

EVPN Type 2 routes are MAC/IP advertisement routes that are described in RFC 7432, [BGP MPLS-Based Ethernet VPN](#). To support the symmetric routing model, we implement the EVPN Router's MAC extended community that is described in RFC 9135, [Integrated Routing and Bridging in EVPN](#). This extended community Type field value is 0x06 (EVPN) with Sub-Type field 0x03, and includes the device's MAC address. For symmetric IRB routing, EVPN leaf devices send this extended community (along with the tunnel type encapsulation extended community) in the EVPN Type 2 route advertisements.

The EVPN Type 2 MAC/IP route advertisement also includes two label fields for:

- The VNI corresponding to the Layer 2 routing instance—the MAC-VRF EVPN instance
- The VNI corresponding to the Layer 3 routing instance—in this case, the Layer 3 transit VNI.

When you enable symmetric IRB routing on an EVPN leaf device, the device checks that received Type 2 route advertisements have the proper fields. The device logs an error and rejects Type 2 routes that don't include the Layer 3 (IP) VNI value, which we require for symmetric IRB routing.

Trade-offs with the Symmetric Model

For inter-subnet routing, the symmetric model enables better scaling over the asymmetric model in configurations with a large number of VLANs. With the symmetric model, you can configure each VTEP with only the VLANs that serve its connected hosts. However, you also need an additional Layer 3 transit VLAN and VNI for each tenant virtual routing and forwarding (VRF) instance.

When your EVPN-VXLAN network has a large number of VLANs, the symmetric model helps to avoid the scaling issues inherent in the asymmetric model. With the asymmetric model, you must configure destination VLANs on a device even if none of its connected hosts use those VLANs. With the symmetric model, you can configure each device only with the VLANs its connected hosts use. However, if the network serves most or all VLANs on all devices in any case, your configuration can be simpler using the asymmetric model.

Enable Symmetric IRB with EVPN Type 2 Routes

Junos OS devices use the asymmetric model with EVPN Type 2 routes by default, or you can enable the symmetric model with EVPN Type 2 routes.

We support EVPN Type 2 symmetric routing as follows:

- In an EVPN-VXLAN fabric with an edge-routed bridging (ERB) overlay.
- With EVPN instances configured using MAC-VRF routing instances with VLAN-based or VLAN-aware bundle service types (see ["MAC-VRF Routing Instance Type Overview" on page 38](#)).



NOTE: QFX5210 switches support symmetric EVPN Type 2 routing, but those switches only support EVPN-VXLAN using a loopback port solution for VXLAN routing in and out of tunnels (RIOT).

See ["Using a RIOT Loopback Port to Route Traffic in an EVPN-VXLAN Network" on page 748](#) for details on how that implementation works, including the configuration steps to enable symmetric EVPN Type 2 routing with EVPN-VXLAN on those switches.

The steps below apply to all other supported platforms.

Here are the high-level steps to enable symmetric EVPN Type 2 routing on leaf devices in an EVPN-VXLAN fabric with an ERB overlay.



NOTE: For an example configuration of an ERB overlay use case with symmetric EVPN Type 2 routing, see Data Center EVPN-VXLAN Fabric Architecture Guide—[Edge-Routed Bridging Overlay Design and Implementation](#), section [Configure Symmetric IRB Routing with EVPN Type 2 Routes on Leaf Devices](#).

1. On the leaf devices that serve as VTEPs, configure the base MAC-VRF EVPN instance(s), tenant L3 VRF instances, associated VLANs, and corresponding IRB interfaces for your EVPN-VXLAN network.
2. Configure EVPN Type 5 routing in the EVPN-VXLAN network to establish Layer 3 connectivity between all leaf devices serving as VTEPs.

In this step, you configure the L3 VRF instances to use EVPN Type 5 routes with a corresponding VNI to advertise the direct next hops for VXLAN traffic. For example:

```
set routing-instances l3-vrf-name protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances l3-vrf-name protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances l3-vrf-name protocols evpn ip-prefix-routes vni l3-vni
```

The device uses the L3 VRF EVPN Type 5 VNI as the L3 transit VNI for symmetric EVPN Type 2 routing.



NOTE: ACX7100 routers don't support asymmetric VNI values on either side of a VXLAN tunnel for a given VRF. To support EVPN Type 5 routing and symmetric IRB routing with EVPN Type 2 routes on ACX7100 routers, you must configure the same L3 VNI value for a particular VRF on each of the leaf devices. On other platforms, the L3 VNI can be different on either side of the tunnel for a given VRF.

3. Enable symmetric Type 2 routing in the L3 VRF instances using the `irb-symmetric-routing vni vni` statement at the `[edit routing instances name protocols evpn]` hierarchy.

For example:

```
set routing-instances l3-vrf-name protocols evpn irb-symmetric-routing vni l3-vni
```

Specify the transit VNI value (*l3-vni*) as the same VNI you configure in step 2 when you enable EVPN Type 5 routing for each L3 VRF.

SEE ALSO

[*irb-symmetric-routing*](#)

[Understanding VXLANs | 606](#)

[Using a RIOT Loopback Port to Route Traffic in an EVPN-VXLAN Network | 748](#)

EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN

SUMMARY

Learn how a device in an EVPN-VXLAN fabric gives preference to either an EVPN Type 2 route or an EVPN Type 5 route when the device learns and advertises both types of routes.

IN THIS SECTION

- [Benefits | 66](#)
- [EVPN Type 2 and Type 5 Coexistence Preference Algorithm | 66](#)
- [Type 5 Route Options and Policy Options | 67](#)
- [CLI Commands to Verify Preferred Route Type | 68](#)

A device in an EVPN-VXLAN edge-routed bridging fabric imports and advertises EVPN Type 2 MAC+IP routes by default. You can also configure the device to import and to advertise EVPN Type 5 IP prefix routes. The device treats either type of route as a unique route, even when it corresponds to the same destination host. Each unique IP host route uses a dedicated next hop in the Packet Forwarding Engine (PFE). As a result, having both types of routes enabled together puts a strain on the PFE's next-hop resources. Also, traffic generally flows more efficiently if the device uses one type of route over the other in certain cases.

Starting in Junos OS Release 21.2R1, when the device has both types of routes together in a routing instance for the same destination, it uses a preference algorithm to choose the preferred route to store.

Benefits

- Supports higher scaling in edge-routed bridging fabrics because the preference algorithm reduces next-hop resource requirements in the PFE.
- Improves bridging performance by choosing the more efficient path for locally learned (Type 2) host IP routes in an Ethernet segment.
- Supports higher scaling with EVPN Type 5 routes while enabling ARP suppression and proxy ARP in the fabric using Type 2 routes. The Type 2 routes carry the customer edge (CE) device MAC address information required for proxy ARP to work.

EVPN Type 2 and Type 5 Coexistence Preference Algorithm

In an EVPN-VXLAN fabric, devices use Type 2 routes by default. You must explicitly enable the device to also import and advertise EVPN Type 5 routes in a virtual routing and forwarding (VRF) instance. To enable Type 5 routes and the coexistence preference algorithm, configure the `ip-prefix-routes` statement at the `[edit routing-instances name protocols evpn]` hierarchy level.

With Type 5 routes enabled, the device might learn an IP host address from a Type 2 MAC + IP route for which it also has a Type 5 route for the same prefix. In that case, the device uses the coexistence preference algorithm to choose only one of the two routes to store as the preferred route.

The preference algorithm works as follows:

- For any destinations for which the device has no Type 5 route, the device uses Type 2 routes.
- If the device has a Type 5 route with a matching prefix for a local ESI Type 2 route, it installs the Type 2 route.
- Otherwise the device prefers the Type 5 route for all other destinations.

With this algorithm, EVPN-VXLAN devices generally prefer Type 5 routes for traffic in or between data centers and in or between VLANs (bridge domains). Type 5 routes are ideal for the purpose of VXLAN

overlay routing. However, a device can forward packets more efficiently using Type 2 routes in some cases. One example is when the device learns Type 2 MAC+IP routes for destinations that it only needs to bridge locally. The preference algorithm accounts for that case.

The device does not use make before break (MBB) actions to adjust preferred routes due to configuration updates. When making a route preference change in this case, the device first deletes the non-preferred route, then adds the new preferred route. Examples of configuration changes that can cause preference changes include:

- If you didn't enable Type 5 routes, and then you enable Type 5 routes.
- When you configure an ESI locally, in which case the device prefers the MAC+IP Type 2 route.

Type 5 Route Options and Policy Options

In your EVPN-VXLAN configuration, use one of these advertising mode options with the `ip-prefix-routes` statement when you enable Type 5 routes:

- `advertise direct-nexthop`—With this option, the device advertises all non-host routes for a VRF as Type 5 advertisements.
- `advertise gateway-address`—With this option, the device only advertises prefixes for the IRB interfaces for the extended EVPN instances and bridge domains.

You can also include policy options in your EVPN-VXLAN Type 5 route configuration to refine the routes the device actually imports or advertises.

For example, you can define:

- A policy to export all local routes and EVPN routes, such as the following:

```
set policy-options policy-statement type5_export_policy term 1 from protocol local
set policy-options policy-statement type5_export_policy term 1 then accept
set policy-options policy-statement type5_export_policy term 2 from protocol evpn
set policy-options policy-statement type5_export_policy term 2 from route-filter 0.0.0.0/0
prefix-length-range /32-/32
set policy-options policy-statement type5_export_policy term 2 then accept
set policy-options policy-statement type5_export_policy term 3 from protocol evpn
set policy-options policy-statement type5_export_policy term 3 from route-filter
00:00:00:00:00:00:00:00/0 prefix-length-range /128-/128
set policy-options policy-statement type5_export_policy term 3 then accept
```

- A policy that only advertises routes for specific host addresses or prefixes, such as the following:

```
set policy-options policy-statement type5_policy term 1 from protocol local
set policy-options policy-statement type5_policy term 1 from route-filter 10.0.2.0/24 orlonger
set policy-options policy-statement type5_policy term 1 then accept
set policy-options policy-statement type5_policy term 2 from protocol local
set policy-options policy-statement type5_policy term 2 from route-filter
2001::192:101:1:100/48 orlonger
set policy-options policy-statement type5_policy term 2 then accept
```

- A policy that filters and doesn't import routes for specific host addresses or prefixes, such as the following:

```
set policy-options policy-statement type5_import_policy term 1 from protocol bgp
set policy-options policy-statement type5_import_policy term 1 from route-filter 10.0.2.0/24
orlonger
set policy-options policy-statement type5_import_policy term 1 then reject
set policy-options policy-statement type5_import_policy term 2 from protocol bgp
set policy-options policy-statement type5_import_policy term 2 from route-filter
2001::192:101:1:100/48 orlonger
set policy-options policy-statement type5_import_policy term 2 then reject
```

Include the desired policy options in your ip-prefix-routes configuration. For example:

```
set routing-instances vrf1 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances vrf1 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances vrf1 protocols evpn ip-prefix-routes vni 100
set routing-instances vrf1 protocols evpn ip-prefix-routes import type5_import_policy
set routing-instances vrf1 protocols evpn ip-prefix-routes export type5_export_policy
set routing-instances vrf1 instance-type vrf
set routing-instances vrf1 interface irb.1
set routing-instances vrf1 interface irb.2
set routing-instances vrf1 route-distinguisher 192.168.0.1:100
set routing-instances vrf1 vrf-target target:100:200
```

CLI Commands to Verify Preferred Route Type

You can use the CLI commands in this section to see the preferred route type.

show ethernet-switching mac-ip-table

The *show ethernet-switching mac-ip-table* CLI command displays the value RTS in the MAC IP flags column when the switch "skips" adding a learned Type 2 route. This value means the device prefers a Type 5 route with a matching prefix. The command displays the definition *Dest Route Skipped* for the value RTS at the top of the output.

The command displays results for all VLANs (bridge domains) by default. You can also specify a particular VLAN (bridge domain). By default, the command displays information for the default-switch instance. You can also use other command line options to display results for a specific routing instance if the device has multiple routing instances.

The following sample output shows MAC+IP table information for the default-switch instance where the device preferred Type 5 routes for some destinations:

```
user@device> show ethernet-switching mac-ip-table
```

MAC IP flags (S - Static, D - Dynamic, L - Local , R - Remote, Lp - Local Proxy,
Rp - Remote Proxy, K - Kernel, RT - Dest Route, (N)AD - (Not) Advt to remote,
RE - Re-ARP/ND, RO - Router, OV - Override, Ur - Unresolved,
RTS - Dest Route Skipped, RGW - Remote Gateway)

Routing instance : default-switch

Bridging domain : v1

IP address	MAC address	Flags	Logical Interface	Active source
10.1.1.254	10:10:10:00:00:01	S,K	irb.1	
2001:db8::a:1:1:fffe	20:20:20:00:00:01	S,K, RTS	irb.1	
10.1.1.2	c8:e7:f0:4b:d1:00	DR,K, RTS	vtep.32769	
192.168.22.22				
2001:db8::a:1:1:2	c8:e7:f0:4b:d1:00	DR,K, RTS	vtep.32769	
192.168.22.22				
fe80::cae7:f000:14b:d100	c8:e7:f0:4b:d1:00	DR,K,RT	vtep.32769	
192.168.22.22				
10.1.1.1	dc:38:e1:e0:30:c0	S,K	irb.1	
2001:db8::a:1:1:1	dc:38:e1:e0:30:c0	S,K	irb.1	
fe80::de38:e100:1e0:30c0	dc:38:e1:e0:30:c0	S,K	irb.1	

MAC IP flags (S - Static, D - Dynamic, L - Local , R - Remote, Lp - Local Proxy,
Rp - Remote Proxy, K - Kernel, RT - Dest Route, (N)AD - (Not) Advt to remote,
RE - Re-ARP/ND, RO - Router, OV - Override, Ur - Unresolved,
RTS - Dest Route Skipped, RGW - Remote Gateway)

Routing instance : default-switch

Bridging domain : v2

IP address	MAC address	Flags	Logical Interface	Active source
10.1.2.254	10:10:10:00:00:02	S,K,RTS	irb.2	
2001:db8::a:1:2:fffe	20:20:20:00:00:02	S,K,RTS	irb.2	
10.1.2.2	c8:e7:f0:4b:d1:00	DR,K,RTS	vtep.32769	
192.168.22.22				
2001:db8::a:1:2:2	c8:e7:f0:4b:d1:00	DR,K,RTS	vtep.32769	
192.168.22.22				
fe80::cae7:f000:24b:d100	c8:e7:f0:4b:d1:00	DR,K,RT	vtep.32769	
192.168.22.22				
10.1.2.1	dc:38:e1:e0:30:c0	S,K	irb.2	
2001:db8::a:1:2:1	dc:38:e1:e0:30:c0	S,K	irb.2	
fe80::de38:e100:2e0:30c0	dc:38:e1:e0:30:c0	S,K	irb.2	

With the extensive option, the show ethernet-switching mac-ip-table command includes the value dest_route_skipped in the MAC IP flags row. The following command shows the MAC+IP table information for MAC address c8:e7:f0:4b:d1:00:

```
user@device> show ethernet-switching mac-ip-table extensive c8:e7:f0:4b:d1:00
```

```
IP address: 10.1.1.2
```

```
MAC address: c8:e7:f0:4b:d1:00
```

```
Routing instance: default-switch
```

```
Bridging domain: v1
```

```
Logical interface: vtep.32769
```

IP address	MAC address	Flags	Logical Interface	Active source
---------------	----------------	-------	----------------------	------------------

```
192.168.22.22
```

```
MAC-IP local mask: 0x0000000000000000
```

```
MAC-IP remote mask: 0x0000000000000000
```

```
MAC-IP remote-proxy mask: 0x0000000000000000
```

```
MAC-IP RPD flags: 0x08000081
```

```
MAC-IP flags: remote, kernel, dest_route_skipped
```

```
IP address: 2001:db8::a:1:1:2
```

```
MAC address: c8:e7:f0:4b:d1:00
```

```
Routing instance: default-switch
```

```
Bridging domain: v1
```

```
Logical interface: vtep.32769
```

IP address	MAC address	Flags	Logical Interface	Active source
---------------	----------------	-------	----------------------	------------------

192.168.22.22
MAC-IP local mask: 0x0000000000000000
MAC-IP remote mask: 0x0000000000000000
MAC-IP remote-proxy mask: 0x0000000000000000
MAC-IP RPD flags: 0x08000101
MAC-IP flags: remote, kernel, **dest_route_skipped**

IP address: fe80::cae7:f000:14b:d100
MAC address: c8:e7:f0:4b:d1:00
Routing instance: default-switch
Bridging domain: v1
Logical interface: vtep.32769

IP address	MAC address	Flags	Logical Interface	Active source
------------	-------------	-------	-------------------	---------------

192.168.22.22
MAC-IP local mask: 0x0000000000000000
MAC-IP remote mask: 0x0000000000000000
MAC-IP remote-proxy mask: 0x0000000000000000
MAC-IP RPD flags: 0x08000101
MAC-IP flags: remote, kernel, dest_route

IP address: 10.1.2.2
MAC address: c8:e7:f0:4b:d1:00
Routing instance: default-switch
Bridging domain: v2
Logical interface: vtep.32769

IP address	MAC address	Flags	Logical Interface	Active source
------------	-------------	-------	-------------------	---------------

192.168.22.22
MAC-IP local mask: 0x0000000000000000
MAC-IP remote mask: 0x0000000000000000
MAC-IP remote-proxy mask: 0x0000000000000000
MAC-IP RPD flags: 0x08000081
MAC-IP flags: remote, kernel, **dest_route_skipped**

IP address: 2001:db8::a:1:2:2
MAC address: c8:e7:f0:4b:d1:00
Routing instance: default-switch
Bridging domain: v2
Logical interface: vtep.32769

IP address	MAC address	Flags	Logical Interface	Active source
192.168.22.22				
MAC-IP local mask: 0x0000000000000000				
MAC-IP remote mask: 0x0000000000000000				
MAC-IP remote-proxy mask: 0x0000000000000000				
MAC-IP RPD flags: 0x0800101				
MAC-IP flags: remote, kernel, dest_route_skipped				
IP address: fe80::cae7:f000:24b:d100				
MAC address: c8:e7:f0:4b:d1:00				
Routing instance: default-switch				
Bridging domain: v2				
Logical interface: vtep.32769				
IP address	MAC address	Flags	Logical Interface	Active source
192.168.22.22				
MAC-IP local mask: 0x0000000000000000				
MAC-IP remote mask: 0x0000000000000000				
MAC-IP remote-proxy mask: 0x0000000000000000				
MAC-IP RPD flags: 0x0800101				
MAC-IP flags: remote, kernel, dest_route				

show route forwarding-table

The *show route forwarding-table* CLI command shows when the device uses a local Type 5 route over a learned Type 2 route for a destination. In this case, when you include the *extensive* option, the output includes the keywords *VxLAN Local* in the *Flags* field. This flag means the route is a Type 5 route.

For example:

```
user@device> show route forwarding-table destination 10.1.1.2 table vrf1 extensive
Routing table: vrf1.inet [Index 8]
Internet:

Destination: 10.1.1.2/32
Route type: user
Route reference: 0                               Route interface-index: 0
Multicast RPF nh index: 0
```

```

P2mpidx: 0
Flags: sent to PFE, prefix load balance , VxLAN Local
Nexthop:
Next-hop type: composite          Index: 1799      Reference: 3
Next-hop type: indirect          Index: 524291   Reference: 2
Nexthop: 10.0.0.2
Next-hop type: unicast           Index: 1796      Reference: 3
Next-hop interface: xe-0/0/11.0

```

show route table

The *show route table* CLI command with the extensive option includes the keyword `VxlanLocalRT` in the State output field for a Type 5 route.

Here's an example using the syntax *show route destination-prefix table table-name extensive*. This command shows that the device installed a Type 5 route for the destination IP prefix 10.1.1.2 in the routing table for routing instance vrf1:

```

user@device> show route 10.1.1.2 table vrf1.inet.0 extensive

vrf1.inet.0: 14 destinations, 18 routes (14 active, 0 holddown, 0 hidden)
10.1.1.2/32 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.1.1.2/32 -> {composite(1799)}
    *EVPN   Preference: 170/-101
            Next hop type: Indirect, Next hop index: 0
            Address: 0x6d06abc
            Next-hop reference count: 6
            Next hop type: Router, Next hop index: 1796
            Next hop: 10.0.0.2 via xe-0/0/11.0, selected
            Label element ptr: 0x83f82d8
            Label parent element ptr: 0x0
            Label element references: 1
            Label element child references: 0
            Label element lsp id: 0
            Session Id: 0x0
            Protocol next hop: 192.168.22.22
            Composite next hop: 0x66fc1b0 1799 INH Session ID: 0x0
            VXLAN tunnel rewrite:
                MTU: 0, Flags: 0x0
                Encap table ID: 0, Decap table ID: 8
                Encap VNI: 100, Decap VNI: 100

```

```

Source VTEP: 192.168.11.11, Destination VTEP: 192.168.22.22
SMAC: dc:38:e1:e0:30:c0, DMAC: c8:e7:f0:4b:d1:00
Indirect next hop: 0x6df0e84 524291 INH Session ID: 0x0
State: <Active Int Ext VxlanLocalRT>
Age: 5:10:00 Metric2: 1
Validation State: unverified
Task: vrf1-EVPN-L3-context
Announcement bits (1): 2-KRT
AS path: I
Communities: target:100:200 encapsulation:vxlan(0x8) router-mac:c8:e7:f0:4b:d1:00
Thread: junos-main
Composite next hops: 1
  Protocol next hop: 192.168.22.22 Metric: 1
  Composite next hop: 0x66fc1b0 1799 INH Session ID: 0x0
  VXLAN tunnel rewrite:
    MTU: 0, Flags: 0x0
    Encap table ID: 0, Decap table ID: 8
    Encap VNI: 100, Decap VNI: 100
    Source VTEP: 192.168.11.11, Destination VTEP: 192.168.22.22
    SMAC: dc:38:e1:e0:30:c0, DMAC: c8:e7:f0:4b:d1:00
    Indirect next hop: 0x6df0e84 524291 INH Session ID: 0x0
  Indirect path forwarding next hops: 1
    Next hop type: Router
    Next hop: 10.0.0.2 via xe-0/0/11.0
    Session Id: 0x0
    192.168.22.22/32 Originating RIB: inet.3
    Metric: 1 Node path count: 1
    Forwarding nexthops: 1
      Next hop type: Router
      Next hop: 10.0.0.2 via xe-0/0/11.0
      Session Id: 0x0

```

RELATED DOCUMENTATION

ip-prefix-routes

Ingress Virtual Machine Traffic Optimization

IN THIS SECTION

- [Benefits of Ingress Virtual Machine Traffic Optimization | 79](#)

When virtual machines and hosts are moved within a data center or from one data center to another, network traffic can become inefficient when the traffic is not routed to the optimal gateway. This can happen when the host is relocated. The arp table does not always get flushed and data flow to the host is sent to the configured gateway even when there is a more optimal gateway. The traffic is “tromboned” and routed unnecessarily to the configured gateway.

Starting in Junos OS Release 18.4R1, Junos supports ingress Virtual machine traffic optimization (VMTO). When ingress VMTO feature is enabled, the remote IP host routes are stored in L3 Virtual Routing and Forwarding (VRF) table and the device routes inbound traffic directly back to host that has been relocated.

[Figure 8 on page 76](#) illustrates tromboned traffic without ingress VMTO and optimized traffic with ingress VMTO enabled. Without ingress VMTO, Spine 1 and 2 from DC1 and DC2 all advertise the remote IP host route 10.0.0.1 when the origin route is from DC2. The ingress traffic can be directed to either Spine 1 and 2 in DC1. It would then be routed to Spine 1 and 2 in DC2 where route 10.0.0.1 has been moved. This causes the tromboning effect. With ingress VMTO, we can achieve optimal forwarding path by configuring a policy for IP host route (10.0.0.1) to only be advertised by Spine 1 and 2 from DC2, and not from DC1 when the IP host is moved to DC2.

Figure 8: Traffic with and without Ingress VMTO

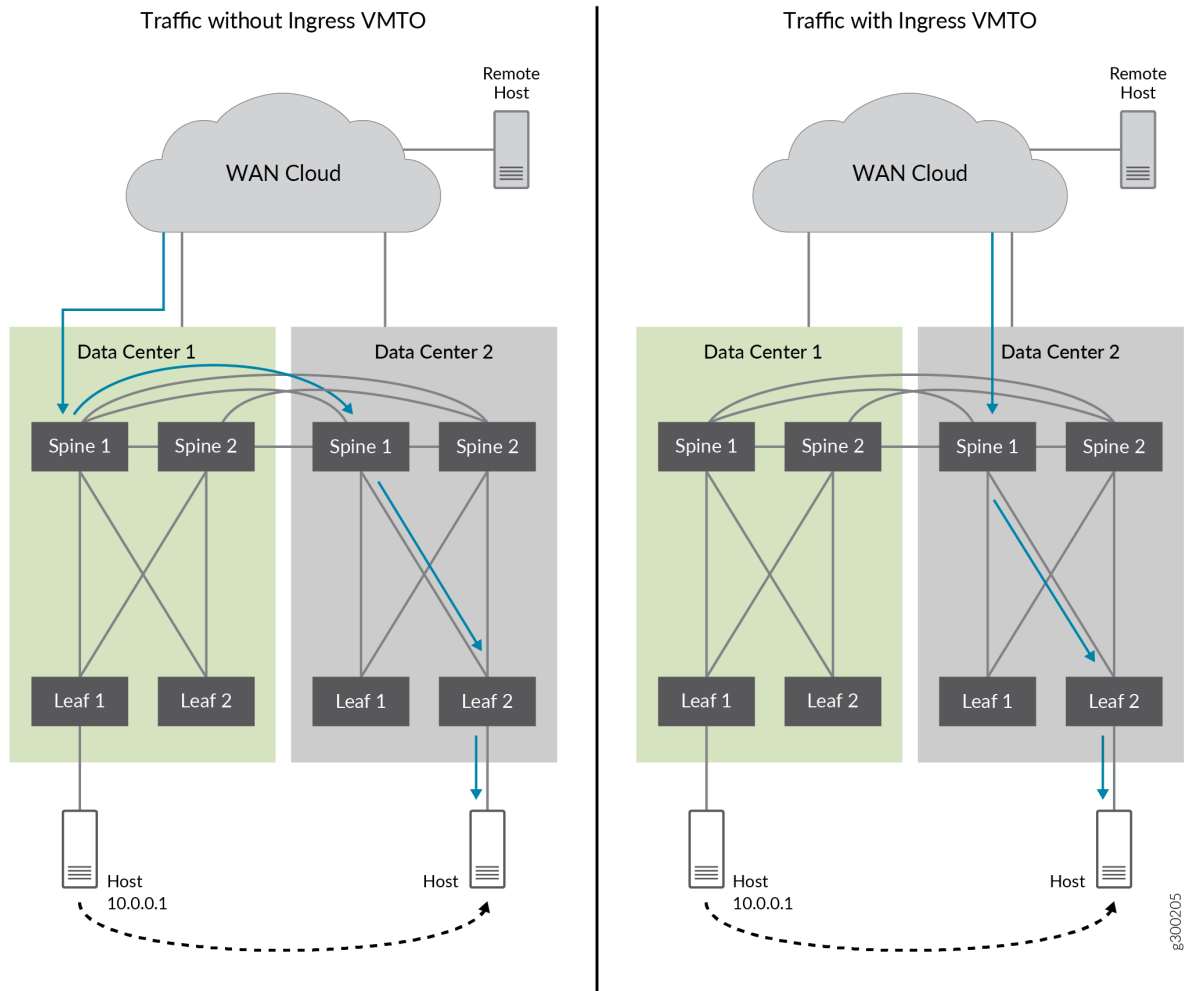
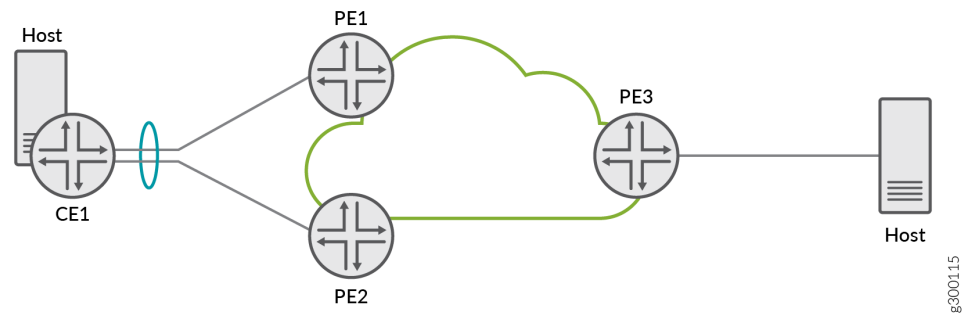


Figure 9 on page 77 illustrates an EVPN network with different elements. PE1 shares an Ethernet Segment with PE2. PE3 is on a separate segment. When PE1 learns of new IP host routes from CE1, PE1 adds the route into the VRF table since it is a locally learned route. If PE2 learns the route from remote peer PE1 (and not from CE1), PE2 will also add the route into VRF table as if the route is also locally learned since PE1 and PE2 are in the same Ethernet Segment. Table 6 on page 77 summarizes the activity taken in the VRF table when a PE device learns of new IP host routes from remote PE devices under EVPN-VXLAN and there are no routing policies configured on the device. Table 7 on page 78 summarizes the activity taken in the VRF table when a PE device learns of new IP host routes from remote PE devices under EVPN-MPLS and there are no routing policies configured on the device.



NOTE: You can also configure policies to selectively add the routes that you want to the VRF table.

Figure 9: EVPN with multihomed devices



NOTE: The IP host routes that PE1 and PE2 learned from each other are described as “From remote device that is connected to a shared Ethernet Segment” and the IP host routes that PE3 learned from PE1 or PE2 are described as “From a remote device that is not connected to a shared Ethernet Segment.”

Table 6: Activity in the VRF table for EVPN-VXLAN

Ingress VMTO Configuration Status	From a remote device that is connected to a shared Ethernet Segment	From a remote device that is not connected to a shared Ethernet Segment
Prior to Junos OS Release 18.4R1.	The IP host route is created with the IRB interface as the next hop and the route is added to the VRF instance table.	The IP host route is not added to the VRF instance table.
Ingress VMTO not configured	The IP host route is created with the IRB interface as the next hop and the route is added to the VRF instance table.	The IP host route is not added to the VRF instance table.
Ingress VMTO configured	The IP host route is created with the IRB interface as the next hop and the route is added to the VRF instance table.	The IP host route is created with the IRB interface as the next hop and the route is added to the VRF instance table.

Table 7: Activity in the VRF table for EVPN-MPLS

Ingress VMTO Configuration Status	From a remote device with a locally shared Ethernet Segment	From a remote device without a locally shared Ethernet Segment
Prior to Junos OS Release 18.4 R1 with chained composite next hop configured.	The IP host route is created with the composite next hop. The route is not advertise to its peer.	The IP host route is created with the composite next hop. The route is not advertise to its peer.
Prior to Junos OS Release 18.4R1 without chained composite next hop.	The IP host route is not added to the VRF instance table.	The IP host route is not added to the VRF instance table.
Ingress VMTO not configured	The IP host route is created with the IRB interface as the next hop and the route is added to the VRF instance table.	The IP host route is not added to the VRF instance table.
Ingress VMTO configured	The IP host route is created with the IRB interface as the next hop and the route is added to the VRF instance table.	The IP host route is created with the IRB interface as the next hop and the route is added to the VRF instance table.
Ingress VMTO configured with composite next hop	The IP host route is created with the IRB interface as the next hop and the route is added to the VRF instance table.	The IP host route is created with the composite next hop and the route is added to the VRF instance table.

To enable ingress VMTO, configure `remote-ip-host-routes` in the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy level. When you enable `remote-ip-host` route, all remote IP host routes will be added to the VRF table. You can specify which remote IP host routes will be added to the VRF table through policies by including an import policy to under `remote-ip-host` routes to filter out the unwanted routes.



NOTE: Juniper recommends that you only enable ingress VMTO on the spine devices in a centrally-routed bridging (CRB) overlay in an EVPN network. This allows devices to advertise learned routes to other routing protocols and to advertise EVPN type 5 routes across different data centers.

To address tomboning in [Figure 8 on page 76](#), you would define the communities for Data Center 1 and Data Center 2 and configure a policy in the spine devices to only import the routes learned from members in its own community. Before the move, the spine devices in Data Center 1 advertise the IP

host route for the local host. After the move, the host becomes part of Data Center 2's community so the spine devices in Data Center 2 will advertise the IP host route. And the next-hop table on the remote host will have the updated route to Data Center 2 after the move.

The following output shows the sample policy and sample configuration with an import policy configured with `remote-ip-host`.

```
user@router1# show policy-options
policy-statement vmto-DC1-import {
  term in-DC1 {
    from community [DC1_devices];
    then accept;
  }
  term not-in-DC1 {
    then reject;
  }
}
```

```
user@router1# show routing-instances
blue {
  instance-type virtual-switch;
  route-distinguisher 10.255.0.3:100;
  vrf-import vmto-DC1-import;
  vrf-target target:100:100;
  protocols {
    evpn {
      remote-ip-host-routes {
        import vmto-DC1-import;
        mpls-use-cnh;
      }
      extended-vlan-list 100;
      default-gateway do-not-advertise;
    }
  }
}
```

Benefits of Ingress Virtual Machine Traffic Optimization

Ingress VMTO provides greater network efficiency and optimizes ingress traffic and can eliminate the trombone effect between VLANs.

Tracing EVPN Traffic and Operations

To configure the EVPN routing instance to trace a variety of different parameters related to EVPN operation:

1. Specify the name of one or more EVPN trace files using the `file` option for the `traceoptions` statement at the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy level:

```
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
}
```

The `file` option includes the following sub-options:

- *filename*—Specify the name of the file to receive the output of the tracing operation. Enclose the name within quotation marks. All files are placed in the directory `/var/log`.
- *files number*—(Optional) Maximum number of trace files. When a trace file named **trace-file** reaches its maximum *size*, it is renamed **trace-file.0**, then **trace-file.1**, and so on, until the specified maximum *number* of trace files specified is reached. Then the oldest trace file is overwritten.
- *size size*—(Optional) Maximum size of each trace file. When a trace file named `trace-file` reaches its maximum size, it is renamed `trace-file.0`, then `trace-file.1`, and so on, until the maximum number of trace files is reached. Then the oldest trace file is overwritten.
- `world-readable | no-world-readable`—(Optional) Enable unrestricted file access or restrict file access to the user who created the file.

2. Specify the `flag` option for the `traceoptions` statement:

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

The `flag` option allows you to specify the scope of the trace by including one of the following sub-options:

- `all`—All EVPN tracing options
- `error`—Error conditions
- `gbptagdb`—GBP tag database related messages
- `general`—General events

- `mac-database`—MAC route database in the EVPN routing instance
- `nlri`—EVPN advertisements received or sent by means of the BGP
- `normal`—Normal events
- `oam`—OAM messages
- `policy`—Policy processing
- `route`—Routing information
- `state`—State transitions
- `task`—Routing protocol task processing
- `timer`—Routing protocol timer processing
- `topology`—EVPN topology changes caused by reconfiguration or advertisements received from other PE routers using BGP

You can also specify one of the following modifiers for any of the traceoptions flags:

- `detail`—Provide detailed trace information.
- `disable`—Disable this trace flag.
- `receive`—Trace received packets.
- `send`—Trace sent packets.

RELATED DOCUMENTATION

[Configuring EVPN Routing Instances | 31](#)

traceoptions

Migrating From BGP VPLS to EVPN Overview

IN THIS SECTION

● [Technology Overview and Benefits | 82](#)

- [BGP VPLS to EVPN Migration | 83](#)
- [EVPN Migration Configuration | 84](#)
- [Reverting to VPLS | 87](#)
- [BGP VPLS to EVPN Migration and Other Features | 87](#)

For service providers using both BGP VPLS and EVPN networks, there is a need to interconnect these networks. Prior to Junos OS Release 18.1, a logical tunnel interfaces on the interconnection point of the VPLS and EVPN routing instances was used for this purpose. In this case, the PE devices in each network are unaware of the PE devices in the other technology network. Starting in Junos OS Release 18.1, a solution is introduced for enabling staged migration from BGP VPLS toward EVPN on a site-by-site basis for every VPN routing instance. In this solution, the PE devices running EVPN and VPLS for the same VPN routing instance and single-homed segments can coexist. The solution supports single-active redundancy of multi-homed networks and multi-homed devices for EVPN PEs. With single-active redundancy, the participant VPN instances may span across both EVPN PEs and VPLS PEs as long as single-active redundancy is employed by EVPN PEs.



NOTE: For migration from BGP VPLS, you should expect some traffic loss when family EVPN is enabled to carry EVPN NLRIs. Routing-instance migration to EVPN should see minimal loss.

The following sections describe the migration from BGP VPLS to EVPN:

Technology Overview and Benefits

VPLS is an Ethernet-based point-to-multipoint Layer 2 VPN. This technology allows you to connect geographically dispersed data center LANs to each other across an MPLS backbone while maintaining Layer 2 connectivity. The high availability features defined in VPLS standards (such as LER dual homing) and topology autodiscovery features using BGP signaling make VPLS scalable and easy to deploy. Because VPLS uses MPLS as its core, it provides low latency variation and statistically bound low convergence times within the MPLS network.

EVPN, on the other hand, is a combined Layer 2 and Layer 3 VPN solution that is more scalable, resilient, and efficient than current technologies. It provides several benefits including greater network efficiency, reliability, scalability, virtual machine (VM) mobility, and policy control for service providers and enterprises.

Although VPLS is a widely deployed Layer 2 VPN technology, service provider networks migrate to EVPN on account of the scaling benefits and ease of deployment. Some of the benefits of EVPN include:

- Control plane traffic is distributed with BGP and the broadcast and multicast traffic is sent using a shared multicast tree or with ingress replication.
- Control plane learning is used for MAC and IP addresses instead of data plane learning. MAC address learning requires the flooding of unknown unicast and ARP frames; whereas, IP address learning does not require any flooding.
- Route reflector is used to reduce a full mesh of BGP sessions among PE devices to a single BGP session between a PE and the route reflector.
- Autodiscovery with BGP is used to discover PE devices participating in a given VPN, PE devices participating in a given redundancy group, tunnel encapsulation types, multicast tunnel type, multicast members, etc.
- All-Active multihoming is used. This allows a given CE device to have multiple links to multiple PE devices, and traffic traversing to-and-from that CE fully utilizes all of these links (Ethernet segment).
- When a link between a CE device and a PE device fails, the PE devices for that EVPN instance (EVI) are notified of the failure with the withdrawal of a single EVPN route. This allows those PE devices to remove the withdrawing PE device as a next hop for every MAC address associated with the failed link (mass withdrawal).

BGP VPLS to EVPN Migration

Some service providers want to preserve their investments in VPLS. This leads to the need to connect the old VPLS networks to new networks that run EVPN. For this purpose, logical tunnel interfaces on the interconnection point of the VPLS and EVPN routing-instances was used. However, all the other PE devices either belonged to the VPLS network or the EVPN network and were unaware of the other technology.

Starting in Junos OS Release 18.1, EVPN can be introduced into an existing BGP VPLS network in a staged manner, with minimal impact to VPLS services. On a BGP VPLS PE device, some customers could be moved to EVPN, while other customers continue to use VPLS pseudowires. Other PE devices could be entirely VPLS and switching customers on other PEs to EVPN.

The seamless migration from BGP VPLS to EVPN solution supports the following functionality:

- Allow for staged migration toward EVPN on a site-by-site basis per VPN instance. For instance, new EVPN sites to be provisioned on EVPN PE devices.
- Allow for the coexistence of PE devices running both EVPN and VPLS for the same VPN instance and single-homed segments.

In the BGP VPLS to EVPN migration, the PE device where some customers have been migrated to EVPN while other customers are being served using VPLS, is called a super PE. As super PE devices discover other super PE devices within a routing instance, they use the EVPN forwarding to communicate with

other super PE devices and VPLS pseudowires to PE devices running VPLS. The PE device with no EVPN awareness, and running only VPLS for all the customers, is called a VPLS PE.

The CE device connected to a super PE can reach both the CE devices connected to EVPN-only PE devices and the CE devices connected to the VPLS-only PE device. The CE devices connected to EVPN-only PE devices cannot reach the CE devices that are connected to VPLS-only PE devices.

Because the migration from BGP VPLS to EVPN is supported at a per routing instance basis, and if the routing instance is serving multiple customers on a PE device, all are migrated together. EVPN is responsible for setting up data forwarding between the PE devices upgraded to EVPN, while VPLS continues to setup data forwarding to PE devices that run VPLS.



NOTE: The following features are not supported with the BGP VPLS to EVPN migration:

- Migration from FEC129 VPLS to EVPN.
- Migration of VPLS virtual switch to EVPN virtual switch.
- Migration of VPLS routing instance to EVPN virtual switch.
- Migration of VPLS routing instance or PBB-VPLS to PBB-EVPN.
- Seamless migration from EVPN back to VPLS.
- Enhancing EVPN to support the set of tools or statements and commands that VPLS supports.
- Spanning all-active across EVPN and VPLS PE devices does not work, as all-active multihoming feature is not supported on VPLS.
- Connecting EVPN-only PE devices with VPLS-only PE devices through super PE devices.
- IPv6, logical systems, multi-chassis support, and SNMP, as they are currently not supported on EVPN.

EVPN Migration Configuration

To perform the BGP VPLS to EVPN migration, do the following:

1. On the backup Routing Engine, load Junos OS Release 18.1R1.
2. Perform ISSU to acquire primary role. Ensure that the VPLS ISSU does not have any impact on the VPLS forwarding.
3. Identify routing instances (customers) that need to be migrated to EVPN.

4. Enable family EVPN signaling in BGP by adding `family evpn` and `signaling` at the `[edit protocols bgp group-session]` hierarchy level.



NOTE: Adding `family evpn` to the BGP protocol will cause the CE IFL to be deleted and recreated, so you should expect some traffic loss after this step.

```
protocols {
  bgp {
    group session {
      family evpn {
        signaling;
      }
    }
  }
}
```

5. Configure the ESI interface using a preference value that reflects the configured VPLS preference.



NOTE: Creating the ESI interface will cause the CE IFL to be deleted and recreated, so you should expect some traffic loss after this step.

```
[edit interfaces interface-name]
esi {
  00:01:02:03:04:05:06:00:00:01;
  single-active;
  df-election-type {
    preference {
      value match vpls preference;
    }
  }
}
```

6. Enable EVPN in a single routing instance.

- Change routing instance type from vpls to evpn, at the [edit routing-instances *routing-instance-name*] hierarchy level in your existing BGP VPLS configuratoin.

```
[edit routing-instances routing-instance-name]
instance-type evpn;
```

7. Include the evpn and vpls statements at the [edit routing-instances *routing-instance-name* protocols] hierarchy level in order to support EVPN and VPLS commands.

```
[edit routing-instances routing-instance-name]
protocols {
    evpn
    vpls
}
```

8. Once all nodes in the VPLS network have been migrated to EVPN, you can optionally decommission the VPLS protocol. This will allow EVPN to setup its single-homing and multi-homing state cleanly, but may result in traffic loss. To decommission the VPLS protocol, delete the protocols vpls statement under [edit routing-instances *routing-instance-name*] hierarchy.

```
[edit routing-instances routing-instance-name]
user@host# delete protocols vpls
```

After the configuration for the EVPN migration is committed, the routing protocol process and the Layer 2 address learning process start building the EVPN state to reflect interfaces, bridge domains, peers and routes. The locally learnt MAC addresses are synchronized by the Layer 2 address learning process in the instance.vpls.0 to the routing protocol process. When a local MAC ages out in the instance.vpls.0, the routing protocol process is informed by the Layer 2 address learning process.

When an EVPN peer is learnt, the routing protocol process sends a new message to the Layer 2 address learning process to remove the peer's label-switched interface or virtual tunnel logical interface from the VE mesh group and disables MAC-learning on it. The EVPN IM next-hop is then added to the VE mesh group. The EVPN behavior in the routing protocol process of learning MAC addresses over BGP and informing Layer 2 address learning process of the MPLS next hop is maintained.

The VPLS statements and commands continue to apply to the VPLS pseudowires between the PE devices and the MAC addresses learnt over them. The EVPN statements and commands apply to PE devices running EVPN.

Reverting to VPLS

If the EVPN migration runs into issues, you can revert back to VPLS until the issue is understood. The routing instance is reverted from a super PE to a VPLS PE in a non-catastrophic manner by enabling the following configuration:

```
[edit routing-instances routing-instance-name]  
user@host# set instance-type vpls  
user@host# delete protocols evpn
```

On reverting the EVPN migration to VPLS, the following happens:

1. The EVPN state information is deleted.
2. There is a trigger for withdrawal of EVPN control plane routes.
3. The routing protocol process sends a new message to the Layer 2 address learning process with the label-switched interface or the virtual tunnel logical interface for the routing instance and peer.
4. The label-switched or virtual tunnel interface adds the new message to the flood group and MAC learning is enabled.
5. The egress IM next hop is deleted by the routing protocols process, prompting the Layer 2 address learning process to remove it from the flood group.
6. Remote MAC addresses are learnt again over the label-switched interface or virtual tunnel logical interface.

BGP VPLS to EVPN Migration and Other Features

[Table 8 on page 88](#) describes the functionality of some of the related features, such as multihoming and integrated routing and bridging (IRB) with the BGP VPLS to EVPN migration.

Table 8: EVPN Migration and Other Features Support

Feature	Supported Functionality in EVPN Migration
MAC move	<p>MAC moves are supported between VPLS-only PE and super PE devices.</p> <p>When a MAC address moves from a VPLS-only PE to a super PE, it is learnt over BGP, and the routing protocol process informs the Layer 2 address learning process of the EVPN next hop to be updated in the foo.vpls.0 routing table.</p> <p>When a MAC address moves from a super PE to a VPLS-only PE, it is learnt in the Packet Forwarding Engine on the label-switched interface or virtual tunnel interface. The Layer 3 address learning process updates it to VPLS or the label-switched interface next hop.</p> <p>When the type 2 route is withdrawn by EVPN BGP, the MAC address is not deleted from the forwarding table, so there is no loss of data.</p> <p>The forwarding MAC table is shared by VPLS and EVPN. Some attributes, such as mac-table-size and mac-table-aging-time could be configured under both EVPN and VPLS. When there is a conflict, the values under EVPN take precedence.</p>
IRB	<p>No changes needed in IRB.</p> <p>On super PE, EVPN populates the /32 host routes learnt over MAC+IP type 2 routes from EVPN peers in a Layer 3 virtual routing and forwarding, while VPLS IRB forwarding using subnet routes work on sites still running VPLS.</p>
Hierarchical VPLS	<p>In a H-VPLS network with hub and spoke PE devices, when the hub PE is migrated to EVPN, local MAC addresses learnt over the access label-switched or virtual tunnel interface need to be advertised into BGP, so that the other EVPN-only PE devices or super PE devices can reach them.</p> <p>Take the following into consideration when migrating a H-VPLS network to EVPN:</p> <ul style="list-style-type: none"> • Hubs typically have local-switching enabled as inter-spoke traffic is forwarded through the hub. If spoke(s) alone is migrated to EVPN and spokes have Layer 3 or MPLS reachability to each other, the label-switched or virtual tunnel interface to the hub and EVPN next hop (remote spoke) is present in the VE floodgroup and this results in two copies of broadcast, unknown unicast, and multicast (BUM) traffic received by the remote spoke. An option to avoid this is to migrate the hub(s) to EVPN too. • EVPN is not aware of hierarchy. All peers are considered core-facing. Once hubs and spokes are migrated to EVPN, split horizon prevents the BUM traffic from being forwarded to other core-facing PE devices.

Table 8: EVPN Migration and Other Features Support (*Continued*)

Feature	Supported Functionality in EVPN Migration
ESI configuration	ESI is configured at the physical interface or port level.

RELATED DOCUMENTATION

| [EVPN Overview](#) | 1429

Configuring EVPN over Transport Class Tunnels

Use transport class to define the transport tunnels that offer the same type of service. You can apply the transport class to tunnels based on the customer's required traffic service, such as low-latency or high-bandwidth traffic. We support EVPN over transport class tunnels. For example, we can have an EVPN network with different types of service and use different colors to identify the different transport class. We can use gold to identify a transport class for the low-latency traffic and bronze to identify the transport class for all other traffic. By mapping the colors to the different tunnels, the device maps the EVPN packets to the configured next hop for the transport tunnel.

We support the following EVPN services over transport tunnels:

- EVPN-VPWS
- EVPN E-LAN
- EVPN E-Tree

You can configure EVPN over transport class tunnels on top of an existing underlay network. The basic steps for configuring EVPN over transport class tunnels are:

1. Define the transport class at the ingress or egress node. For this example, we define a gold and bronze transport class and assign community values to them.

```

routing-options {
  transport-class {
    auto-create;
    name gold {
      color 100;
    }
  }
}

```

```

    }
    name bronze {
        color 200;
    }
}
}

```

2. Provision the transport tunnel and assign the tunnel to a specific transport class at the ingress node of the tunnels.

- **BGP-CT**

For information about BGP-CT, see *Color-Based Traffic Engineering Configuration*.

```

mpls {
    label-switched-path toPE11-gold {
        to 10.1.1.1;
        transport-class gold;
    }
    label-switched-path toPE11-bronze {
        to 10.1.1.1;
        transport-class bronze;
    }
}

```

- **IS-IS Flex Algorithm**

For information about IS-IS flexible algorithm, see *Configuring Flexible Algorithm for Segment Routing*.

```

flex-algorithm 128 {
    definition {
    priority 13;
    }
    color 100;
    use-transport-class;
}
flex-algorithm 129 {
    definition {
        metric-type te-metric;
    }
    color 200;
}

```

```

    use-transport-class;
}

```

- **RSVP-TE**

For information about RSVP-TE, see *MPLS Traffic Engineering Configuration*.

```

regress@asbr12# show protocols mpls
label-switched-path toPE11-gold {
    to 10.1.1.1;
    transport-class gold;
}
label-switched-path toPE11-bronze {
    to 10.1.1.1;
    transport-class bronze;
}

```

- **SR-TE**

For information about SR-TE, see *Understanding Source Packet Routing in Networking (SPRING)*.

```

source-packet-routing {
    segment-list sl-pe1-pe3-gold {
        hop-1 label 1000112;
        hop-2 label 1000223;
    }
    segment-list sl-pe1-pe3-bronze {
        hop-1 label 1000112;
        hop-2 label 1000323;
    }
    source-routing-path lsp-pe1-pe3-gold {
        to 10.3.3.3;
        color 100;
        primary {
            sl-pe1-pe3-gold;
        }
    }
    source-routing-path lsp-pe1-pe3-bronze {
        to 10.3.3.3;
        color 200;
        primary {
            sl-pe1-pe3-bronze;
        }
    }
}

```

```

    }
  }
  use-transport-class;
}

```

3. Configure policies to import and export traffic.

```

policy-statement vrf-import-evpn {
  term a {
    from {
      protocol bgp;
      community rt-evpn;
    }
    then {
      community add map2gold;
      accept;
    }
  }
  term b {
    then reject;
  }
}

policy-statement vrf-export-evpn {
  term a {
    then {
      community add map2gold;
      community add rt-evpn;
      accept;
    }
  }
  term b {
    then reject;
  }
}

community map2gold members color:0:100;
community rt-evpn members target:100:2;

```

4. Configure EVPN services.

- **EVPN-VPWS**

```

routing-instances {
  evpn-vpws {
    instance-type evpn-vpws;
    protocols {
      evpn {
        interface ge-0/0/1.4 {
          vpws-service-id {
            local 102;
            remote 201;
          }
        }
      }
    }
  }
  interface ge-0/0/1.4;
  route-distinguisher 10.255.1.1:20;
  vrf-import vrf-import-evpnvpws;
  vrf-export vrf-export-evpnvpws;
}

```

- **EVPN E-LAN**

```

routing-instances {
  evpn {
    instance-type evpn;
    protocols {
      evpn;
    }
    vlan-id 2;
    routing-interface irb.1;
    interface ge-0/0/0.3;
    route-distinguisher 10.255.1.1:4;
    vrf-import vrf-import-evpn;
    vrf-export vrf-export-evpn;
  }
}

```

- **EVPN E-Tree**

```

xe-2/1/1 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 100 {
        encapsulation vlan-bridge;
        vlan-id 100;
        etree-ac-role root;
    }
}

evpn-etree {
    instance-type evpn;
    protocols {
        evpn {
            interface xe-2/1/1.100;
            evpn-etree;
        }
    }
    vlan-id 100;
    interface xe-2/1/1.100;
    route-distinguisher 10.255.0.1:100;
    vrf-import vrf-import-evpn;
    vrf-export vrf-export-evpn;
}

```

RELATED DOCUMENTATION

Color-Based Traffic Engineering Configuration

Configuring Policies for the VRF Table on PE Routers in VPNs

Example: Configuring EVPN-VPWS over Transport Class Tunnels

IN THIS SECTION

● Overview | 95

- Requirements | 95
- Topology | 95
- Configuration | 96
- Verification | 104

Overview

In this example, we use a basic network topology to configure EVPN-VPWS over two RSVP-TE transport class tunnels between PE1 and PE2. We define two transport class and assign different tunnels to the transport class. We configure the devices with the following features:

- EVPN-VPWS routing instance.
- RSVP-TE tunnels using MPLS LSP and BGP.
- Gold and bronze transport class.
- Policy filters specifying separate communities for the gold and bronze transport class.

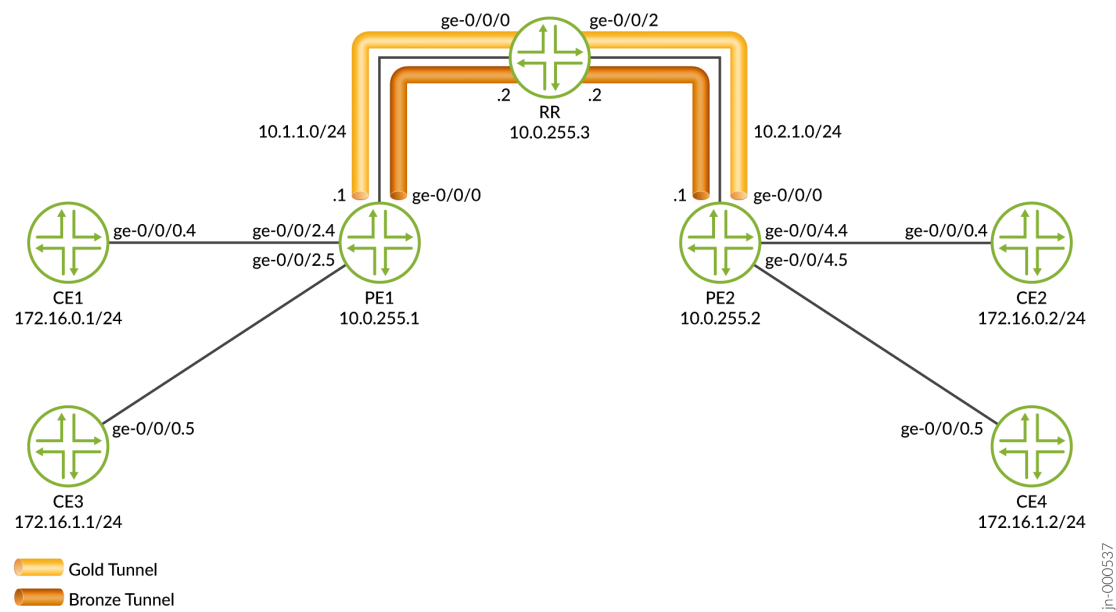
Requirements

- Two MX series routers configured as a PE router
- One MX series router configured as a route reflector
- Junos OS Release 23.1R1 or later running on all devices.

Topology

[Figure 10 on page 96](#) shows a basic topology with 2 PE devices and a route reflector forwarding the routes between the PE devices. A pair of CE devices is connected to PE1 and PE2 respectively. PE1 and PE2 routes traffic from CE1 and CE2 over the gold tunnel and traffic from CE3 and CE4 over the bronze tunnel.

Figure 10: Basic EVPN over RSVP-TE Transport Tunnel



Configuration

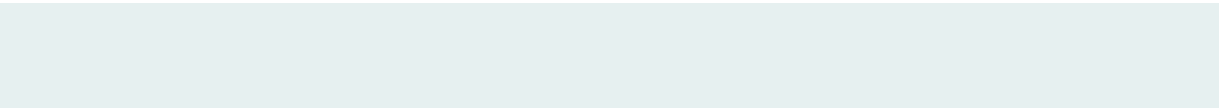
IN THIS SECTION

- [CLI Quick Configuration | 96](#)
- [Step-by-Step Procedure | 101](#)

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. The configuration for PE1, PE2, and RR are as follows:

PE1



```

set chassis network-services enhanced-ip
set interfaces ge-0/0/0 description pe1-rr
set interfaces ge-0/0/0 unit 0 family inet address 10.1.1.1/24
set interfaces ge-0/0/0 unit 0 family iso
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/2 vlan-tagging
set interfaces ge-0/0/2 encapsulation flexible-ethernet-services
set interfaces ge-0/0/2 unit 4 description pe1-ce1
set interfaces ge-0/0/2 unit 4 encapsulation vlan-ccc
set interfaces ge-0/0/2 unit 4 vlan-id 300
set interfaces ge-0/0/2 unit 5 description pe1-ce3
set interfaces ge-0/0/2 unit 5 encapsulation vlan-ccc
set interfaces ge-0/0/2 unit 5 vlan-id 301
set interfaces lo0 unit 0 family inet address 10.0.255.1/32
set interfaces lo0 unit 0 family iso address 49.0000.0010.0255.0001.00
set interfaces lo0 unit 0 family mpls
set policy-options policy-statement vrf-export-evpnvpws term a then community add rt-evpnvpws
set policy-options policy-statement vrf-export-evpnvpws term a then community add map2gold
set policy-options policy-statement vrf-export-evpnvpws term a then accept
set policy-options policy-statement vrf-export-evpnvpws term b then reject
set policy-options policy-statement vrf-export-evpnvpws-2 term a then community add rt-evpnvpws-2
set policy-options policy-statement vrf-export-evpnvpws-2 term a then community add map2bronze
set policy-options policy-statement vrf-export-evpnvpws-2 term a then accept
set policy-options policy-statement vrf-export-evpnvpws-2 term b then reject
set policy-options policy-statement vrf-import-evpnvpws term a from protocol bgp
set policy-options policy-statement vrf-import-evpnvpws term a from community rt-evpnvpws
set policy-options policy-statement vrf-import-evpnvpws term a then accept
set policy-options policy-statement vrf-import-evpnvpws term b then reject
set policy-options policy-statement vrf-import-evpnvpws-2 term a from protocol bgp
set policy-options policy-statement vrf-import-evpnvpws-2 term a from community rt-evpnvpws-2
set policy-options policy-statement vrf-import-evpnvpws-2 term a then accept
set policy-options policy-statement vrf-import-evpnvpws-2 term b then reject
set policy-options community map2bronze members color:0:200
set policy-options community map2gold members color:0:100
set policy-options community rt-evpnvpws members target:200:1
set policy-options community rt-evpnvpws-2 members target:300:1
set routing-instances evpn-vpws instance-type evpn-vpws
set routing-instances evpn-vpws protocols evpn interface ge-0/0/2.4 vpws-service-id local 102
set routing-instances evpn-vpws protocols evpn interface ge-0/0/2.4 vpws-service-id remote 201
set routing-instances evpn-vpws interface ge-0/0/2.4
set routing-instances evpn-vpws route-distinguisher 65000:1
set routing-instances evpn-vpws vrf-import vrf-import-evpnvpws
set routing-instances evpn-vpws vrf-export vrf-export-evpnvpws

```

```

set routing-instances evpn-vpws-2 instance-type evpn-vpws
set routing-instances evpn-vpws-2 protocols evpn interface ge-0/0/2.5 vpws-service-id local 103
set routing-instances evpn-vpws-2 protocols evpn interface ge-0/0/2.5 vpws-service-id remote 301
set routing-instances evpn-vpws-2 interface ge-0/0/2.5
set routing-instances evpn-vpws-2 route-distinguisher 65000:2
set routing-instances evpn-vpws-2 vrf-import vrf-import-evpnpws-2
set routing-instances evpn-vpws-2 vrf-export vrf-export-evpnpws-2
set routing-options route-distinguisher-id 10.0.255.1
set routing-options resolution preserve-nexthop-hierarchy
set routing-options router-id 10.0.255.1
set routing-options autonomous-system 65000
set routing-options transport-class auto-create
set routing-options transport-class name gold color 100
set routing-options transport-class name bronze color 200
set protocols bgp group BGP_PEERs type internal
set protocols bgp group BGP_PEERs local-address 10.0.255.1
set protocols bgp group BGP_PEERs family inet transport
set protocols bgp group BGP_PEERs family evpn signaling
set protocols bgp group BGP_PEERs neighbor 10.0.255.3
set protocols isis interface all
set protocols isis interface fxp0.0 disable
set protocols mpls label-switched-path pe1-rr to 10.0.255.3
set protocols mpls label-switched-path pe1-pe2 to 10.0.255.2
set protocols mpls label-switched-path pe1-pe2-gold to 10.0.255.2
set protocols mpls label-switched-path pe1-pe2-gold transport-class gold
set protocols mpls label-switched-path pe1-pe2-bronze to 10.0.255.2
set protocols mpls label-switched-path pe1-pe2-bronze transport-class bronze
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable

```

PE2

```

set chassis network-services enhanced-ip
set interfaces ge-0/0/0 description pe2-rr
set interfaces ge-0/0/0 unit 0 family inet address 10.2.1.1/24
set interfaces ge-0/0/0 unit 0 family iso
set interfaces ge-0/0/0 unit 0 family mpls

```

```

set interfaces ge-0/0/4 description pe2-ce2
set interfaces ge-0/0/4 vlan-tagging
set interfaces ge-0/0/4 encapsulation flexible-ethernet-services
set interfaces ge-0/0/4 unit 4 encapsulation vlan-ccc
set interfaces ge-0/0/4 unit 4 vlan-id 300
set interfaces lo0 unit 0 family inet address 10.0.255.2/32
set interfaces lo0 unit 0 family iso address 49.0000.0010.0255.0002.00
set interfaces lo0 unit 0 family mpls
set policy-options policy-statement vrf-export-evpnvpws term a then community add rt-evpnvpws
set policy-options policy-statement vrf-export-evpnvpws term a then community add map2bronze
set policy-options policy-statement vrf-export-evpnvpws term a then accept
set policy-options policy-statement vrf-export-evpnvpws term b then reject
set policy-options policy-statement vrf-import-evpnvpws term a from protocol bgp
set policy-options policy-statement vrf-import-evpnvpws term a from community rt-evpnvpws
set policy-options policy-statement vrf-import-evpnvpws term a then accept
set policy-options policy-statement vrf-import-evpnvpws term b then reject
set policy-options community map2bronze members color:0:200
set policy-options community map2gold members color:0:100
set policy-options community rt-evpnvpws members target:200:1
set routing-instances evpn-vpws instance-type evpn-vpws
set routing-instances evpn-vpws protocols evpn interface ge-0/0/4.4 vpws-service-id local 201
set routing-instances evpn-vpws protocols evpn interface ge-0/0/4.4 vpws-service-id remote 102
set routing-instances evpn-vpws interface ge-0/0/4.4
set routing-instances evpn-vpws route-distinguisher 65000:1
set routing-instances evpn-vpws vrf-import vrf-import-evpnvpws
set routing-instances evpn-vpws vrf-export vrf-export-evpnvpws
set routing-options route-distinguisher-id 10.0.255.2
set routing-options resolution preserve-nexthop-hierarchy
set routing-options router-id 10.0.255.2
set routing-options autonomous-system 65000
set routing-options transport-class auto-create
set routing-options transport-class name gold color 100
set routing-options transport-class name bronze color 200
set protocols bgp group BGP_PEERS type internal
set protocols bgp group BGP_PEERS local-address 10.0.255.2
set protocols bgp group BGP_PEERS family inet transport
set protocols bgp group BGP_PEERS family evpn signaling
set protocols bgp group BGP_PEERS neighbor 10.0.255.3
set protocols isis interface all
set protocols isis interface fxp0.0 disable
set protocols mpls label-switched-path pe2-pe1-gold to 10.0.255.1
set protocols mpls label-switched-path pe2-pe1-gold transport-class gold
set protocols mpls label-switched-path pe2-pe1 to 10.0.255.1

```

```

set protocols mpls label-switched-path pe2-pe1-bronze to 10.0.255.1
set protocols mpls label-switched-path pe2-pe1-bronze transport-class bronze
set protocols mpls label-switched-path pe2-rr to 10.0.255.3
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable

```

RR

```

set chassis network-services enhanced-ip
set interfaces ge-0/0/0 description rr-pe1
set interfaces ge-0/0/0 unit 0 family inet address 10.1.1.2/24
set interfaces ge-0/0/0 unit 0 family iso
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/2 description rr-pe2
set interfaces ge-0/0/2 unit 0 family inet address 10.2.1.2/24
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces lo0 apply-groups-except global
set interfaces lo0 unit 0 family inet address 10.0.255.3/32
set interfaces lo0 unit 0 family iso address 49.0000.0010.0255.0003.00
set interfaces lo0 unit 0 family mpls
set policy-options policy-statement pplb then load-balance per-packet
set routing-options route-distinguisher-id 10.0.255.3
set routing-options router-id 10.0.255.3
set routing-options autonomous-system 65000
set routing-options transport-class auto-create
set protocols bgp group BGP_PEERS type internal
set protocols bgp group BGP_PEERS local-address 10.0.255.3
set protocols bgp group BGP_PEERS family inet transport
set protocols bgp group BGP_PEERS family evpn signaling
set protocols bgp group BGP_PEERS cluster 10.0.255.3
set protocols bgp group BGP_PEERS neighbor 10.0.255.1
set protocols bgp group BGP_PEERS neighbor 10.0.255.2
set protocols isis interface all

```

```

set protocols isis interface fxp0.0 disable
set protocols mpls label-switched-path rr-pe1 to 10.0.255.1
set protocols mpls label-switched-path rr-pe2 to 10.0.255.2
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable

```

Step-by-Step Procedure

To set up transport class tunnels on the PE device:

1. Configure the device to support enhanced IP and tunnel services.

```

[edit]
set chassis network-services enhanced-ip

```

2. Configure the interfaces.

```

[edit]
set interfaces ge-0/0/0 description pe1-rr
set interfaces ge-0/0/0 unit 0 family inet address 10.1.1.1/24
set interfaces ge-0/0/0 unit 0 family iso
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/2 vlan-tagging
set interfaces ge-0/0/2 encapsulation flexible-ethernet-services
set interfaces ge-0/0/2 unit 4 description pe1-ce1
set interfaces ge-0/0/2 unit 4 encapsulation vlan-ccc
set interfaces ge-0/0/2 unit 4 vlan-id 300
set interfaces ge-0/0/2 unit 5 description pe1-ce3
set interfaces ge-0/0/2 unit 5 encapsulation vlan-ccc
set interfaces ge-0/0/2 unit 5 vlan-id 301
set interfaces lo0 unit 0 family inet address 10.0.255.1/32
set interfaces lo0 unit 0 family iso address 49.0000.0010.0255.0001.00
set interfaces lo0 unit 0 family mpls

```

3. Define the gold and bronze transport class on PE1.

```
[edit]
set routing-options transport-class auto-create
set routing-options transport-class name gold color 100
set routing-options transport-class name bronze color 200
```

4. Configure the routing protocols and the routing options to support transport-tunnels. We are using RSVP-TE with MPLS LSP and BGP.

```
[edit]
set protocols bgp group BGP_PEERs type internal
set protocols bgp group BGP_PEERs local-address 10.0.255.1
set protocols bgp group BGP_PEERs family inet transport
set protocols bgp group BGP_PEERs family evpn signaling
set protocols bgp group BGP_PEERs neighbor 10.0.255.3
set protocols isis interface all
set protocols isis interface fxp0.0 disable
set protocols mpls label-switched-path pe1-rr to 10.0.255.3
set protocols mpls label-switched-path pe1-pe2 to 10.0.255.2
set protocols mpls label-switched-path pe1-pe2-gold to 10.0.255.2
set protocols mpls label-switched-path pe1-pe2-gold transport-class gold
set protocols mpls label-switched-path pe1-pe2-bronze to 10.0.255.2
set protocols mpls label-switched-path pe1-pe2-bronze transport-class bronze
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set routing-options route-distinguisher-id 10.0.255.1
set routing-options resolution preserve-nexthop-hierarchy
set routing-options router-id 10.0.255.1
set routing-options autonomous-system 65000
```

5. Configure the policy to apply the EVPN routes to the transport tunnels. We use vrf-export to advertise the tunnel to the remote PE.

```
[edit]

set policy-options policy-statement vrf-export-evpnvpws term a then community add rt-evpnvpws
```

```

set policy-options policy-statement vrf-export-evpnvpws term a then community add map2gold
set policy-options policy-statement vrf-export-evpnvpws term a then accept
set policy-options policy-statement vrf-export-evpnvpws term b then reject
set policy-options policy-statement vrf-export-evpnvpws-2 term a then community add rt-
evpnvpws-2
set policy-options policy-statement vrf-export-evpnvpws-2 term a then community add map2bronze
set policy-options policy-statement vrf-export-evpnvpws-2 term a then accept
set policy-options policy-statement vrf-export-evpnvpws-2 term b then reject
set policy-options policy-statement vrf-import-evpnvpws term a from protocol bgp
set policy-options policy-statement vrf-import-evpnvpws term a from community rt-evpnvpws
set policy-options policy-statement vrf-import-evpnvpws term a then accept
set policy-options policy-statement vrf-import-evpnvpws term b then reject
set policy-options policy-statement vrf-import-evpnvpws-2 term a from protocol bgp
set policy-options policy-statement vrf-import-evpnvpws-2 term a from community rt-evpnvpws-2
set policy-options policy-statement vrf-import-evpnvpws-2 term a then accept
set policy-options policy-statement vrf-import-evpnvpws-2 term b then reject
set policy-options community map2bronze members color:0:200
set policy-options community map2gold members color:0:100
set policy-options community rt-evpnvpws members target:200:1
set policy-options community rt-evpnvpws-2 members target:300:1

```

6. Configure the EVPN-VPWS routing instance that will be using the policy above.

```

[edit]
set routing-instances evpn-vpws instance-type evpn-vpws
set routing-instances evpn-vpws protocols evpn interface ge-0/0/2.4 vpws-service-id local 102
set routing-instances evpn-vpws protocols evpn interface ge-0/0/2.4 vpws-service-id remote 201
set routing-instances evpn-vpws interface ge-0/0/2.4
set routing-instances evpn-vpws route-distinguisher 65000:1
set routing-instances evpn-vpws vrf-import vrf-import-evpnvpws
set routing-instances evpn-vpws vrf-export vrf-export-evpnvpws
set routing-instances evpn-vpws-2 instance-type evpn-vpws
set routing-instances evpn-vpws-2 protocols evpn interface ge-0/0/2.5 vpws-service-id local
103
set routing-instances evpn-vpws-2 protocols evpn interface ge-0/0/2.5 vpws-service-id remote
301
set routing-instances evpn-vpws-2 interface ge-0/0/2.5
set routing-instances evpn-vpws-2 route-distinguisher 65000:2
set routing-instances evpn-vpws-2 vrf-import vrf-import-evpnvpws-2
set routing-instances evpn-vpws-2 vrf-export vrf-export-evpnvpws-2

```



```
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 3.631/4.380/7.976/0.608 ms
```

From operational mode on PE1, run the `show mpls lsp statistics` command to display the LSP information.

```
user@PE1> show mpls lsp statistics
Ingress LSP: 4 sessions
```

To	From	State	Packets	Bytes	LSPname
10.0.255.2	10.0.255.1	Up	0	0	pe1-pe2
10.0.255.2	10.0.255.1	Up	20	2040	pe1-pe2-bronze
10.0.255.2	10.0.255.1	Up	10	1020	pe1-pe2-gold
10.0.255.3	10.0.255.1	Up	0	0	pe1-rr

Meaning

The output shows that the ping was successful. The output from the `show mpls lsp statistics` command shows that packets were routed to the bronze and gold tunnels.

Verify Configured Transport Tunnels

Purpose

Verify that the EVPN uses the configured transport tunnel.

Action

From operational mode on PE1, run the `show route table mpls.0 protocol evpn` command to identify the LSP route.

```
user@PE1> show route table mpls.0 protocol evpn
mpls.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

94	*[EVPN/7] 6d 01:07:24
	> via ge-0/0/2.4, Pop
105	*[EVPN/7] 06:20:50
	> via ge-0/0/2.5, Pop
106	*[EVPN/7] 06:17:47, remote-pe 10.0.255.2, routing-instance evpn-vpws-2, route-type Egress, vlan-id 301
	> to 10.1.1.2 via ge-0/0/0.0, label-switched-path pe1-pe2-bronze

```

107          *[EVPN/7] 06:08:41, remote-pe 10.0.255.2, routing-instance evpn-vpws, route-
type Egress, vlan-id 201
          > to 10.1.1.2 via ge-0/0/0.0, label-switched-path pe1-pe2-gold
ge-0/0/2.5    *[EVPN/7] 06:17:47, route-type Egress
          > to 10.1.1.2 via ge-0/0/0.0, label-switched-path pe1-pe2-bronze
ge-0/0/2.4    *[EVPN/7] 06:08:41, route-type Egress
          > to 10.1.1.2 via ge-0/0/0.0, label-switched-path pe1-pe2-gold

```

From operational mode on PE1, run the `show route table mpls.0 protocol evpn label label-number extensive` command with the route label number to display the transport class information.

```

user@PE1> show route table mpls.0 protocol evpn label 107 extensive

mpls.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
107 (1 entry, 1 announced)
TSI:
KRT in-kernel 107    /52 -> {composite(554)}
    *EVPN    Preference: 7
              Next hop type: Indirect, Next hop index: 0
              Transport class: gold
              Address: 0x7b406d4
              Next-hop reference count: 5, key opaque handle: 0x0, non-key opaque handle: 0x0
              Next hop type: Router, Next hop index: 549
              Next hop: 10.1.1.2 via ge-0/0/0.0, selected
              Label-switched-path pe1-pe2-gold
.
.
.

regress@PE1> show route table mpls.0 protocol evpn label 106 extensive

mpls.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
106 (1 entry, 1 announced)
TSI:
KRT in-kernel 106    /52 -> {composite(553)}
    *EVPN    Preference: 7
              Next hop type: Indirect, Next hop index: 0
              Transport class: bronze
              Address: 0x7b40584
              Next-hop reference count: 5, key opaque handle: 0x0, non-key opaque handle: 0x0
              Next hop type: Router, Next hop index: 551

```

```
Next hop: 10.1.1.2 via ge-0/0/0.0, selected  
Label-switched-path pe1-pe2-bronze
```

```
.  
.   
.
```

Meaning

The output from PE1 and PE2 shows that we are routing traffic from our EVPN VPWS routing instance through the gold and bronze tunnels.

CHAPTER 4

Configuring Route Targets

IN THIS CHAPTER

- [Auto-derived Route Targets | 108](#)
- [Example: Configuring VNI Route Targets Automatically | 112](#)
- [Example: Configuring VNI Route Targets Manually | 115](#)
- [Example: Configuring VNI Route Targets Automatically with Manual Override | 120](#)

Auto-derived Route Targets

IN THIS SECTION

- [Benefits of Auto-Derived Route Targets | 108](#)
- [Understanding Auto-Derived Route Targets | 109](#)

Benefits of Auto-Derived Route Targets

Auto-derived route targets simplify the configuration of VLAN services for EVPN, especially in VLAN-aware bundle services where you can have multiple VLANs, multiple bridge domains and the VLANs for a given service that are not present on all PE devices. Without the auto-derived target option enabled, EVPN Type 2 and Type 3 routes are imported into the EVPN instances (EVIs) on all receiving PE devices and the routes subsequently dropped for non-existing VLANs (bridge-domains). To minimize the number of routes that are distributed, different auto-derived route targets can be used within each bridge-domain. Together with constrained route distribution as described in RFC 4684, you can limit the distribution of bridge domain-specific EVPN route types (Type 2 and Type 3) to only the interested PE devices.

Understanding Auto-Derived Route Targets

Route targets identify the different routes that are imported and exported into the VRF tables. When you enable the auto-derived route targets option, the device derives the route targets based on the EVPN encapsulation for EVPN route Type 2 (MAC/IP Advertisement Route) and EVPN route Type 3 (Inclusive Multicast Ethernet Tag route).

Devices don't auto-derive the route targets for EVPN route types other than Type 2 and Type 3. As a result, in some cases you must manually configure route targets in routing instances for EVPN route types such as:

- EVPN Type 1 routes: Devices use these routes to reach all the multihomed devices associated with a ESI. You configure route targets for this type of route manually in the EVPN instance.
- EVPN Type 5 routes: Devices use these routes to advertise IP prefixes assigned within a data center to devices in other data centers, enabling communication across data centers. You configure route targets for this type of route manually in Type 5 virtual routing and forwarding (VRF) instances.

See ["Example: Configuring VNI Route Targets Manually" on page 115](#) and ["Example: Configuring VNI Route Targets Automatically with Manual Override" on page 120](#) for more on configuring route targets manually at the global level (applicable to all EVPN route types) or manually at the virtual network identifier (VNI) level (applicable to EVPN Type 2 and Type 3 routes).

Devices automatically derive route targets for EVPN Type 2 and Type 3 routes based on the following parameters:

- For EVPN-MPLS: From the VLAN ID (VID).
- For EVPN-VXLAN: From the VXLAN network identifier (VNI).
- For PBB-EVPN: From the instance service identifier (ISID).

For EVPN Type 2 and Type 3 routes, the auto-derived route targets have higher precedence over route targets you configure manually at the global level in `vrf-target` statements, `vrf-export` policies, and `vrf-import` policies.

As defined in [RFC8365](#), the auto-derived route target field includes the following fields:

- Global Administrator—A 2-octet field containing an autonomous system (AS) number assigned by Internet Assigned Numbers Authority (IANA).
- Local Administrator—A 4-Octet field that includes the following:
 - A single bit field with a value of zero indicating that the RT is auto-derived.
 - Type—A 3-bit field identifying the service.
 - D-ID—A 4-bit field identifying the domain ID.

- Service ID—A 3-octet field set to the VNI, VSID, I-SID, or VID.



NOTE: We don't support auto-derived route targets for inter-AS routing.

To enable auto-derived route targets, include the `auto` statement at the [edit routing-instances *routing-instance-name* vrf-target]. We support configuring auto-derived route targets with these L2 instance types:

- The default switch instance at the [edit switch-options] hierarchy level.
- Virtual switch instances using instance-type `virtual-switch` at the [edit routing-instances *virtual-switch-instance-name*] hierarchy level.
- EVPN instances using:
 - instance-type `evpn` at the [edit routing-instances *evpn-instance-name*] hierarchy level.
 - instance-type `mac-vrf` at the [edit routing-instances *mac-vrf-instance-name*] hierarchy level.

The following is a sample configuration for auto-derived route targets for an EVPN instance and a virtual switch routing instance. We also manually configure a route target here in either type of EVPN instance at the global level to support EVPN Type 1 routes. The auto-derived route target applies to EVPN Type 2 and Type 3 routes, and for those route types, takes precedence over the manually defined route target at the global level.

```
routing-instances {
  VS-1 {
    instance-type virtual-switch;
    interface ae0.110;
    interface ae1.120;
    interface ae2.130;
    route-distinguisher 100.100.100.2:101;
    vrf-target {
      target:100:101;
      auto;
    }
    protocols {
      evpn {
        extended-vlan-list [ 110 120 130 ];
      }
    }
    bridge-domains {
      bd-110 {
        vlan-id 110;
      }
    }
  }
}
```

```
    }  
    bd-120 {  
        vlan-id 120;  
    }  
    bd-130 {  
        vlan-id 130;  
    }  
}  
}  
EVPN-1 {  
    instance-type evpn;  
    vlan-id 10;  
    interface ae0.0;  
    interface ae1.0;  
    interface ae2.0;  
    route-distinguisher 100.100.100.2:1;  
    vrf-target {  
        target:100:1  
        auto;  
    }  
    protocols {  
        evpn;  
    }  
}
```

RELATED DOCUMENTATION

[Example: Configuring VNI Route Targets Automatically | 112](#)

[Example: Configuring VNI Route Targets Manually | 115](#)

[Example: Configuring VNI Route Targets Automatically with Manual Override | 120](#)

Example: Configuring VNI Route Targets Automatically

IN THIS SECTION

- [Requirements | 112](#)
- [Overview | 112](#)
- [Configuration | 113](#)

This example shows how to automatically derive route targets for multiple VNIs in an EVPN-VXLAN topology.

Requirements

This example uses the following hardware and software components:

- A QFX Series switch.
- Junos OS version 15.1X53-D30

Overview

You can configure the `vrf-target` statement with the `auto` option to automatically derives route targets in a routing instance. Devices auto-derive route targets only for EVPN Type 2 and Type 3 routes. As a result, in some cases you must also manually configure route targets for other EVPN route types such as Type 1 (in L2 routing instances) or Type 5 (in L3 routing instances). See ["Example: Configuring VNI Route Targets Manually" on page 115](#) for manual route target configuration examples.

You can also manually set specific route targets for individual virtual network identifiers (VNIs) if you configure the `vrf-target` statement at the `[edit <routing-instances name> protocols evpn vni-options vni vni]` hierarchy level. Like auto-derived route targets, VNI level route targets apply only to EVPN Type 2 and Type 3 routes. See ["Example: Configuring VNI Route Targets Manually" on page 115](#) and ["Example: Configuring VNI Route Targets Automatically with Manual Override" on page 120](#) for VNI-level route target configuration examples.

In this example we show how to configure auto-derived route targets for EVPN Type 2 and Type 3 routes for all VNIs in an EVPN instance.



NOTE: We also include configuring a route target manually to support EVPN Type 1 routes in the instance.

Configuration

IN THIS SECTION

- [Configuring VNI Route Target Automatic Derivation | 113](#)
- [Results | 114](#)

To configure the `vrf-target` statement with the `auto` option, perform these tasks:

Configuring VNI Route Target Automatic Derivation

Step-by-Step Procedure

To configure the automatic derivation of VNI route targets:

1. At the `[switch-options]` hierarchy level, configure the `vtep-source-interface`, and `route-distiguisher` statements. Then configure the `vrf-import` statement with a policy that will be configured in a later step. Next, configure the `vrf-target` statement with a target and the `auto` option. Type 1 EVPN routes will use the route target you configure under `vrf-target`. Type 2 and Type 3 EVPN routes will use the auto-derived route target for export and import for all VNIs.

```
[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 192.0.2.11:1
user@switch# set vrf-import import-policy
user@switch# set vrf-target target:1111:11
user@switch# set vrf-target auto
```

2. At the `[evpn]` hierarchy level, configure the `encapsulation` and `extended-vni-list` statements. For the purposes of this example, the `extended-vni-list` statement will be configured with `all`, to apply automatic route targets to all VNIs.

```
[edit protocols evpn]
user@switch# set encapsulation vxlan
user@switch# set extended-vni-list all
```

3. Configure a community at the [edit policy-options] hierarchy level named `comglobal`. Configure an import policy named `import-policy` at the [edit policy-options policy-statement] hierarchy level. This community and this policy function as an import filter that accepts routes with the auto-derived route target. You apply the policy in step ["1" on page 113](#).

```
[edit policy-options]
user@switch# set community comglobal members target:1111:11
```

```
[edit policy-options policy-statement import-policy]
user@switch# set term 1 from community comglobal
user@switch# set term 1 then accept
user@switch# set term 100 then reject
```

Results

Use the `show` command to verify the results of your configuration.

```
user@switch> show configuration switch-options
vtep-source-interface lo0.0;
route-distinguisher 192.0.2.11:1;
vrf-import imp;
vrf-target {
    target:1111:11;
    auto;
}
```

```
user@switch> show configuration protocols evpn
encapsulation vxlan;
extended-vni-list all;
```

```
user@switch> show configuration policy-options community comglobal
members target:1111:11;
```

```
user@switch> show configuration policy-options policy-statement import-policy
term 1{
```

```

    from community [ comglobal ];
    then accept;
}
term 100 {
    then reject;
}

```

RELATED DOCUMENTATION

[Auto-derived Route Targets | 108](#)

vrf-target

[Example: Configuring VNI Route Targets Automatically with Manual Override | 120](#)

Example: Configuring VNI Route Targets Manually

IN THIS SECTION

- [Requirements | 115](#)
- [Overview | 116](#)
- [Configuration | 117](#)

This example shows how to manually set route targets for multiple virtual network identifiers (VNIs) in an EVPN-VXLAN topology.

Requirements

This example uses the following hardware and software components:

- A QFX Series switch.
- Junos OS version 15.1X53-D30

Overview

If you configure the `vrf-target` statement with the `auto` option, the device automatically derives route targets in a routing instance. You can also configure `vrf-target` with a specific *target* value to configure a route target manually in a routing instance.

In this example we show how to configure route targets manually by specifying a *target* value. You can manually configure route targets using the `vrf-target` statement at the following levels:

- Global level—For all VNIs extended in a routing instance.

This setting applies to all EVPN route types.

- vni-options level—For a specific VNI in a routing instance.

This setting applies only to EVPN Type 2 and Type 3 routes. With this setting:

- If you also manually configure a route target at the global level, for the specified VNIs, this setting overrides the global setting.
- If you also set the `auto` option to auto-derive route targets, for the specified VNIs, this setting overrides the auto setting.

[Table 9 on page 116](#) shows the corresponding CLI hierarchies in which you can manually configure a route target:

Table 9: Configuration Levels for `vrf-target` Statement

Configuration Level	Default Switch Instance Hierarchy	Configured Routing Instance Hierarchy	Applicable EVPN Route Types
Global (all VNIs) in Routing Instance	[edit switch-options]	[edit routing-instances <i>name</i>]	All EVPN Route Types
Specified VNI in Routing Instance	[edit protocols evpn vni-options vni <i>vni</i>]	[edit routing-instances <i>name</i> protocols evpn vni-options vni <i>vni</i>]	Type 2 Type 3

We include sample configurations at each level next.

Configuration

IN THIS SECTION

- [Configure VNI Route Targets Manually for all VNIs in the Default Switch Instance | 117](#)
- [Configure VNI Route Targets Manually for all VNIs and Specific VNIs in a MAC-VRF EVPN Instance | 118](#)

This section shows some use cases to manually configure VNI route targets at the supported configuration levels.

Configure VNI Route Targets Manually for all VNIs in the Default Switch Instance

Step-by-Step Procedure

This procedure shows how to configure a route target manually for all VNIs in the default switch instance. This is a global level manual route target configuration.

1. At the [edit switch-options] hierarchy level, configure the `vtep-source-interface` and `route-distinguisher` statements. Next, configure the `vrf-target` statement with a *target* value. All EVPN routes for all VLANs and corresponding VNIs will use the `vrf-target` address configured in this step.

```
[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 192.168.1.11:1
user@switch# set vrf-target target:1111:11
```



NOTE: You can optionally include `vrf-import` and `vrf-export` policies to further distinguish the routes to import and export that match the route target.

2. At the `[edit protocols evpn]` hierarchy level, configure EVPN with VXLAN encapsulation, and specify the VNIs you want to extend into the EVPN instance. In this example, we configure the `extended-vni-list` statement with the `all` option to apply the route target to all VNIs.

```
[edit protocols evpn]
user@switch# set encapsulation vxlan
user@switch# set extended-vni-list all
```

Results

After following the steps above and committing the configuration, use the `show configuration` command to verify the results of your configuration.

```
user@switch> show configuration switch-options
vtep-source-interface lo0.0;
route-distinguisher 192.168.1.11:1;
vrf-target {
    target:1111:11;
}
user@switch> show configuration protocols evpn
encapsulation vxlan;
extended-vni-list all;
.
.
.
```

Configure VNI Route Targets Manually for all VNIs and Specific VNIs in a MAC-VRF EVPN Instance

Step by Step Procedure

This procedure shows how to configure a route target manually in an EVPN-VXLAN fabric for all VNIs in a MAC-VRF instance (MAC-VRF1) with the `vlan-aware` service type. We also manually configure a different route target specifically for VNI 100 in the same instance.

1. At the `[edit routing-instances MAC-VRF1]` hierarchy level, configure the routing instance with the `mac-vrf` instance type and service type `vlan-aware`. Set the `vtep-source-interface` and `route-distiguisher` statements for the instance. Manually configure a `vrf-target` value at the global level in the routing instance. This

global level route target corresponds to EVPN Type 1, Type 2, and Type 3 routes for the VNIs you extend into the EVPN instance (see the next step).

```
[edit routing-instances MAC-VRF1]
user@switch# set vtep-source-interface lo0.0
user@switch# set instance-type mac-vrf
user@switch# set service-type vlan-aware
user@switch# set route-distinguisher 192.168.2.11:1
user@switch# set vrf-target target:1111:11
```

2. At the [edit routing instances MAC-VRF1 protocols evpn] hierarchy level, configure EVPN with VXLAN encapsulation. List the VNIs you want to extend into the EVPN instance using the extended-vni-list statement. In this example, we list two VNIs, 100 and 101. We also configure an export route target value at the [edit routing-instances *name* protocols evpn vni-options vni *vni*] hierarchy level specifically for VNI 100. As a result, the route target you set the previous step applies to any EVPN Type 1 routes and only to VNI 101 for EVPN Type 2 and Type 3 routes. The VNI level route target you set in this step applies to VNI 100 for EVPN Type 2 and Type 3 routes.

```
[edit routing instances MAC-VRF1 protocols evpn]
user@switch# set encapsulation vxlan
user@switch# set extended-vni-list 100 101
user@switch# set vni-options vni 100 vrf-target export target:1234:11
```



NOTE: You can optionally include vrf-import and vrf-export policies to further distinguish the routes to import and export that match the route target.

Results

After following the steps above and committing the configuration, use the `show configuration` command to verify the results of your configuration.

```
user@switch> show configuration routing-instances MAC-VRF1
vtep-source-interface lo0.0;
instance-type mac-vrf;
service-type vlan-aware;
route-distinguisher 192.168.2.11:1;
vrf-target {
    target:1111:11;
```



```

}
protocols {
  evpn {
    encapsulation vxlan;
    extended-vni-list 100 101;
    vni-options {
      vni 100 {
        vrf-target export target:1234:11;
      }
    }
  }
}
.
.
.

```

RELATED DOCUMENTATION

vrf-target

[Auto-derived Route Targets | 108](#)

[Example: Configuring VNI Route Targets Automatically | 112](#)

[Example: Configuring VNI Route Targets Automatically with Manual Override | 120](#)

Example: Configuring VNI Route Targets Automatically with Manual Override

IN THIS SECTION

- [Requirements | 121](#)
- [Overview | 121](#)
- [Configuration | 121](#)

This example shows how to automatically set route targets for multiple VNIs, and manually override the route target for a single VNI in an EVPN-VXLAN topology.

Requirements

This example uses the following hardware and software components:

- A QFX Series switch.
- Junos OS version 15.1X53-D30

Overview

You can configure the `vrf-target` statement to automatically derive route targets for each VNI or you can configure route targets manually. You can also manually override the auto-derived route target specifically for one or more VNIs.

This example explains how to enable auto-derivation of route targets and manually override the automatically assigned route targets for a specific VNI.

To manually assign a route target for a specific VNI, you configure the `vrf-target` statement with a route target *value* at either of the following hierarchy levels:

- In the default switch instance—[edit protocols evpn vni-options vni *vni*]
- In a configured routing instance—[edit routing-instances *name* protocols evpn vni-options vni *vni*]

When you manually set a route target for a specific VNI, the setting applies only to EVPN Type 2 and Type 3 routes.



NOTE: To set a route target for other EVPN route types, such as Type 1 routes, you must also manually configure the `vrf-target` statement with a route target *value* at the global level, as follows:

- In the default switch instance—[edit switch-options] hierarchy level
- In a configured routing instance—[edit routing-instances *name*] hierarchy level

For EVPN Type 2 and Type 3 routes only, the auto route target setting or a VNI level route target setting overrides a route target you set manually at the global level.

Configuration

IN THIS SECTION

- [Configuring Automatic VNI Route Targets with Manual Override | 122](#)
- [Results | 123](#)

To manually override an automatically configured VNI route target, perform these tasks:

Configuring Automatic VNI Route Targets with Manual Override

Step-by-Step Procedure

To configure automatic VNI route targets with manual override:

1. At the `[switch-options]` hierarchy level, configure the `vtep-source-interface`, and `route-distiguisher` statements. Then configure the `vrf-import` statement with a policy that will be configured in a later step. Next, configure the `vrf-target` statement with a target and the `auto` option. The route target configured under `vrf-target` will be used by Type 1 EVPN routes and all Type 2 and Type 3 EVPN routes for all VLANs except the ones that do not match the VNI under `vni-options` in the next step.

```
[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 192.0.2.11:1
user@switch# set vrf-import import-policy
user@switch# set vrf-target target:1111:11
user@switch# set vrf-target auto
```

2. The `[evpn]` hierarchy level is where you can override the automatic assignment of VNI route targets. Configure the `vni-options` statement for VNI 100 with an export target of 1234:11. This route target will be used by Type 2 and Type 3 EVPN routes for all VLANs that match VNI 100. Next, configure the `encapsulation` and `extended-vni-list` statements. For the purposes of this example, the `extended-vni-list` statement will be configured with only two VNIs.

```
[edit protocols evpn]
user@switch# set vni-options vni 100 vrf-target export target:1234:11
user@switch# set encapsulation vxlan
user@switch# set extended-vni-list 100 101
```

3. Configure two communities at the `[edit policy-options community]` hierarchy level. The first community is named `comglobal`, and the next community is named `com1234`. Configure an import policy at the `[edit policy-options policy-statement]` hierarchy level. The policy is named `import-policy`, which you apply in

the first step in this procedure. The communities and policy function as an import filter that accepts routes using the auto-derived route target and the manual override route target.

```
[edit policy-options community comglobal]
user@switch# set members target:1111:11
```

```
[edit policy-options community com1234]
user@switch# set members target:1234:11
```

```
[edit policy-options policy-statement import-policy]
user@switch# set term 1 from community comglobal com1234
user@switch# set term 1 then accept
user@switch# set term 100 then reject
```

Results

After following the steps above, use the `show` command to verify the results of your configuration.

```
user@switch> show configuration switch-options
vtep-source-interface lo0.0;
route-distinguisher 192.0.2.11:1;
vrf-import imp;
vrf-target {
    target:1111:11;
    auto;
}
```

```
user@switch> show configuration protocols evpn
vni-options {
    vni 100 {
        vrf-target export target:1234:11;
    }
}
```

```
encapsulation vxlan;  
extended-vni-list [ 100 101 ];
```

```
user@switch> show configuration policy-options community comglobal  
members target:1111:11;
```

```
user@switch> show configuration policy-options community com1234  
members target:1234:11;
```

```
user@switch> show configuration policy-options policy-statement import-policy  
term 1{  
    from community [ com1234 comglobal ];  
    then accept;  
}  
term 100 {  
    then reject;  
}
```

RELATED DOCUMENTATION

vrf-target

[Example: Configuring VNI Route Targets Automatically | 112](#)

[Example: Configuring VNI Route Targets Manually | 115](#)

Routing Policies for EVPN

IN THIS CHAPTER

- [Routing policies for EVPN | 125](#)
- [Example: Using Policy Filters to Filter EVPN Routes | 132](#)

Routing policies for EVPN

SUMMARY

Create routing policies to control the EVPN routing information that will be imported and exported to the different routing tables.

Routing policies allow you to control the routing information that the device imports and exports to the routing and forwarding tables. Starting in Junos OS 19.4R1, Junos has expanded routing policy support to include creating and applying policy filters specific to EVPN routes.

You can apply policies at the routing-instance level or at the BGP group level. When you apply policies at the BGP group level, they affect all EVPN routing instances. When applied at the routing-instance level, they affect the specified EVPN routing instance only. To apply the policy at the BGP group level, include the `vpn-apply-export` statement at the `[edit protocols bgp]` hierarchy level, and import or export the policy. To apply the policy at the routing-instance level, use the `vrf-export` or `vrf-import` statement to apply the policy for that routing instance.



NOTE: You can modify EVPN Type 1 route advertisements through `vrf-export` policy configuration. You must configure the policy with the `accept` clause for the target communities you want advertised or they will be rejected by the default policy action.

Policies are composed of match conditions, actions, and terms. For more information on policies, see [Policy Framework Overview](#).

[Table 10 on page 127](#) lists the match conditions supported for use in filtering EVPN routes.

Table 10: List of Match Conditions for Filtering EVPN Routes

Match Condition	Description																																			
<code>community [names]</code>	<p>BGP EVPN routes can have a set of EVPN extended communities carried in the BGP update message path attribute, and as such, you can use these extended communities for filtering BGP EVPN routes. The EVPN specific information available in extended communities includes, for example, encapsulation type, MAC-mobility information, EVPN split-horizon label information, EVPN ESI split-horizon label, ESI mode, E-tree leaf label, and more.</p> <p>Use the following syntax to specify BGP EVPN extended communities:</p> <ul style="list-style-type: none"><code>set policy-options community <i>name</i> members type:val1:val2</code> <p>All values (including type) are in decimal; type is 2 octets, with the higher-order octet defining the type of extended community, and the low-order octet defining the community sub-type. <i>val1</i> and <i>val2</i> can be specified as [2 + 4] octets, or as [4 + 2] octets.</p> <p>The extended communities most commonly used with BGP EVPN routes are provided here.</p> <table><tr><th>High-order</th><th>Low-order (Sub-type)</th><th>Type (Hex)</th><th>Type (Dec)</th><th>Name</th></tr><tr><td>0x03</td><td>0x0c</td><td>0x030c</td><td>780</td><td>BGP Encapsulation</td></tr><tr><td>0x03</td><td>0x0d</td><td>0x030d</td><td>781</td><td>Default Gateway</td></tr><tr><td>0x06</td><td>0x00</td><td>0x0600</td><td>1536</td><td>EVPN MAC Mobility</td></tr><tr><td>0x06</td><td>0x01</td><td>0x0601</td><td>1537</td><td>EVPN ESI Label</td></tr><tr><td>0x06</td><td>0x02</td><td>0x0602</td><td>1538</td><td>EVPN ES-Import Route Target</td></tr><tr><td>0x06</td><td>0x04</td><td>0x0604</td><td>1540</td><td>EVPN Layer 2 Attributes</td></tr></table>	High-order	Low-order (Sub-type)	Type (Hex)	Type (Dec)	Name	0x03	0x0c	0x030c	780	BGP Encapsulation	0x03	0x0d	0x030d	781	Default Gateway	0x06	0x00	0x0600	1536	EVPN MAC Mobility	0x06	0x01	0x0601	1537	EVPN ESI Label	0x06	0x02	0x0602	1538	EVPN ES-Import Route Target	0x06	0x04	0x0604	1540	EVPN Layer 2 Attributes
High-order	Low-order (Sub-type)	Type (Hex)	Type (Dec)	Name																																
0x03	0x0c	0x030c	780	BGP Encapsulation																																
0x03	0x0d	0x030d	781	Default Gateway																																
0x06	0x00	0x0600	1536	EVPN MAC Mobility																																
0x06	0x01	0x0601	1537	EVPN ESI Label																																
0x06	0x02	0x0602	1538	EVPN ES-Import Route Target																																
0x06	0x04	0x0604	1540	EVPN Layer 2 Attributes																																

Table 10: List of Match Conditions for Filtering EVPN Routes (Continued)

Match Condition	Description					
	<table><tr><td>0x06</td><td>0x05</td><td>0x0605</td><td>1541</td><td>EVPN E-Tree</td></tr></table>	0x06	0x05	0x0605	1541	EVPN E-Tree
0x06	0x05	0x0605	1541	EVPN E-Tree		
	<p>For full list of Extended Communities please refer to Border Gateway Protocol (BGP) Extended Communities .</p>					
evpn-esi	<p>You can filter BGP EVPN routes on the basis of Ethernet Segment Identifiers (ESIs) information for routes types 1, 2, 4, 7, and 8, which are the only types to include the ESI attribute in their prefix. (ESI values are encoded as 10-byte integers and are used to identify a multihomed segment.) Note that the evpn-esi matching statement is valid only together with “family evpn” matching statement.</p>					
evpn-etag	<p>You can filter BGP EVPN routes on the basis of EVPN Ethernet Tag information, which is part of the prefix of the EVPN route. This matching statement is valid only together with family evpn match statement.</p>					
evpn-mac-route	<p>Filtering BGP EVPN Type 2 routes based on if it has any IP address.</p> <p>EVPN Type 2 MAC/IP Advertisement routes can have IP address in the prefix along with MAC address. The IP address carried in the MAC-IP Advertisement route can be either IPv4 or IPv6 address. It is possible to filter out Type 2 routes based on MAC address only, MAC+IPv4 address, or MAC+IPv6 address. To do so requires the following CLI statement be set:</p> <ul style="list-style-type: none">from evpn-mac-route [mac-ipv4 mac-ipv6 mac-only] <p>Note that this match statement is valid only together with the family evpn match statement.</p>					
local-preference	<p>Set the local preference (LOCAL_PREF) attribute. The preference value can be a number in the range from 0 through 4,294,967,295.</p>					
mac-filter-list	<p>(BGP only) Named MAC filter list. EVPN Type 2 routes have MAC address as part of the prefix, which you can use to create a list of MAC addresses.</p>					

Table 10: List of Match Conditions for Filtering EVPN Routes (*Continued*)

Match Condition	Description
<code>metric</code>	Metric corresponds to the MED, and <code>metric2</code> corresponds to the IGP metric if the BGP next hop loops through another router. You can specify up to four metric values, <code>metric</code> , <code>metric2</code> , <code>metric3</code> , and <code>metric4</code> .
<code>next-hop (address / discard / next-table table-name / peer-address / reject / self)</code>	<p>Requires IBGP or EBGp confederations (third-party next hop must be advertised).</p> <ul style="list-style-type: none"> • <i>discard</i>—The next-hop address is replaced by a discard next hop. • <i>next-table</i>—The routing device performs a forwarding lookup in the specified table. • <i>self</i>—The next-hop address is replaced by one of the local routing device's addresses, as determined by the advertising protocol, typically the local IP address used for the BGP adjacency. • <i>specify peer-address</i>—The next-hop address is replaced by the peer's IP address (import only), typically an advertising routing device or another directly connected routing device.
<code>nlri-route-type</code>	<p>For EVPN, NLRI route types range from 1 to 8 (the first octet of the route prefix in the BGP update message is the EVPN route type).</p> <p>Multiple route types can be specified in a single policy.</p>
<code>origin</code>	<p>Set the BGP path origin attribute to one of the following values:</p> <ul style="list-style-type: none"> • <i>egp</i>—Path information originated in another AS. • <i>igp</i>—Path information originated within the local AS. • <i>incomplete</i>—Path information learned by some other means.

Table 10: List of Match Conditions for Filtering EVPN Routes (Continued)

Match Condition	Description
<pre>prefix-list-filter prefix-list- name match-type</pre>	<p>Both prefix-list and prefix-list-filter match conditions are supported. prefix-list is similar to prefix-list-filter, with the exception that a match-type can be specified only with prefix-list-filter. You can specify prefix length qualifiers for the list of prefixes in the prefix list.</p> <p>When used with EVPN NRLI route Types 2 and 5, the following are supported:</p> <ul style="list-style-type: none"> from prefix-list-filter [exact longer orlonger]
route-distinguisher	<p>Value of the route-distinguisher (RD).</p> <p>Filtering BGP EVPN routes based on RD is supported. The RD information is carried in the prefix of the EVPN route.</p>
<pre>route-filter route-filter-list</pre>	<p>Named route filter or route filter list. You can specify prefix length qualifiers for the list of routes in the route filter list.</p> <p>When used with EVPN NRLI route types 2 and 5, the following are supported:</p> <ul style="list-style-type: none"> from route-filter [address-mask exact longer orlonger prefix-length-range through upto]

When using policy filters to filter EVPN routes, in Junos OS Release 19.4R1 and later, the following policy actions are supported (that is, they can be specified as the **then** qualifier in the policy).

[Table 11 on page 130](#) lists actions that can be used when filtering EVPN routes.

Table 11: List of Actions for Filtering EVPN Routes

Action	Description
accept	Accept a route.
apply-groups <i>group-name</i>	Apply a configuration group to a policy. If you specify more than one group name, the first group listed takes priority over the next, and so on.

Table 11: List of Actions for Filtering EVPN Routes *(Continued)*

Action	Description
<code>apply-groups-except <i>group-name</i></code>	Disable inheritance of a configuration group. This action is useful when you use the <code>apply-group</code> statement in a policy but also want to override the values inherited from the configuration group for a specific parameter.
<code>as-path-prepend</code>	<p>Appends one or more AS numbers at the beginning of the AS path. If you are specifying more than one AS number, include the numbers in quotation marks.</p> <p>The AS numbers are added after the local AS number has been added to the path. This action adds AS numbers to AS sequences only, not to AS sets. If the existing AS path begins with a confederation sequence or set, the appended AS numbers are placed within a confederation sequence. Otherwise, the appended AS numbers are placed with a non-confederation sequence.</p>
<code>default-action</code>	Accept or Reject any action log protocol by overriding them. This is a non-terminating policy action.
<code>next</code>	Skip to next policy or term.
<code>preference</code>	Sets the BGP local preference attribute for the route. The preference can be a number from 0 through 4,294,967,295), with lower numbers being more preferred. Selected routes are installed into the forwarding table.
<code>priority</code>	Set the priority for route installation: high, low, or medium. High priority routes are updated first in the in the RIB (routing table) and the FIB (forwarding table), before medium and low priority routes. Routes are placed in different priority queues according to the priority.
<code>reject</code>	Rejects the route and does not propagate it. After a route is rejected, no other terms in the routing policy and no other routing policies are evaluated.
<code>tag (add subtract) tag2 (add subtract) <i>number</i></code>	Change the tag value by the specified amount.

RELATED DOCUMENTATION

[Policy Framework Overview](#)

[Example: Using Policy Filters to Filter EVPN Routes | 132](#)

export

import

vpn-apply-export

vrf-export

vrf-import

Example: Using Policy Filters to Filter EVPN Routes

IN THIS SECTION

- [Requirements | 132](#)
- [Overview | 133](#)
- [Base Configuration | 133](#)

In Junos, routing policies can be used to control Border Gateway Protocol (BGP) route advertisements and to filter routes using different address families. But, although Ethernet VPN (EVPN) uses BGP to exchange MAC-IP addresses between different PE routers, differences such as the EVPN route prefix format and extended community information that is encoded in the BGP update message, mean that special match conditions are needed to be able to filter EVPN routes.

The examples in this topic show the various router configurations available in Junos for filtering EVPN routes.

Requirements

EVPN route filtering is supported on MX, VMX, EX, ACX, and QFX devices running Junos Release 19.4R1 or later. It is available at the routing-instance level of the hierarchy (where it is configured with **vrf-export** or **vrf-import** policy), and at the protocols bgp level (in which case you also need to configure vpn-apply-export for the policy to take effect).

Overview

IN THIS SECTION

- [Topology | 133](#)

You can use policy filters to filter EVPN routes, for example to specify particular extended community attributes. Routes are filtered according to the match conditions you specify in the **from** qualifier of the policy. Supported match criteria for EVPN routes include EVPN NLRI type, BGP path attributes, route distinguishers, EVPN Ethernet Tag, Ethernet Segment Identifier (ESI), and MAC addresses in EVPN Type 2 routes.

The following route filters are also supported: local-preference, as-path, community, next-hop, metric, and origin.

Actions are taken according to the criteria you specify in the **then** qualifier specified in the policy.

See "[Routing policies for EVPN](#)" on [page 125](#) for a complete list and description of supported match conditions and actions.

Topology

The following network scenarios show the configuration used for setting up various EVPN match conditions.

Base Configuration

IN THIS SECTION

- [Verification | 156](#)

CLI Quick Configuration

For EVPN routes, a policy can be applied at the routing-instance level of the hierarchy, or at the protocols bgp level. The configuration for both is shown below. At the routing-instance level, the policy is applied as an vrf-export or vrf-import policy. When an *export* policy is applied at the BGP group level, you must configure vpn-apply-export for the policy to work properly.

Case 1 shows the mandatory use of the statement `vpn-apply-export` when a policy is applied at the BGP level of the hierarchy.

To use the example, you need to navigate to various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode*.

To quickly configure the examples, copy the list of 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.

Case 1: Applying the policy at the protocol BGP level of the hierarchy.

```
set protocols bgp group evpn-sessions type internal
set protocols bgp group evpn-sessions local-address 10.255.255.4
set protocols bgp group evpn-sessions import bgp-evpn-exp
set protocols bgp group evpn-sessions family evpn signaling
set protocols bgp group evpn-sessions neighbor 10.255.255.1
set protocols bgp group evpn-sessions neighbor 10.255.255.6
set protocols bgp group evpn-sessions neighbor 10.255.255.8
set protocols bgp group evpn-sessions vpn-apply-export
set policy-options policy-statement bgp-evpn-exp term 1 from family evpn
set policy-options policy-statement bgp-evpn-exp term 1 from nlri-route-type 2
set policy-options policy-statement bgp-evpn-exp term 1 from nlri-route-type 3
set policy-options policy-statement bgp-evpn-exp term 1 then community add COM5
set policy-options policy-statement bgp-evpn-exp term 1 then as-path-prepend 999
```

Case 2 shows the mandatory use of the statements `vrf-export` and `vrf-import` when match conditions are being applied at the routing instances level of the hierarchy.

EVPN uses 8 different route types to extend Layer 2 connectivity. The EVPN NLRI route type is defined in the first octet of the route prefix field in the BGP update message.



NOTE: In Junos, the following EVPN route types, **Type 1 AD per ESI**, **Type 4 ES**, **Type 7 IGMP join**, and **Type 8 IGMP leave**, routes are not specific to a given routing-instance. Instead, they are automatically added to the default routing-instance table when exported. As a result no routing-instance **vrf-export** or **vrf-import** policies are applied to these route types. If you want to apply an export policy to these routes, you need to do it at the BGP export level of the hierarchy. The same is true for importing Type 1 per ESI, Type 4, Type 7, and Type 8 routes (they are automatically imported into the default-routing instance table). So, to apply an import policy to these route types, you need to do so at the BGP import level of the hierarchy rather than at the routing-instance level.

Case 2: Applying the policy at the routing-instance level of the hierarchy.

```

set routing-instances evpa protocols evpn
set routing-instances evpa instance-type evpn
set routing-instances evpa vlan-id none
set routing-instances evpa routing-interface irb.600
set routing-instances evpa interface ge-0/0/1.600
set routing-instances evpa route-distinguisher 2:3
set routing-instances evpa vrf-export vrf-exp-pol
set routing-instances evpa vrf-target target:1:1
set policy-options policy-statement vrf-exp-pol term 1 from family evpn
set policy-options policy-statement vrf-exp-pol term 1 from nlri-route-type 1
set policy-options policy-statement vrf-exp-pol term 1 then community add COM11
set policy-options policy-statement vrf-exp-pol term 1 then accept

```

Filtering BGP EVPN routes based on EVPN NLRI type

CLI Quick Configuration

A complete list of set commands used in the example are presented first, followed by the same commands in step-by-step format, as well as instructions for confirming your configuration. Verification commands that you can use to see relevant output from a properly configured system are shown at the end of this topic.

Filtering BGP EVPN routes based on EVPN NLRI type

```

set policy-options policy-statement bgp-evpn-exp term 1 from family evpn
set policy-options policy-statement bgp-evpn-exp term 1 from nlri-route-type 2
set policy-options policy-statement bgp-evpn-exp term 1 from nlri-route-type 3
set policy-options policy-statement bgp-evpn-exp term 1 then community add COM5
set policy-options policy-statement bgp-evpn-exp term 1 then as-path-prepend 999
set protocols bgp group evpn-session type internal
set protocols bgp group evpn-session local-address 10.255.255.4
set protocols bgp group evpn-session family evpn signaling
set protocols bgp group evpn-session export bgp-evpn-exp
set protocols bgp group evpn-session neighbor 10.255.255.1
set protocols bgp group evpn-session neighbor 10.255.255.6
set protocols bgp group evpn-session neighbor 10.255.255.8
set protocols bgp group evpn-session vpn-apply-export

```


Step-by-Step Procedure

To set up the filtering of BGP EVPN routes based on BGP path attributes:

1. Configure the BGP path attributes you want to filter on (enclose multiple types in brackets and separate with a space) and the action to take on the matching routes.

```
[edit policy-options policy-statement bgp-evpn-exp]
user@PE1# set term 1 from family evpn
user@PE1# set term 1 from nlri-route-type [2 3]
user@PE1# set term 1 then community add COM5
user@PE1# set term 1 then as-path-prepend 999
```

2. Configure the BGP group protocol session.

```
[edit protocols bgp group evpn-session ]
user@PE1# set type internal
user@PE1# set local-address 10.255.255.4
user@PE1# set family evpn signaling
user@PE1# set import bgp-evpn-exp
user@PE1# set neighbor 10.255.255.1
user@PE1# set neighbor 10.255.255.6
user@PE1# set neighbor 10.255.255.8
user@PE1# set vpn-apply-export
```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the `show policy-options policy-statement bgp-evpn-exp`, and `show protocols bgp group evpn-sessions` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
  from {
    family evpn;
    nlri-route-type [ 2 3 ];
  }
  then {
    community add COM5;
```

```

        as-path-prepend 999;
    }
}

```

```

user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
    type internal;
    local-address 10.255.255.4;
    export bgp-evpn-exp;
    family evpn {
        signaling;
    }
    neighbor 10.255.255.1;
    neighbor 10.255.255.6;
    neighbor 10.255.255.8;
    vpn-apply-export;
}

```

Filtering BGP EVPN routes based on route distinguisher

CLI Quick Configuration

Route distinguisher (RD) information is encoded in the EVPN route prefix. This example shows how to filter EVPN routes on the basis of the route distinguisher.

A complete list of set commands used in the example are presented first, followed by the same commands in step-by-step format, as well as instructions for confirming your configuration. Verification commands that you can use to see relevant output from a properly configured system are shown at the end of this topic.

Filtering BGP EVPN routes based on route distinguisher

```

set policy-options policy-statement bgp-evpn-exp term 1 from family evpn
set policy-options policy-statement bgp-evpn-exp term 1 from route-distinguisher 100:200
set policy-options policy-statement bgp-evpn-exp term 1 then community add COM5
set policy-options policy-statement bgp-evpn-exp term 1 then as-path-prepend 999
set protocols bgp group evpn-session type internal
set protocols bgp group evpn-session local-address 10.255.255.4
set protocols bgp group evpn-session family evpn signaling
set protocols bgp group evpn-session export bgp-evpn-exp
set protocols bgp group evpn-session neighbor 10.255.255.1
set protocols bgp group evpn-session neighbor 10.255.255.6

```

```
set protocols bgp group evpn-session neighbor 10.255.255.8
set protocols bgp group evpn-session vpn-apply-export
```

Step-by-Step Procedure

To set up the filtering of BGP EVPN routes based on route distinguisher:

1. Configure the route distinguisher you want to filter on and the action to take on the matching routes.

```
[edit policy-options policy-statement bgp-evpn-exp]
user@PE1# set term 1 from family evpn
user@PE1# set term 1 from route-distinguisher 100:200
user@PE1# set term 1 then community add COM5
user@PE1# set term 1 then as-path-prepend 999
```

2. Configure the BGP group protocol session.

```
[edit protocols bgp group evpn-session ]
user@PE1# set type internal
user@PE1# set local-address 10.255.255.4
user@PE1# set family evpn signaling
user@PE1# set export bgp-evpn-exp
user@PE1# set neighbor 10.255.255.1
user@PE1# set neighbor 10.255.255.6
user@PE1# set neighbor 10.255.255.8
user@PE1# set vpn-apply-export
```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the `show policy-options policy-statement bgp-evpn-exp`, `show policy-options route-distinguisher RD1`, and `show protocols bgp group evpn-sessions` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
  from {
    family evpn;
    route-distinguisher 100:200;
```

```

    }
    then {
        community add COM5;
        as-path-prepend 999;
    }
}

```

```

user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
    type internal;
    local-address 10.255.255.4;
    import bgp-evpn-exp;
    family evpn {
        signaling;
        neighbor 10.255.255.1;
        neighbor 10.255.255.6;
        neighbor 10.255.255.8;
        vpn-apply-export;
    }
}

```

Filtering BGP EVPN routes based on EVPN Ethernet Tags

CLI Quick Configuration

EVPN Ethernet Tag information (or vlan-id information) is carried in the prefix of the EVPN route. This example shows how to filter EVPN routes based on the Ethernet Tag carried in the prefix of the route. Note that you must include the `family evpn` qualifier when configuring this filtering option.

A complete list of set commands used in the example are presented first, followed by the same commands in step-by-step format, as well as instructions for confirming your configuration. Verification commands that you can use to see relevant output from a properly configured system are shown at the end of this topic.

Filtering BGP EVPN routes based on EVPN Ethernet Tags

```

set policy-options policy-statement bgp-evpn-exp term 1 from family evpn
set policy-options policy-statement bgp-evpn-exp term 1 from evpn-tag [ 10 12 13 ]
set policy-options policy-statement bgp-evpn-exp term 1 then community add COM5
set policy-options policy-statement bgp-evpn-exp term 1 then as-path-prepend 999
set protocols bgp group evpn-session type internal
set protocols bgp group evpn-session local-address 10.255.255.4

```

```

set protocols bgp group evpn-session family evpn signaling
set protocols bgp group evpn-session export bgp-evpn-exp
set protocols bgp group evpn-session neighbor 10.255.255.1
set protocols bgp group evpn-session neighbor 10.255.255.6
set protocols bgp group evpn-session neighbor 10.255.255.8
set protocols bgp group evpn-session vpn-apply-export

```

Step-by-Step Procedure

To set up the filtering of BGP EVPN routes based on the EVPN Ethernet Tag:

1. Configure the EVPN Ethernet Tag you want to filter on and the action to take on the matching routes.

```

[edit policy-options policy-statement bgp-evpn-exp]
user@PE1# set term 1 from family evpn
user@PE1# set term 1 from evpn-tag [ 10 12 13 ]
user@PE1# set term 1 then community add COM5
user@PE1# set term 1 then as-path-prepend 999

```

2. Configure the BGP group protocol session.

```

[edit protocols bgp group evpn-session ]
user@PE1# set type internal
user@PE1# set local-address 10.255.255.4
user@PE1# set family evpn signaling
user@PE1# set import bgp-evpn-exp
user@PE1# set neighbor 10.255.255.1
user@PE1# set neighbor 10.255.255.6
user@PE1# set neighbor 10.255.255.8
user@PE1# set vpn-apply-export

```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the `show policy-options policy-statement bgp-evpn-exp`, and `show protocols bgp group`

evpn-sessions commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
    from {
        family evpn;
        evpn-tag [ 10 12 13 ];
    }
    then {
        community add COM5;
        as-path-prepend 999;
    }
}
```

```
user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
    type internal;
    local-address 10.255.255.4;
    import bgp-evpn-exp;
    family evpn {
        signaling;
    }
    neighbor 10.255.255.1;
    neighbor 10.255.255.6;
    neighbor 10.255.255.8;
    vpn-apply-export;
}
```

Filtering BGP EVPN routes based on ESI

CLI Quick Configuration

You can use Ethernet Segment Identifier (ESI) based policy filters for Type 1, Type 2, Type 4, Type 7, and Type 8 routes, which are the only types to contain ESI information in the prefix.

A complete list of set commands used in the example are presented first, followed by the same commands in step-by-step format, as well as instructions for confirming your configuration. Verification commands that you can use to see relevant output from a properly configured system are shown at the end of this topic.

Filtering BGP EVPN routes based on ESI

```

set policy-options policy-statement bgp-evpn-exp term 1 from family evpn
set policy-options policy-statement bgp-evpn-exp term 1 from evpn-esi
00:11:22:33:44:55:66:77:88:99
set policy-options policy-statement bgp-evpn-exp term 1 then community add COM1
set protocols bgp group evpn-session type internal
set protocols bgp group evpn-session local-address 10.255.255.8
set protocols bgp group evpn-session family evpn signaling
set protocols bgp group evpn-session export bgp-evpn-exp
set protocols bgp group evpn-session vpn-apply-export
set protocols bgp group evpn-session neighbor 10.255.255.1
set protocols bgp group evpn-session neighbor 10.255.255.4
set protocols bgp group evpn-session neighbor 10.255.255.6

```

Step-by-Step Procedure

To set up the filtering of BGP EVPN routes based on the ESI:

1. Configure the EVPN ESI you want to filter on and the action to take on the matching routes.

```

[edit policy-options policy-statement bgp-evpn-exp]
user@PE1# set term 1 from family evpn
user@PE1# set term 1 from evpn-esi 00:11:22:33:44:55:66:77:88:99
user@PE1# set term 1 then community add COM1

```

2. Configure the BGP group protocol session.

```

[edit protocols bgp group evpn-session ]
user@PE1# set type internal
user@PE1# set local-address 10.255.255.8
user@PE1# set family evpn signaling
user@PE1# set export bgp-evpn-exp
user@PE1# set vpn-apply-export
user@PE1# set neighbor 10.255.255.1
user@PE1# set neighbor 10.255.255.4
user@PE1# set neighbor 10.255.255.6

```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the `show policy-options policy-statement bgp-evpn-exp`, and `show protocols bgp group evpn-sessions` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
  from {
    family evpn;
    evpn-esi 00:11:22:33:44:55:66:77:88:99;
  }
  then {
    community add COM1;
  }
}
```

```
user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
  type internal;
  local-address 10.255.255.8;
  family evpn {
    signaling;
    export bgp-evpn-exp;
    vpn-apply-export;
    neighbor 10.255.255.1;
    neighbor 10.255.255.4;
    neighbor 10.255.255.6;
  }
}
```

Filtering BGP EVPN Type 2 and Type 5 routes based on IP address.

CLI Quick Configuration

You can use IPv4 or IPv6 addresses embedded in the EVPN prefix field to filter EVPN Type 2 and Type 5 routes. The following prefix-list and route-filter qualifiers are also supported:

- from prefix-list

- from prefix-list-filter [exact | longer | orlonger]
- from route-filter [address-mask | exact | longer | orlonger | prefix-length-range | through | upto]
- from route-filter-list

A complete list of set commands used in the example are presented first, followed by the same commands in step-by-step format, as well as instructions for confirming your configuration. Verification commands that you can use to see relevant output from a properly configured system are shown at the end of this topic.

Filtering BGP EVPN Type 2 and Type 5 routes based on the IP address

```
set policy-options prefix-list pp1 10.1.1.10/32
set policy-options prefix-list pp1 10.1.1.11/32
set policy-options policy-statement bgp-evpn-exp term 1 from family evpn
set policy-options policy-statement bgp-evpn-exp term 1 from prefix-list pp1
set policy-options policy-statement bgp-evpn-exp term 1 then community add COM1
set protocols bgp group evpn-session type internal
set protocols bgp group evpn-session local-address 10.255.255.8
set protocols bgp group evpn-session family evpn signaling
set protocols bgp group evpn-session export bgp-evpn-exp
set protocols bgp group evpn-session vpn-apply-export
set protocols bgp group evpn-session neighbor 10.255.255.1
set protocols bgp group evpn-session neighbor 10.255.255.4
set protocols bgp group evpn-session neighbor 10.255.255.6
```

Step-by-Step Procedure

To set up the filtering of BGP EVPN Type 2 and Type 5 routes based on the IP address:

1. Create a prefix list to be used in the policy statement.

```
[ edit policy-options prefix-list pp1]
user@PE1# set 10.1.1.10/32
user@PE1# set 10.1.1.11/32
```

2. Configure the Type 2 and Type 5 IP address you want to filter on and the action to take on the matching routes.

```
[edit policy-options policy-statement bgp-evpn-exp]
user@PE1# set term 1 from family evpn
user@PE1# set term 1 from prefix-list pp1
user@PE1# set term 1 then community add COM1
```

3. Configure the BGP group protocol session.

```
[edit protocols bgp group evpn-session ]
user@PE1# set type internal
user@PE1# set local-address 10.255.255.8
user@PE1# set family evpn signaling
user@PE1# set export bgp-evpn-exp
user@PE1# set vpn-apply-export
user@PE1# set neighbor 10.255.255.1
user@PE1# set neighbor 10.255.255.4
user@PE1# set neighbor 10.255.255.6
```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the `show policy-options policy-statement bgp-evpn-exp`, and `show protocols bgp group evpn-sessions` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options prefix-list pp1
10.1.1.10/32;
10.1.1.11/32;
```

```
user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
  from {
    family evpn;
    prefix-list pp1;
  }
  then {
```

```

        community add COM1;
    }
}

```

```

user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
    type internal;
    local-address 10.255.255.8;
    family evpn {
        signaling;
    }
    export bgp-evpn-exp;
    vpn-apply-export;
    neighbor 10.255.255.1;
    neighbor 10.255.255.4;
    neighbor 10.255.255.6;
}

```

Filtering BGP EVPN Type 2 routes using MAC address

CLI Quick Configuration

You can use the MAC address in EVPN prefix to filter EVPN Type 2 routes.

A complete list of set commands used in the example are presented first, followed by the same commands in step-by-step format, as well as instructions for confirming your configuration. Verification commands that you can use to see relevant output from a properly configured system are shown at the end of this topic.

Filtering BGP EVPN Type 2 routes using MAC address

```

set policy-options mac-list mfl1 01:87:88:04:50:00
set policy-options mac-list mfl1 02:87:88:04:50:00
set policy-options mac-list mfl1 03:87:88:04:50:00
set policy-options mac-list mfl1 04:87:88:04:50:00
set policy-options mac-list mfl1 05:87:88:04:50:00
set policy-options mac-list mfl1 06:87:88:04:50:00
set policy-options mac-list mfl1 07:87:88:04:50:00
set policy-options mac-list mfl1 08:87:88:04:50:00
set policy-options mac-list mfl1 64:87:88:04:50:00
set policy-options policy-statement bgp-evpn-exp term 1 from family evpn
set policy-options policy-statement bgp-evpn-exp term 1 from mac-filter-list mfl1

```

```

set policy-options policy-statement bgp-evpn-exp term 1 then accept
set protocols bgp group evpn-session type internal
set protocols bgp group evpn-session local-address 10.255.255.8
set protocols bgp group evpn-session family evpn signaling
set protocols bgp group evpn-session export bgp-evpn-exp
set protocols bgp group evpn-session vpn-apply-export
set protocols bgp group evpn-session neighbor 10.255.255.1
set protocols bgp group evpn-session neighbor 10.255.255.4
set protocols bgp group evpn-session neighbor 10.255.255.6

```

Step-by-Step Procedure

To set up the filtering of BGP EVPN Type 2 routes using MAC address:

1. Create the list of the MAC addresses you want to filter on (**mfl1** in this example).

```

[edit policy-options mac-list mfl1]
user@PE1# set 01:87:88:04:50:00;
user@PE1# set 02:87:88:04:50:00;
user@PE1# set 03:87:88:04:50:00;
user@PE1# set 04:87:88:04:50:00;
user@PE1# set 05:87:88:04:50:00;
user@PE1# set 06:87:88:04:50:00;
user@PE1# set 07:87:88:04:50:00;
user@PE1# set 08:87:88:04:50:00;

```

2. Apply a list of the MAC addresses you want to filter on, and the action you want to take (**Accept**, in this example).

```

[edit policy-options policy-statement bgp-evpn-exp]
user@PE1# set term 1 from family evpn
user@PE1# set term 1 from mac-filter-list mfl1
user@PE1# set term 1 then accept

```

3. Configure the BGP group protocol session.

```

[edit protocols bgp group evpn-session ]
user@PE1# set type internal
user@PE1# set local-address 10.255.255.8
user@PE1# set family evpn signaling

```

```

user@PE1# set export bgp-evpn-exp
user@PE1# set neighbor 10.255.255.1
user@PE1# set neighbor 10.255.255.4
user@PE1# set neighbor 10.255.255.6
user@PE1# set vpn-apply-export

```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the `show policy-options mac-list mfl1`, `show policy-options policy-statement bgp-evpn-exp`, and `show protocols bgp group evpn-sessions` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```

user@PE1# show policy-options mac-list mfl1
01:87:88:04:50:00;
02:87:88:04:50:00;
03:87:88:04:50:00;
04:87:88:04:50:00;
05:87:88:04:50:00;
06:87:88:04:50:00;
07:87:88:04:50:00;
08:87:88:04:50:00;

```

```

user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
    from {
        family evpn;
        mac-filter-list mfl1;
    }
    then {
        accept;
    }
}

```

```

user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
    type internal;
    local-address 10.255.255.8;
}

```

```

family evpn {
    signaling;
    export bgp-evpn-exp;
    vpn-apply-export;
    neighbor 10.255.255.1;
    neighbor 10.255.255.4;
    neighbor 10.255.255.6;
}

```

Filtering BGP EVPN Type 2 routes that contain (or do not contain) an IP address

CLI Quick Configuration

EVPN Type 2 routes have a MAC address and can additionally have an IP address (IPv4 or IPv6) in the prefix. With BGP EVPN Type 2 filters, you can filter Type 2 routes based according to whether it has only a MAC address, a MAC address and IPv4 address, or a MAC address and IPv6 address (not a specific IP address, but any IP address in the prefix). These options are mutually exclusive.

A complete list of set commands used in the example are presented first, followed by the same commands in step-by-step format, as well as instructions for confirming your configuration. Verification commands that you can use to see relevant output from a properly configured system are shown at the end of this topic.

Filtering BGP EVPN Type 2 routes with MAC address only

```

set policy-options policy-statement bgp-evpn-exp term 1 from family evpn
set policy-options policy-statement bgp-evpn-exp term 1 from evpn-mac-route mac-only
set policy-options policy-statement bgp-evpn-exp term 1 then community add COM1
set protocols bgp group evpn-session type internal
set protocols bgp group evpn-session local-address 10.255.255.8
set protocols bgp group evpn-session family evpn signaling
set protocols bgp group evpn-session export bgp-evpn-exp
set protocols bgp group evpn-session vpn-apply-export
set protocols bgp group evpn-session neighbor 10.255.255.1
set protocols bgp group evpn-session neighbor 10.255.255.4
set protocols bgp group evpn-session neighbor 10.255.255.6

```

Step-by-Step Procedure

To set up the filtering of BGP EVPN Type 2 routes with MAC address only:

1. Create a policy and the action you want to take.

```
[edit policy-options policy-statement bgp-evpn-exp]
user@PE1# set term 1 from family evpn
user@PE1# set term 1 from evpn-mac-route mac-only
user@PE1# set term 1 then community add COM1
```

2. Configure the BGP group protocol session (we use export bgp-evpn-exp here to apply the policy).

```
[edit protocols bgp group evpn-session ]
user@PE1# set type internal
user@PE1# set local-address 10.255.255.8
user@PE1# set family evpn signaling
user@PE1# set export bgp-evpn-exp
user@PE1# set neighbor 10.255.255.1
user@PE1# set neighbor 10.255.255.4
user@PE1# set neighbor 10.255.255.6
```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the, show policy-options policy-statement bgp-evpn-exp, and show protocols bgp group evpn-sessions commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
  from {
    family evpn;
    evpn-mac-route mac-only;
  }
  then {
    community add COM1;
  }
}
```

```
user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
```

```

type internal;
local-address 10.255.255.8;
family evpn {
    signaling;
    export bgp-evpn-exp;
    vpn-apply-export;
    neighbor 10.255.255.1;
    neighbor 10.255.255.4;
    neighbor 10.255.255.6;
}

```

Filtering BGP EVPN routes according to an EVPN extended community

CLI Quick Configuration

BGP EVPN routes can have a set of extended communities carried in the BGP update message path attribute, and as such, you can use these extended communities for filtering BGP EVPN routes. . The EVPN specific information included in the extended communities includes encapsulation type, MAC-mobility information, EVPN split-horizon label,, ESI mode, E-Tree leaf label, and more.

See [Border Gateway Protocol \(BGP\) Extended Communities](#) for the full list of extended communities.

An extended community is an eight-octet value divided into two main sections, and typically uses a notation of **type:administrator:assigned-number**. However, to specify EVPN extended communities in the Junos configuration for BGP EVPN, instead of using a word to specify the type, all values (including *type*) are in decimal. Type is 2 octet, with the higher-order octet defining the actual type of extended community, and the low-order octet defining the community. The sub-type; val1 and val2 can be specified as [2 + 4] octets, or as [4 + 2] octets.

Typical configuration for extended communities in Junos:

- set policy-options community *name* members type:val1:val2

Specifying an extended community numerically for BGP EVPN configurations in Junos. See [BGP MPLS-Based Ethernet VPN](#) for more information on numerical representations of extended communities.

In the example below, the decimal 780 is used to match the encapsulation extended community (for example, VXLAN). For 780, the value of the high-order octet of the extended type field is 0x03, which indicates that it is transitive. The value of the low-order octet of the extended type field is 0x0c; thus, the first 2 octet value is 0x030c, which is where the decimal 780 comes from. The remaining value fields, where *val1* is 0 and *val2* is 8, are used to identify VXLAN tunnel type.

The full list of tunnel types related to EVPN is defined in RFC 8365, Section 11 (link below), but some pertinent ones are listed here:

- Value 8 = VXLAN Encapsulation
- Value 9 = NVGRE Encapsulation
- Value 10 = MPLS Encapsulation
- Value 11 = MPLS in GRE Encapsulation
- Value 12 = VXLAN GPE Encapsulation

See [RFC 5512, Section 4.5, Reserved field](#) and [RFC 8365, Section 11](#) for details.

- `set policy-options community name members 780:0:8`

A complete list of set commands used in the example are presented first, followed by the same commands in step-by-step format, as well as instructions for confirming your configuration. Verification commands that you can use to see relevant output from a properly configured system are shown at the end of this topic.

Filtering BGP EVPN routes according to the EVPN extended communities

```
set policy-options community COM5 members 780:0:8
set policy-options policy-statement bgp-evpn-exp term 1 from community COM5
set policy-options policy-statement bgp-evpn-exp term 1 then reject
set protocols bgp group evpn-session type internal
set protocols bgp group evpn-session local-address 10.255.255.4
set protocols bgp group evpn-session family evpn signaling
set protocols bgp group evpn-session export bgp-evpn-exp
set protocols bgp group evpn-session neighbor 10.255.255.1
set protocols bgp group evpn-session neighbor 10.255.255.6
set protocols bgp group evpn-session neighbor 10.255.255.8
```

Step-by-Step Procedure

To set up the filtering of BGP EVPN routes according to an EVPN extended community:

1. Create a list of the community members you want to filter on, and the action you want to take.

```
[edit policy-options]
user@PE1# set community COM5 members 780:0:8
```

2. Create a list of the community members you want to filter on, and the action you want to take.

```
[edit policy-options policy-statement bgp-evpn-exp]
user@PE1# set term 1 from community COM5
user@PE1# set term 1 then reject
```

3. Configure the BGP group protocol session (we use export bgp-evpn-exp here to apply the policy).

```
[edit protocols bgp group evpn-session ]
user@PE1# set type internal
user@PE1# set local-address 10.255.255.4
user@PE1# set family evpn signaling
user@PE1# set export bgp-evpn-exp
user@PE1# set neighbor 10.255.255.1
user@PE1# set neighbor 10.255.255.6
user@PE1# set neighbor 10.255.255.8
```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the, show policy-options policy-statement bgp-evpn-exp, and show protocols bgp group evpn-sessions commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options community COM5 members
members 780:0:8;
```

```
user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
  from {
    community COM5;
  }
  then {
    reject;
```

```
}
}
```

```
user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
    type internal;
    local-address 10.255.255.4;
    family evpn {
        signaling;
    }
    export bgp-evpn-exp;
    vpn-apply-export;
    neighbor 10.255.255.1;
    neighbor 10.255.255.6;
    neighbor 10.255.255.8;
}
```

Copying community information from EVPN Type 2 routes into EVPN Type 5 routes

You can use BGP EVPN filtering to include the MAC address (if any) and IPv4 or IPv6 addresses from EVPN type 2 route advertisements received from remote PEs as EVPN Type 5 routes. Likewise, you can copy the community information from EVPN Type 2 routes into EVPN Type 5 route that have been generated from routes in the **vrf.inet table**(specifically, VPN-IPv4 (AFI/SAFI 1/128), VPN-IPv6 (AFI/SAFI 2/128), IPv4 (AFI/SAFI 1/1) and IPv6 (AFI/SAFI 2/1)).

To include any contained MAC address and IPv4 or IPv6 addresses from EVPN Type 2 route advertisements into EVPN Type 5, enable the following command:

- `set routing-instances evpna protocols evpn remote-ip-host-routes no-advertise-community`

You can also control which routing attributes are carried between the IP and EVPN routes. In other words, you can choose which route attributes to include from the import direction when generating IP routes from EVPN Type 5 routes, and for the export direction, also choose which route attributes to include when generating EVPN Type 5 routes from IP routes. These route attributes are, as-path, community, and preference. Note that if you do not explicitly include the community route attribute during import, due to how Junos handles route attributes in the **vrf.inet.0** table, color community information will not be included (and thus this information not available for the nexthop resolution of the affected routes).

To include a given route attribute, use the following commands, and set an import or export action, which can be either allow or skip (here, the import-action is allow):

```
set routing-instances evpna protocols evpn ip-prefix-routes route-attributes as-path import-
action allow
set routing-instances evpna protocols evpn ip-prefix-routes route-attributes preference import-
action allow
set routing-instances evpna protocols evpn ip-prefix-routes route-attributes community import-
action allow
```

Results

To see your configuration results, from configuration mode at the top of the CLI hierarchy, confirm your configuration by entering the, `show policy-options policy-statement bgp-evpn-exp`, and `show protocols bgp group evpn-sessions` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options policy-statement bgp-evpn-exp
term 1 {
  from {
    family evpn;
    evpn-mac-route mac-only;
  }
  then {
    community add COM1;
  }
}
```

```
user@PE1# show protocols bgp group evpn-sessions
group evpn-sessions {
  type internal;
  local-address 10.255.255.8;
  family evpn {
    signaling;
  }
  export bgp-evpn-exp;
  vpn-apply-export;
  neighbor 10.255.255.1;
  neighbor 10.255.255.4;
```

```
neighbor 10.255.255.6;
}
```

Verification

IN THIS SECTION

- [Verifying the various BGP EVPN filtering | 156](#)

Confirm that the configuration is working properly. For each of the examples given above, run a version of these commands that uses the configuration you want to confirm. The verification example below is based on the example given for filtering BGP EVPN routes based on the EVPN NLRI type.

Verifying the various BGP EVPN filtering

Purpose

Display information about the BGP EVPN routes filtered according to the specified criteria.

Action

From operational mode on the target device, enter following commands:

```
user@device> show evpn instance
user@device> show evpn instance extensive
user@device> show evpn database
user@device> show evpn mac-ip-table
```

From operational mode on PE1, enter following commands:

```
user@PE1> show route table bgp.evpn.0
user@PE1> show route table EVPN-1.evpn.0
user@PE1> show route table default_evpn___.evpn.0
user@PE1> show route advertising-protocol bgp
100.100.100.2 table bgp.evpn.0
user@PE1> show route advertising-protocol bgp
100.100.100.2 table EVPN-1.evpn.0
```

```
user@PE1> show route advertising-protocol bgp
100.100.100.2 table default_evpn__.evpn.0
```

From operational mode on PE2, enter following commands:

```
user@PE2> show route receive-protocol bgp 100.100.100.1
table bgp.evpn.0
user@PE2> show route receive-protocol bgp 100.100.100.1
table EVPN-1.evpn.0
user@PE2> show route receive-protocol bgp 100.100.100.1
table default_evpn__.evpn.0
user@PE2> show route table bgp.evpn.0
user@PE2> show route table EVPN-1.evpn.0
user@PE2> show route table default_evpn__.evpn.0
```

RELATED DOCUMENTATION

[Routing policies for EVPN | 125](#)

[How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions](#)

export

import

vpn-apply-export

vrf-export

vrf-import

Layer 3 Gateways with Integrated Routing and Bridging for EVPN Overlays

IN THIS CHAPTER

- [Understanding the MAC Addresses For a Default Virtual Gateway in an EVPN-VXLAN or EVPN-MPLS Overlay Network | 158](#)

Understanding the MAC Addresses For a Default Virtual Gateway in an EVPN-VXLAN or EVPN-MPLS Overlay Network

In an Ethernet VPN (EVPN) centrally-routed bridging overlay, a device can function as a Layer 3 gateway on which you can configure integrated routing and bridging (IRB) interfaces. When you configure an IRB interface with a virtual gateway address (VGA), the device creates a default Layer 3 virtual gateway with the specified IP address. Through its IRB interface, the default virtual gateway enables the communication between non-virtualized hosts, virtual machines (VMs), and servers in different VXLANs, MPLS networks, or IP subnetworks.

When you configure a VGA for an IRB interface, the Layer 3 gateway automatically generates IPv4 media access control (MAC) address 00:00:5E:00:01:01 or IPv6 MAC address 00:00:5E:00:02:01 for that particular virtual gateway. In this topic, we refer to the virtual gateway MAC address as a *virtual MAC*. We refer to the MAC address for the IRB interface as the *IRB MAC*.

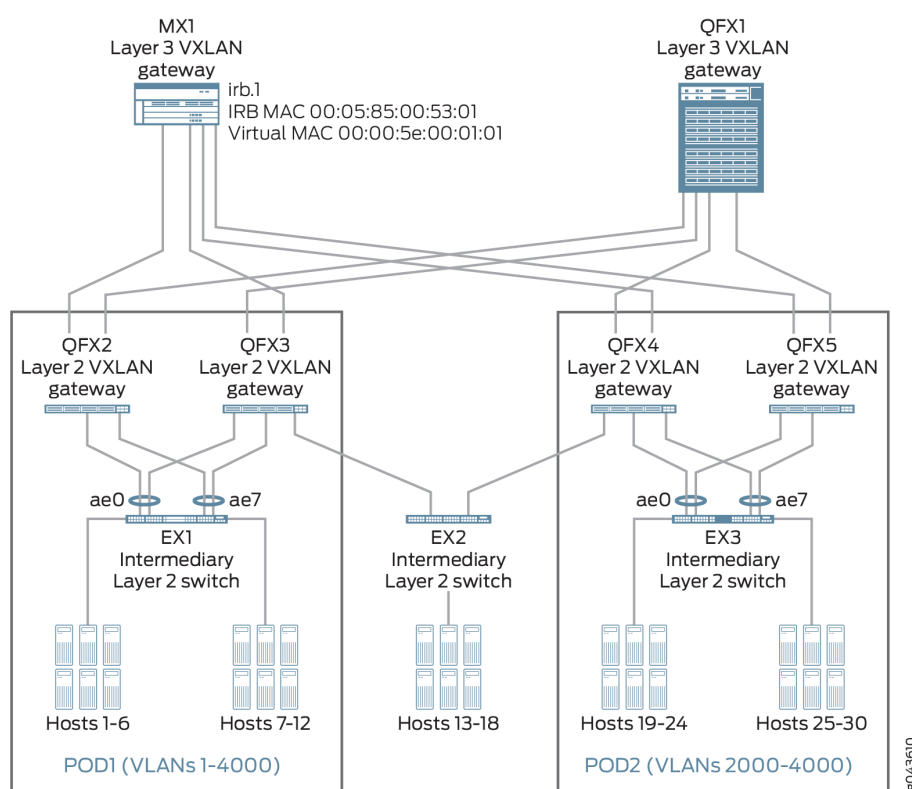
The Layer 3 gateway doesn't include the automatically generated virtual MAC as the source MAC address in the packets it generates. Instead, the device includes the IRB MAC in:

- Data packets
- The source MAC address field in the outer Ethernet header of:
 - Address Resolution Protocol (ARP) replies
 - Neighbor advertisement packets

When an ARP reply includes the IRB MAC as the source MAC address instead of the virtual MAC, in centrally-routed bridging (CRB) overlays you might see unknown unicast packet flooding throughout the domain.

For example, consider the EVPN-VXLAN overlay network in [Figure 11 on page 159](#). In this network, an MX Series router and a QFX10000 switch function as Layer 3 VXLAN gateways, and four QFX5100 switches function as Layer 2 VXLAN gateways. The overlay network also includes three intermediary Layer 2 switches, in this case, EX4300 switches, with connected hosts.

Figure 11: EVPN-VXLAN Centrally-Routed Bridging Overlay



On the MX Series router, an IRB interface named `irb.1` has MAC address `00:05:85:00:53:01` and VFA `10.2.1.254`. The MX Series router automatically generates the MAC address `00:00:5e:00:01:01` for the default virtual gateway.

In this overlay network, `irb.1` on the MX Series router receives an ARP request from host 1. In its ARP reply, the MX Series router includes the following:

- Source MAC address in outer Ethernet header: `00:05:85:00:53:01` (IRB MAC) → intermediary Layer 2 switch EX1 learns this MAC address.

- Sender MAC address within ARP reply packet: 00:00:5e:00:01:01 (virtual MAC) → intermediary Layer 2 switch EX1 cannot see this MAC address, and therefore, does not learn it.

When intermediary Layer 2 switch EX1 receives the ARP reply, it learns only the source MAC address (IRB MAC). As a result, if Host 1 sends packets that include the virtual MAC in the header, EX1 is unable to find the virtual MAC in its MAC table. Therefore, EX1 floods the domain with unknown unicast packets.



NOTE: Unknown unicast packet flooding isn't an issue in EVPN edge-routed bridging (ERB) overlays, where a single layer of QFX10000 switches function as both Layer 3 and Layer 2 gateways. In the ERB overlay, hosts are directly connected to the Layer 3 and Layer 2 gateways. Also, each IRB interface is typically configured with an IP address and a static MAC address. You repeat each IRB interface configuration on each gateway in the edge-routed bridging overlay. With the same MAC address configured for each IRB interface on each gateway, each host uses the same MAC address when sending inter-subnet traffic regardless of where the host is located or which gateway receives the traffic. As a result, you don't need to configure a default virtual gateway. For more information about ERB overlays, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway"](#) on page 845.

You can explicitly configure an IPv4 or IPv6 MAC address for a default virtual gateway in EVPN-VXLAN networks. Starting with Junos OS Release 22.1R1 on MX Series routers, you can similarly configure a default virtual gateway IPv4 or IPv6 address in an EVPN-MPLS network. Use the `virtual-gateway-v4-mac` or `virtual-gateway-v6-mac` configuration statement at the `[edit interfaces name irb unit logical-unit-number]` hierarchy level.

When you configure these statements, the configured virtual MAC overrides the automatically generated virtual MAC. For example, refer again to [Figure 11 on page 159](#). When the Layer 3 gateway MX1 sends data packets, ARP replies, and neighbor advertisement packets, it uses the configured virtual MAC in the outer Ethernet header of these packets. As a result, the intermediary Layer 2 switch EX1 also learns the configured virtual MAC, which eliminates the possibility that the switch floods the domain with unknown unicast packets.



NOTE: The MAC address range 02:00:00:00:00:00 through 02:00:00:00:00:FF is used for internal communication. Don't use addresses in this range if you explicitly configure a virtual MAC address.

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
22.1R1	Starting with Junos OS Release 22.1R1 on MX Series routers, you can explicitly configure an IPv4 or IPv6 MAC address for a default virtual gateway in an EVPN-MPLS network with a CRB overlay.
14.2R5	Starting with Junos OS Release 14.2R5 for MX Series routers and Junos OS Release 15.1X53-D63 for QFX10000 switches, you can explicitly configure an IPv4 or IPv6 MAC address for a default virtual gateway in an EVPN-VXLAN network . Use the <code>virtual-gateway-v4-mac</code> or <code>virtual-gateway-v6-mac</code> configuration statement at the <code>[edit interfaces <i>name</i> irb unit <i>logical-unit-number</i>]</code> hierarchy level.

RELATED DOCUMENTATION

[Anycast Gateways](#) | 1511

[Using a Default Layer 3 Gateway to Route Traffic in an EVPN-VXLAN Overlay Network](#) | 741

EVPN Multihoming

IN THIS CHAPTER

- [EVPN Multihoming Overview | 162](#)
- [EVPN Multihoming Designated Forwarder Election | 186](#)
- [Understanding Automatically Generated ESIs in EVPN Networks | 200](#)
- [Easy EVPN LAG \(EZ-LAG\) Configuration | 215](#)
- [Configuring EVPN Active-Standby Multihoming to a Single PE Device | 277](#)
- [Configuring EVPN-MPLS Active-Standby Multihoming | 280](#)
- [Example: Configuring Basic EVPN-MPLS Active-Standby Multihoming | 284](#)
- [Example: Configuring EVPN-MPLS Active-Standby Multihoming | 308](#)
- [Example: Configuring EVPN-MPLS Active-Standby Multihoming with ESI | 355](#)
- [Example: Configuring Basic EVPN Active-Active Multihoming | 362](#)
- [Example: Configuring EVPN Active-Active Multihoming | 375](#)
- [Example: Configuring LACP for EVPN Active-Active Multihoming | 442](#)
- [Example: Configuring LACP for EVPN VXLAN Active-Active Multihoming | 464](#)
- [Example: Configuring an ESI on a Logical Interface With EVPN-MPLS Multihoming | 493](#)
- [Configuring Dynamic List Next Hop | 510](#)

EVPN Multihoming Overview

IN THIS SECTION

- [Introduction to EVPN Multihoming | 163](#)
- [EVPN MPLS Multihoming Features Supported by QFX10000 Switches | 164](#)
- [EVPN MPLS Multihoming on ACX5448 Routers | 165](#)
- [Understanding EVPN Multihoming Concepts | 165](#)

- [EVPN Multihoming Mode of Operation | 167](#)
- [EVPN Multihoming Implementation | 169](#)
- [Designated Forwarder Election | 182](#)
- [ESIs on Physical, Aggregated Ethernet, and Logical Interfaces | 183](#)
- [Automatically Generated ESIs | 183](#)
- [Convergence in an EVPN Network | 183](#)

Introduction to EVPN Multihoming

An Ethernet VPN (EVPN) comprises of customer edge (CE) devices that are connected to provider edge (PE) devices, which form the edge of the MPLS infrastructure. A CE device can be a host, a router, or a switch. The PE devices provide Layer 2 virtual bridge connectivity between the CE devices. There can be multiple EVPNs in the provider network. Learning between the PE routers occurs in the control plane using BGP, unlike traditional bridging, where learning occurs in the data plane.



NOTE: In releases earlier than Junos OS Release 15.1, EVPN functionality support on MX Series routers was limited to routers using MPC and MIC interfaces only. In more recent Junos OS releases, MX Series routers using DPCs can be leveraged to provide EVPN support on the CE device-facing interface.

DPC support for EVPN is provided with the following considerations:

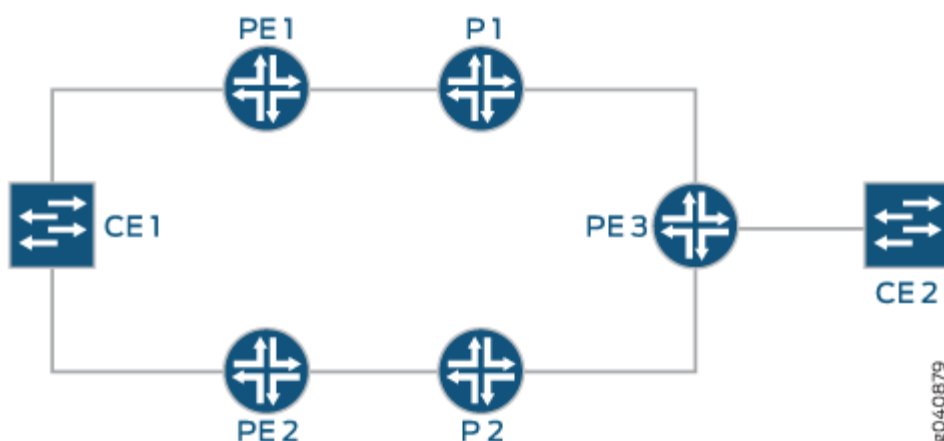
- DPCs provide support for EVPN in the active-standby mode of operation including support for the following:
 - EVPN instance (EVI)
 - Virtual switch
 - Integrated routing and bridging (IRB) interfaces
- DPCs intended for providing the EVPN active-standby support must be the CE device-facing line card. The PE device in the EVPN domain must be MPC interfaces or MIC interfaces.

The EVPN multihoming feature enables you to connect a customer site to two or more PE devices to provide redundant connectivity. A CE device can be multihomed to different PE devices or the same PE device. A redundant PE device can provide network service to the customer site as soon as a failure is detected. Thus, EVPN multihoming helps to maintain EVPN service and traffic forwarding to and from the multihomed site in the event of the following types of network failures:

- PE device to CE device link failure
- PE device failure
- MPLS-reachability failure between the local PE device and a remote PE device

Figure 12 on page 164 illustrates how a CE device can multihomed to two PE routers. Device CE 1 is multihomed to Routers PE 1 and PE 2. Device CE 2 has two potential paths to reach Device CE 1, and depending on the multihoming mode of redundancy, only one path or both the paths are active at any time. The multihoming mode of operation also determines which PE router or routers forward traffic to the CE device. The PE router forwarding traffic to the CE device (also called a designated forwarder) uses MPLS LSP or GRE tunnels to forward traffic. If a failure occurs over this path, a new designated forwarder is elected to forward the traffic to Device CE 1.

Figure 12: CE Device Multihomed to Two PE Routers



EVPN MPLS Multihoming Features Supported by QFX10000 Switches

Starting in Junos OS 17.4R1, QFX10000 switches support multihoming for EVPN MPLS. Only active-active multihoming is supported. The following subfeatures are supported:

- ESI configuration (only type 0 manual configuration and IFD (physical interfaces) are supported)
- Aliasing and label route
- EVPN Type 4 route (Ethernet segment route)
- Extended communities
- BUM traffic

- Designated Forwarder Election (DF) roles: DF and BDF

QFX10000 switches over an MPLS EVPN core only support the default-switch routing instance. An EVPN instance (EVI) is not supported.

EVPN MPLS Multihoming on ACX5448 Routers

Starting in Junos OS Release 19.4R1, ACX5448 routers support multihoming for EVPN MPLS. Only active-active multihoming is supported. To enable EVPN active-active multihoming on ACX5448 router, include the `evpn-mh-profile` configuration statement at the `[edit system packet-forwarding-options firewall-profile]` hierarchy level.

```
user@host# set system packet-forwarding-options firewall-profile ?
Possible completions:
  default-profile      Set the profile to support default services.
  evpn-mh-profile      Set the profile to support evpn-mh
```



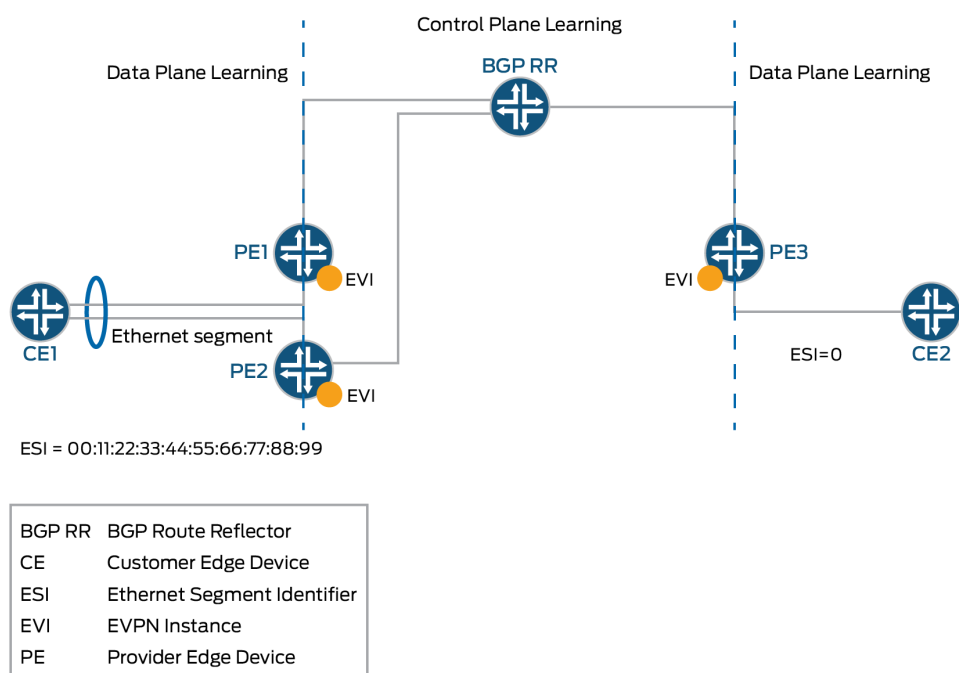
NOTE: After changing the profile and committing it, you need to restart the chassis management process by issuing the `restart chassis-control` CLI command to bring up the new profile.

A syslog warning appears to restart the PFE.

Understanding EVPN Multihoming Concepts

[Figure 13 on page 166](#) shows a simple EVPN network topology to define EVPN multihoming concepts.

Figure 13: Simple EVPN Topology



- **Ethernet segment**—When a CE device is multihomed to two or more PE routers, the set of Ethernet links constitutes an Ethernet segment. An Ethernet segment appears as a link aggregation group (LAG) to the CE device.

The links from Routers PE1 and PE2 to Device CE1 form an Ethernet segment.

In active-standby multihoming, the links that constitute an Ethernet segment form a bridge domain. In active-active multihoming, an Ethernet segment appears as a LAG to the CE device.

- **ESI**—An Ethernet segment must have a unique nonzero identifier, called the Ethernet segment identifier (ESI). The ESI is encoded as a 10-octet integer. When manually configuring an ESI value, the most significant octet, known as the type byte, must be 00. When a single-homed CE device is attached to an Ethernet segment, the entire ESI value is zero.

The Ethernet segment of the multihomed Device CE1 has an ESI value of 00:11:22:33:44:55:66:77:88:99 assigned. The single-homed Device CE2 has an ESI value of 0.

- **EVI**—An EVPN instance (EVI) is an EVPN routing and forwarding instance spanning all the PE routers participating in that VPN. An EVI is configured on the PE routers on a per-customer basis. Each EVI has a unique route distinguisher and one or more route targets.

An EVI is configured on Routers PE1, PE2, and PE3.

- **Ethernet tag**—An Ethernet tag identifies a particular broadcast domain, such as a VLAN. An EVPN instance consists of one or more broadcast domains. Ethernet tags are assigned to the broadcast domains of a given EVPN instance by the provider of that EVPN. Each PE router in that EVPN instance performs a mapping between broadcast domain identifiers understood by each of its attached CE devices and the corresponding Ethernet tag.
- **Ethernet segment route (EVPN Type 4 route)**—The PE routers that are connected to a multihomed CE device use BGP Ethernet segment route messages to discover that each of the PE routers is connected to the same Ethernet segment. The PE routers advertise the Ethernet segment route, which consists of an ESI and ES-import extended community.

Routers PE1 and PE2 advertise an ES route with an ES-import extended community (along with other extended communities like the route target). The PE routers also construct a filter that is based on an ES-import extended community, which results in only these PE routers importing the ES route and identifying that they are connected to the same Ethernet segment.

- **Extended community**— An extended community is similar in most ways to a regular community. EVPNs use extended communities because the 4-octet regular community value does not provide enough expansion and flexibility. An extended community is an 8-octet value divided into two main sections.
- **BUM traffic**—This type of traffic is sent to multiple destinations, including broadcast traffic, unknown unicast traffic that is broadcast in the Ethernet segment, and multicast traffic.
- **DF**—When a CE device is multihomed to two or more PE routers, either one or all of the multihomed PE routers are used to reach the customer site depending on the multihoming mode of operation. The PE router that assumes the primary role for forwarding BUM traffic to the CE device is called the designated forwarder (DF).
- **BDF**—Each router in the set of other PE routers advertising the autodiscovery route per Ethernet segment for the same ESI, and serving as the backup path in case the DF encounters a failure, is called a backup designated forwarder (BDF). A BDF is also called a non-DF router.
- **DF election**—On every Ethernet segment, the PE routers participate in a procedure called *designated forwarder* election to select the DF and the BDF PE routers.

EVPN Multihoming Mode of Operation

The different modes of operation for EVPN multihoming include:

- **Single**—When a PE router is connected to a single-homed customer site, this mode is in operation. The *single* mode is the default mode of operation, and does not require Ethernet segment values to be configured.

- **Active-standby**—When only a single PE router, among a group of PE routers attached to an Ethernet segment, is allowed to forward traffic to and from that Ethernet segment, the Ethernet segment is defined to be operating in the *active-standby* redundancy mode.

To configure the active-standby mode, include the ESI value and the `single-active` statement under the `[edit interfaces]` hierarchy level.



NOTE: We don't support active-standby multihoming mode on QFX Series switches or in EVPN configurations with VXLAN overlays. As a result, if you configure the `single-active` option on QFX Series switches or in EVPN-VXLAN configurations, the device ignores that configuration item.

- **Active-active**—When all PE routers attached to an Ethernet segment are allowed to forward traffic to and from the Ethernet segment, the Ethernet segment is defined to be operating in the *active-active* redundancy mode.



NOTE: In Junos OS Release 14.2 and earlier, the EX9200 Series switch supports only the active-standby mode of operation for EVPN multihoming.



NOTE: Starting with Junos OS Release 14.1x53-D30 for QFX5100 switches and Junos OS Release 18.2R1 for EX4600 switches, these switches support the active-active mode of operation for EVPN multihoming. In this scenario, QFX5100 and EX4600 switches function as top-of-rack (ToR) switches in the data center for virtual networks. EVPN multihoming active-active functionality is used to provide access to the bare-metal servers connected to the top-of-rack switches.



NOTE: Starting with Junos OS Release 14.1R4, 14.2, 15.1F6, and 16.1R1, Junos OS supports the active-active mode for EVPN multihoming on MX Series routers. Starting with Junos OS Releases 16.1R4 and 16.2R2, all EX9200 switches support the active-active mode for EVPN multihoming.

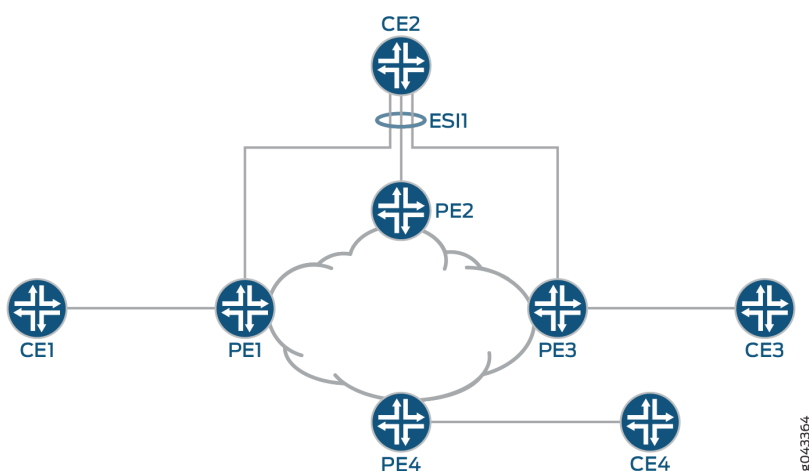
Starting with Junos OS Releases 17.4R1 QFX10000 switches support the active-active mode for EVPN multihoming.

To configure the active-active mode, include the ESI value and the `all-active` statement at the `[edit interfaces]` hierarchy level.

Figure 14 on page 169 shows a reference topology for EVPN active-active multihoming. The ESI1 Ethernet segment for Device CE2 is multihomed to Routers PE1, PE2, and PE3. The Ethernet

segment on the CE device can either be configured as a link aggregation group (LAG) or as an ECMP path. Devices CE1 and CE3 are the single-homed customer edge devices and have an ESI value of 0.

Figure 14: Active-Active EVPN Multihoming



EVPN Multihoming Implementation

The EVPN active-standby multihoming mode of operation provides redundancy for access link failures and PE node failure for the multihomed CE device, and is based on the EVPN *draft-ietf-l2vpn-evpn-03*.

The Junos OS implementation of the EVPN multihoming active-standby and active-active modes of operation includes the following:

New BGP NLRI

To support EVPN multihoming, the following new BGP network layer reachability information (NLRI) routes have been introduced:

Autodiscovery Route per Ethernet Segment

Autodiscovery Route Features

The autodiscovery route NLRI features include:

- This is a Type 1 mandatory route, used for fast convergence and for advertising the split horizon label. It is also known as the mass withdraw route.

- Type 1 route distinguishers are used with the IP address (loopback) of the originating PE router as the route distinguisher value.
- This route carries the ESI in the NLRI (nonzero when it is a multihomed PE, zero otherwise).
- The split horizon label is per ESI only, and carries an explicit NULL (0).
- The bit in the active-standby flag field in the ESI label extended community is used for signaling the active-standby mode (bit set).
- The 3-byte label values in the NLRI and the Ethernet tag is zero.
- This route is advertised and imported by all multihomed and remote PE routers that share the same EVI on the advertising ESI.

Autodiscovery Route Advertisement

- Active-standby mode

In active-standby mode, the designated forwarder (DF) advertises the autodiscovery route per Ethernet segment with an ESI MPLS label extended community that has the standby bit set to 1. The autodiscovery route is advertised per ESI, and the ESI label is set to 0 when active-standby mode is in operation.

The autodiscovery route is imported by all the multihomed and remote PE routers that are part of the EVI. On receiving the autodiscovery route, the PE routers in the network topology learn that active-standby multihoming mode is in operation for the ESI advertised.

- Active-active mode

In active-active mode, each of the multihomed PE device advertises a mandatory autodiscovery route per Ethernet segment as in the active-standby state. However, in the active-active state, the autodiscovery route per Ethernet segment is modified such that the active-standby bit carried in the MPLS extended community is cleared to indicate that the active-active mode is in operation. The autodiscovery route per Ethernet segment in the active-active mode also includes the split horizon label.

In [Figure 14 on page 169](#), for the ESI1 Ethernet segment, Routers PE1, PE2, and PE3 advertise the autodiscovery route. Router PE4 receives this autodiscovery route.

Autodiscovery Route Withdrawal

The autodiscovery route per Ethernet segment withdrawal may result in mass withdrawal. The mass withdrawal feature is used when there is a link failure on the ESI, or when the ESI configuration changes.

When the link between a multihomed CE device and a multihomed PE device fails, the PE device withdraws the autodiscovery route per Ethernet segment. In such a case, the mass withdrawal feature is handled in the following ways by the other PE devices:

- Remote PE device

When a remote PE device receives the BGP update for mass withdrawal, the following is performed at the remote PE device:

1. The current next hop to reach the remote ESI or CE device is deleted.
2. A new next hop through the remaining multihomed PE devices is created to reach the remote ESI or CE device.
3. All the MAC routes behind the CE device are updated with the newly created next hop.

Starting with Junos OS Release 17.4R1, Junos OS supports Dynamic List Next Hops in an EVPN network. Now when the link between the CE device and a multihomed PE device fails, the next hop to the ESI or CE is updated, thus reducing the need for a mass withdrawal. For more information on enabling Dynamic List Next Hop, see ["Configuring Dynamic List Next Hop" on page 510](#).

- Other multihomed PE device

As a result of the mass withdrawal, load balancing on the multihomed CE device happens because of the following:

- When the other multihomed PE devices receive the same set of MAC addresses on the link to the concerned ESI.

In this case, the local routes are preferred. If the remote routes learned from the DF PE device gets withdrawn, it does not affect routes pointing to the local ESI.

- When the other multihomed PE devices have not received the same set of MAC addresses on the link to the concerned ESI.

In this case, the PE devices install the MAC routes pointing to the concerned ESI, although the MACs are remotely learned from the DF PE device. When the DF PE device withdraws these routes, the withdrawn routes are flushed. Packets that are destined to the flushed MAC addresses are flooded on all the local segments.

Ethernet Segment Route

Ethernet Segment Route Features

The Ethernet segment route NLRI features include:

- This is an EVPN Type 4 route. The purpose of this route is to enable the PE routers connected to the same Ethernet segment to automatically discover each other with minimal configuration on exchanging this route.
- This route is associated with an ES-import extended community with an ESI value condensed to 6 bytes, similar to a route target.
- This route is advertised and imported only by PE routers that are multihomed on the advertising Ethernet segment.

Ethernet Segment Route Advertisement

The Ethernet segment route is exchanged among all the PE routers within a data center with the ES-import extended community. The ES-import extended community is constructed based on the ESI PE routers that are multihomed, and the Ethernet segment route carries the ESI value related to the Ethernet segment on which the PE routers are multihomed.

The Ethernet segment routes are filtered based on the ES-import extended community, such that only the PE routers that are multihomed on the same Ethernet segment import this route. Each PE router that is connected to a particular Ethernet segment constructs an import filtering rule to import a route that carries the ES-import extended community.

Autodiscovery Route per EVPN Instance

In active-active mode, each of the multihomed PE devices advertise an autodiscovery route per EVPN instance (EVI) with a valid MPLS label. This route is advertised per ESI and is imported by the remote PE devices. The MPLS label included in the autodiscovery route per EVI is used later for aliasing.

New Extended Communities

An extended community is similar in most ways to a regular community. Some networking implementations, such as virtual private networks (VPNs), use extended communities because the 4-octet regular community value does not provide enough expansion and flexibility. An extended community is an 8-octet value divided into two main sections.

To support active-standby multihoming, the following extended communities have been introduced:

ESI-Import

This extended community is attached to the ES route, and is populated from the ESI-import value extracted from the configured ESI value under the interface. To solve the problem of a conflict with another regular route target, the type is set to 0x06, which has been allocated by IANA.

The ESI-import extended community route target populates the list of import route targets configured for the special instance from where the ES route using this community is advertised.

Therefore, incoming ESI routes with the same ESI-import value in the extended community are imported by the PE routers, if the PE router is configured with an Ethernet segment that has the same ESI value. Once the PE router receives a set of these ESI routes that have the same ESI-import extended community value, the DF and BDF election can be done locally.



NOTE: When the ESI-import extended community is not created implicitly, a policy must be configured to attach all the route targets to the autodiscovery route per Ethernet segment.

Split Horizon

With reference to [Figure 14 on page 169](#) for example, when a CE device that is multihomed to two or more PE devices on an Ethernet segment (ESI1) and operating in the active-active redundancy mode sends a BUM packet to one of the non-DF PE devices (say PE1), then Device PE1 forwards that packet to all or a subset of the other PE devices in that EVPN instance, including the DF PE device for that Ethernet segment. In this case the DF PE device that the CE device is multihomed to drops the packet without forwarding it back to the CE device. This filtering is referred to as split horizon.

- Split horizon signaling

The split horizon extended community is attached to the autodiscovery route per Ethernet segment. The value of the extended community is the split horizon or the Poisson label itself, which is 3 bytes, and is advertised as an opaque attribute.

- Split horizon advertisement

- In active-standby mode, the standby bit in the split horizon extended community is set to 1, and the ESI split horizon label is set to 0.
- In the active-active mode, the split horizon extended community is modified to clear the standby bit to 0 and includes a valid ESI label used for split horizon purposes.

- Split horizon MPLS routes

The DF PE device advertises an autodiscovery route per Ethernet segment with a split horizon label A, and an inclusive multicast route with label B for BUM traffic forwarding. On the DF, the BUM packet from the core can come with following labels:

- When the non-DF PE devices receive a BUM packet on their single-homed ESIs, the BUM packet is sent to the DF PE device with multicast label B.

- When the non-DF PE devices receive a BUM packet on ESI1, the BUM packet is sent to the DF PE device with two MPLS labels — the multicast label B as the outer label, and the split horizon label A as the inner label.

In the EVPN multihoming scenario, the multicast label B has the S-bit set to 1 when it is the only label in the label stack. In this case, the BUM packet needs to be flooded on all the local ESIs on the DF PE device. But the label B has the S-bit set to 0 when split horizon label A is the innermost label in the label stack. In this case, the BUM packets need to be flooded on all local ESIs on the DF PE device, except the ESI that maps to the split horizon label A.

Assuming that packets originated from a multihomed CE device to a non-DF PE device on multihomed segment ESI1, when the non-DF PE device sends this packet to the DF PE device, the ESI label that the DF advertised to the non-DF PE device in its autodiscovery route per Ethernet segment is pushed first. The non-DF PE device also pushes the inclusive multicast label that the DF PE device advertised in its inclusive multicast route and further pushes the LSP label. The MPLS header thus contains two labels within a 32-bit field.

The base EVPN functionality uses a table-next hop to stitch the MPLS table with its corresponding EVPN EVI table. In the EVPN EVI table, the mac-lookup is performed to switch the packet.

The following routes are programmed in the mpls.0 table for EVPN multicast:

- The (multicast-label, S=1) route points to the EVPN-EVI table-next hop.
- The (multicast-label, S=0) route points to the MPLS table-next hop. This route loops the packet back to the MPLS table after popping the multicast-label.
- The (split horizon-label) route points to the EVPN-EVI table-next hop. This is the same table-next hop that is used by the multicast-label, S=1 route.

Newer EVPN Route Types

EVPN multihoming mode supports the following EVPN route types:

- Autodiscovery route per Ethernet segment
- Autodiscovery route per EVPN instance (EVI)
- Ethernet segment route

These route types conform to the following naming convention:

`<route-type>:<RD>::<esi>::<route-specific>/304`

For example:

1. Autodiscovery route per Ethernet segment—1:10.255.0.2:0::112233445566778899::0/304

2. Autodiscovery route per EVI—1:100.100.100.1:1::22222222222222222222::0/304

3. Ethernet segment route—4:10.255.0.1:0::112233445566778899:10.255.0.1/304

where:

- **route-type**—Type of EVPN route.
 - 1—Autodiscovery route per Ethernet segment.
 - 1—Autodiscovery route per EVI.
 - 4—Ethernet segment route.
 - 5—Route with VXLAN/MPLS encapsulation
- **RD**—Route distinguisher value.

The route distinguisher value is set to the IP address of the PE router followed by 0.

- **esi**—Ethernet segment identifier. Displayed as 10 bytes of hexadecimal bytes, and leading 00 bytes are not displayed.
- **route-specific**—Differs per route type.
 - Autodiscovery route per Ethernet segment and autodiscovery route per EVI—This value is an MPLS label.



NOTE: The MPLS label is displayed in the extensive output, although it is not included in the prefix.

- Ethernet segment route—This value is the originating IP address.
- **304**—Maximum number of bits in an EVPN route. This is not very useful information and could be removed from the display. However, it might be useful in quickly identifying an EVPN route, either visually or with match operators.

Multihomed Proxy MAC and IP Address Route Advertisement

Starting in Junos OS Release 18.4R1, Junos sends proxy MAC and IP Address route advertisement from PEs that are multihomed to a CE device. Junos uses a proxy flag in the EVPN layer 2 attributes extended community to identify the message as a proxy MAC and IP Address advertisement. A PE that learns of a MAC and IP Address sends a normal EVPN type 2 (MAC and IP Address) route advertisement. The other PEs on the Ethernet Segment that learns of the new route from the remote PE now send a MAC and IP Address route message with the proxy bit set. If the MAC and IP address entry ages out or if the link between the PE and CE fails, the entries has to be relearned and traffic can be lost. This prevents traffic

loss when one of the connections to a leaf device fails. Multihomed Proxy MAC is automatically enabled.

Update to the MAC Forwarding Table

In active-standby EVPN multihoming, the MAC addresses are treated as routable addresses, and the MP-IBGP protocol is used to carry the customer MAC addresses. MAC learning at the PE routers does not occur in the data plane but in the control plane. This leads to more control applied in terms of the learning mechanism.

A PE router performs MAC learning in the data plane for packets coming from a customer network for a particular EVI. For CE MAC addresses that are behind other PE routers, the MAC addresses are advertised in BGP NLRI using a new MAC advertisement route type.

The MAC learning is of two types:

- Local MAC learning—PE routers must support the local MAC learning process through standard protocols.
- Remote MAC learning—Once the local learning process is completed, the PE routers can advertise the locally learned MAC address to remote PE router nodes through MP-IBGP. This process of receiving the remote MAC addresses of attached customers through MP-IBGP is known as the remote MAC learning process.

The MAC advertisement route type is used to advertise locally learned MAC addresses in BGP to remote PE routers. If an individual MAC address is advertised, the IP address field corresponds to that MAC address. If the PE router sees an ARP request for an IP address from a CE device, and if the PE router has the MAC address binding for that IP address, the PE router performs ARP proxy and responds to the ARP request.



NOTE: The ARP proxy is performed only for the gateway and not for the host.

The MPLS label field depends on the type of allocation. The PE router can advertise a single MPLS label for all MAC addresses per EVI, which requires the least number of MPLS labels and saves the PE router memory. However, when forwarding to the customer network, the PE router must perform a MAC lookup which can cause a delay and increase the number of CPU cycles.

Traffic Flow

In EVPN multihoming, traffic flow is performed in the forwarding-plane. Flood routes are created for flooding the packets, and are used in the following scenarios:

- When a packet is received on a local ESI

- When a packet is received from the core

The traffic flows in EVPN multihoming can be based on the two traffic types:

- Unicast traffic

Unicast traffic is a point-to-point communication with one sender and one receiver. In a multihomed EVPN, unicast traffic is forwarded as follows:

- In active-standby mode
 - CE to core—Traffic is learned and forwarded by the DF PE router.
 - Core to CE—The remote PE router learns the MAC addresses from the DF, and forwards all unicast traffic to the DF PE router.
- In active-active mode
 - CE to core—Traffic is load-balanced to all the connected multihomed PE devices.
 - Core to CE—Traffic from the remote PE devices is load-balanced to all the multihomed PE devices connected to the remote CE device.

- BUM traffic

Traffic that is sent to multiple destinations, including broadcast traffic, unknown unicast traffic that is broadcast in the Ethernet segment, and multicast traffic is known as BUM traffic. In a multihomed EVPN, BUM traffic is forwarded as follows:

- In active-standby mode
 - CE to core—The CE device floods any BUM traffic to all the links in the Ethernet segment. The DF PE router with the active path forwards the BUM packets to the core. The BDF PE router in the standby mode drops all the traffic from the CE device, because the EVPN multihomed status of the interface is in blocking state. However, if the CE device is connected to the PE devices using separate links or LAGs, the BUM traffic reaches both the DF and BDF PE devices.
 - Core to CE—The remote PE routers flood all BUM traffic to both the DF and BDF PE routers. Only the DF forwards the BUM traffic to the CE device. The BDF PE router drops all the traffic, because the EVPN multihomed status of the interface is in blocking state.
- In active-active mode

Based on the requirements, flooding and switching among local ESIs can be enabled or disabled in the active-active mode. This is referred to as the no-local-switching behavior.

The core of EVPN service provides a full-mesh connectivity among the multihomed PE devices. Because of this, EVPN uses split horizon in the core, so a packet received from the core is never

switched or flooded back to the core. Instead, ingress replication is used to replicate the packets to the remote PE devices.

To flood packets to remote PE devices, the multicast and the split horizon next hops are used. The multicast next hop tunnels the packet with the inclusive multicast label, and the split horizon next hop tunnels the packet with a multicast-label and a split horizon label. One such next hop is required per multihomed ESI per remote PE device.

The following flood routes are used in the active-active mode:

- All-CE flood route

This flood route is used by the local ESIs for the following:

- Flooding the packet on the local ESIs (when local-switching is allowed).
- Flooding the packet to the remote PE devices. The remote PE devices flood the packet on their local ESIs.

Because BUM traffic is forwarded only by the designated forwarder (DF), and not by the non-DF multihomed PE devices, the non-DFs use the split horizon next hop to flood this packet to other PE devices. However, the multihomed local ESIs for which the PE device is a non-DF does not participate in the flooding.

The all-CE flood route is not used by the non-DF ESIs, and the next hop for these flood routes is created accordingly. In such cases, the non-DF ESI flood route is used.

- All-VE flood route

This flood route is used when the packet is received from the core. It is used for flooding the packet received from the core to the local ESIs. Because the packet received from the core can come with multicast-label only or with both multicast-label and split horizon label, appropriate forwarding rules must be followed to drop the packet on the multihomed ESI that maps to the split horizon label.

- Non-DF flood route

This flood route is used for the following:

- Flooding the packet on the local ESIs.
- Flooding the packet to the remote PE devices using ingress replication with SH-label for the DF for the ESI.

Aliasing

Starting in Junos OS Release 15.1, Junos OS supports aliasing in an EVPN. Aliasing is the ability of a remote PE device to load balance Layer 2 unicast traffic on all the other PE devices that have same Ethernet segment towards a CE device.

Aliasing in the Active-Active Mode

In [Figure 14 on page 169](#), aliasing in the active-active mode works as follows:

1. ESI1 is configured on Routers PE1, PE2, and PE3. Routers PE1, PE2, and PE3 advertise the autodiscovery route per Ethernet segment for ESI1.
2. Device CE1 sends Layer 2 traffic with source MAC address (MAC1) to Router PE1.
3. Router PE1 learns the MAC1 address on (ESI1, vlan X) and advertises it to all PE routers using BGP.
4. Router PE4 receives the MAC1 route through BGP.
5. Because Router PE4 also received the autodiscovery route per EVI from Routers PE2 and PE3, it knows that MAC1 must be reachable through Routers PE2 and PE3. Router PE4 builds its forwarding state to load-balance the Layer 2 traffic for MAC1 among Routers PE1, PE2, and PE3.

Aliasing and Autodiscovery Routes

Autodiscovery routes from Routers PE2 and PE3 can come in any order. As a result, these routes are installed by the Layer 2 process as follows:

1. After receiving MAC1 from Router PE1, and if any autodiscovery routes have not been received by Router PE4, MAC1 is programmed by PE4 with a next hop pointing toward Router PE1. When PE4 receives the autodiscovery route from Router PE2 for the same ESI, the next hop is installed so the traffic for MAC1 is load-balanced to Routers PE1 and PE2. When PE4 receives the autodiscovery route from Router PE3 for the same ESI, the next hop is updated to load-balance the traffic for MAC1 among Routers PE1, PE2, and PE3.
2. If Router PE4 has already received the autodiscovery routes from more than one PE device (PE1, PE2, and PE3), PE4 installs the MAC routes with the multi-destination next hop.

Aliasing and Label Route

Any PE device that advertises the autodiscovery route per EVI with a valid MPLS label programs the advertised label in the mpls.0 routing table. For instance, if Router PE2 advertised the autodiscovery route per EVI with label A, the mpls.0 entry is as follows:

Label A route points to the EVPN-EVI table-next hop.

When the remote Router PE4 sends a unicast data packet toward Router PE2 with this label A, lookup is done in Router PE2's forwarding table, and as a result of this lookup, the packet is forwarded on ESI1.

Aliasing and Unicast Packet Forwarding

When the unicast packets for MAC1 come from the remote Router PE4 to Router PE2, there could be two cases:

- Router PE2 also received the same set of MACs on its link to ESI1—In this case, local routes are preferred and as a result of the MAC lookup, packets are forwarded to ESI1.
- Router PE2 has not received the same set of MACs on its link to ESI1—In this case, Router PE2 still installs MAC routes pointing to ESI1, although MACs are remotely learned from Router PE1. As a result, the packets are forwarded to ESI1.

EVPN Active-Active Multihoming and Multichassis Link Aggregation

When a CE device is configured with a LAG toward the PE devices, the following two options are available to run LACP on the PE devices:

- Configure the same LACP system ID on all the PE devices.
- Configure multichassis link aggregation on the PE devices.

When multichassis link aggregation is configured with EVPN, a reduced set of procedures for active-active multichassis link aggregation are required. These procedures provide link and node level redundancy. The multichassis link aggregation is completely transparent to the CE device, and is realized as pure LAG. Multichassis link aggregation operates at the port level as well. This essentially means that if multichassis link aggregation is configured as active-active, all VLANs on the multichassis link aggregation ports work in the active-active multihoming mode.

When multichassis link aggregation is configured along with EVPN, the following is considered:

- Both multichassis link aggregation and EVPN ESI must be enabled to work in the active-active mode only.
- The following functions are not required for multichassis link aggregation with EVPN:
 - Mac synchronization—This is performed in the BGP control plane of EVPN.
 - ICL linking—This is handled by the aliasing feature of EVPN.
 - ARP synchronization—This is handled by the BGP control plane with IRB functionality.

EVPN Active-Active Multihoming and IRB

When IRB is configured, the EVPN routes contain both MAC and IP information. The active-active multihoming requires ARP synchronization among the multihomed PE devices because the ARP responses can get hashed to a particular PE device.

Sample Configuration

The following is a sample configuration for EVPN active-standby multihoming on the following types of interfaces:

- Ethernet interface configuration

```
ge-0/1/2 {
  encapsulation ethernet-bridge;
  esi XX:XX:XX:XX:XX:XX:XX:XX:XX:XX;
  unit 0 {
    family bridge;
  }
}
```

- Single VLAN interface configuration

```
ge-0/1/3 {
  encapsulation extended-vlan-bridge;
  esi XX:XX:XX:XX:XX:XX:XX:XX:XX:XX;
  vlan-tagging
  unit 0 {
    family bridge;
    vlan-id 1;
  }
}
```



NOTE:

- An ESI value of 0 and all FFs are reserved and are not used for configuring a multihomed Ethernet segment.
- Two interfaces in the same EVI cannot be configured with the same ESI value.

The following is a sample routing instance configuration for EVPN active-standby multihoming:

- Routing instance configuration

```

routing-instances {
  evpn-0 {
    instance-type evpn;
    route-distinguisher value;
    vrf-target value;
    vlan-id vlan-ID;
    interface ge-0/1/2.0;
    interface ge-1/1/1.0;
    interface ge-2/2/2.0;

    protocols {
      evpn {
        designated-forwarder-election-hold-time time;
      }
    }
  }
}

```



NOTE: With the active-standby mode configuration, the autodiscovery route per Ethernet segment is advertised with the active-standby bit set to 1 for each Ethernet segment.

Designated Forwarder Election

The designated forwarder (DF) election procedure in EVPN provides a robust mechanism for electing a designated forwarder among the devices that serve a multihomed Ethernet segment. The DF election ensures efficient and effective broadcast, unknown unicast, and multicast (BUM) traffic forwarding, load balancing, high availability, and traffic management. Key features include:

- Modulo based or preference-based algorithms for DF selection.
- Dynamic DF re-election triggered by network state changes.
- DF and Backup DF roles to prevent traffic loops.

The system also processes ESI routes to optimize route processing, and uses mass withdrawal mechanisms to maintain network integrity. You can use these features and their CLI configurations to ensure seamless traffic flow and effectively manage your EVPN multihoming scenarios.

For more information, see ["EVPN Multihoming Designated Forwarder Election" on page 186](#).

ESIs on Physical, Aggregated Ethernet, and Logical Interfaces

In releases before Junos OS Release 15.1F6 and 17.1R1 for MX Series routers and Junos OS Release 17.3R1 for EX9200 switches, you can specify an ESI only on a physical or aggregated Ethernet interface, for example, `set interfaces ae0 esi 00:11:22:33:44:55:66:77:88:99`. If you specify an ESI on a physical or aggregated Ethernet interface, keep in mind that an ESI is a factor in the designated forwarder (DF) election process. For example, assume that you configure EVPN multihoming active-standby on aggregated Ethernet interface ae0, and given the ESI configured on ae0 and other determining factors, the DF election results in ae0 being in the down state. Further, all logical interfaces configured on aggregated Ethernet interface ae0, for example, `set interfaces ae0 unit 1` and `set interfaces ae0 unit 2` are also in the down state, which renders logical interfaces ae0.1 and ae0.2 unable to provide services to their respective customer sites (VLANs).

To better utilize logical interfaces in EVPN multihoming active-standby or active-active mode, starting with Junos OS Releases 15.1F6 and 17.1R1 for MX Series routers and Junos OS Release 17.3R1 for EX9200 switches, you can specify an ESI on a logical interface. As a result, even if a logical interface is a non-DF, other logical interfaces on the same physical or aggregated Ethernet interface are still able to provide services to their respective customer sites (VLANs).

For more information, see ["Example: Configuring an ESI on a Logical Interface With EVPN Multihoming" on page 493](#).

Automatically Generated ESIs

Starting with Junos OS Release 18.4R1, you can configure aggregated Ethernet interfaces and aggregated Ethernet logical interfaces to automatically derive ESIs from the LACP configuration. We support this feature in the following environments:

- On Juniper Networks devices that support this feature and are multihomed in active-active mode in an EVPN-VXLAN overlay network.
- On Juniper Networks devices that support this feature and are multihomed in active-standby or active-active mode in an EVPN-MPLS overlay network.

For more information, see ["Understanding Automatically Generated ESIs in EVPN Networks" on page 200](#).

Convergence in an EVPN Network

When there are changes in the network topology in a large-scale EVPN system, the convergence time might be significant. You can prioritize NLRI updates that are critical to route selection in routing policies to improve convergence. [Table 12 on page 184](#) lists the NLRI route types and the priority that must be configured in the routing policy.

Table 12: Priority for NLRI Route Type

NLRI Route Type	Description	Priority
NLRI Route Type 1	Ethernet auto-discovery route—Type 1 supports fast convergence and aliasing and is used to signal MAC mass withdrawal.	High
NLRI Route Type 2	MAC/IP advertisement route—Type 2 is used to advertise MAC addresses and IP addresses in EVPN networks.	Low
NLRI Route Type 3	Inclusive multicast Ethernet tag—Type 3 is used to set up a path for BUM traffic.	Low
NLRI Route Type 4	Ethernet segment route—EVPN Type 4 is used in the selection of a designated forwarder.	High

To prioritize the NLRI route type, set the `bgp-output-queue-priority` priority for `nlri-route-type` at the `[edit policy-options policy-statement]` hierarchy level on all provider edge routers and route reflectors in the EVPN network. In this example, a high priority was configured for NLRI route type 1 and NLRI route Type 4.

```

user@PE1#show policy-options
policy-statement evpn-rt-priority-policy {
  term 1 {
    from {
      family evpn;
      nlri-route-type 1;
    }
    then {
      bgp-output-queue-priority priority 16;
    }
  }
  term 2 {
    from {
      family evpn;
      nlri-route-type 2;
    }
    then {
      bgp-output-queue-priority priority 1;
    }
  }
}

```

```
}
term 3 {
  from {
    family evpn;
    nlri-route-type 3;
  }
  then {
    bgp-output-queue-priority priority 2;
  }
}
term 4 {
  from {
    family evpn;
    nlri-route-type 4;
  }
  then {
    bgp-output-queue-priority priority 16;
  }
}
}
```



NOTE: There are 17 prioritized output queues: an expedited queue that has the highest priority, and 16 numbered queues for which 1 is the lowest priority and 16 is the highest.

For more information about how to configure routing policies, see [Routing Policies, Firewall Filters, and Traffic Policers User Guide](#).

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
18.4R1	Starting with Junos OS Release 18.4R1, you can configure aggregated Ethernet interfaces and aggregated Ethernet logical interfaces to automatically derive ESIs from the LACP configuration.
17.4R1	Starting with Junos OS Release 17.4R1, Junos OS supports Dynamic List Next Hops in an EVPN network.
16.1R4	Starting with Junos OS Releases 16.1R4 and 16.2R2, all EX9200 switches support the active-active mode for EVPN multihoming.

16.1R4	Starting with Junos OS Releases 17.4R1 QFX10000 switches support the active-active mode for EVPN multihoming.
15.1F6	To better utilize logical interfaces in EVPN multihoming active-standby or active-active mode, starting with Junos OS Releases 15.1F6 and 17.1R1 for MX Series routers and Junos OS Release 17.3R1 for EX9200 switches, you can specify an ESI on a logical interface. As a result, even if a logical interface is a non-DF, other logical interfaces on the same physical or aggregated Ethernet interface are still able to provide services to their respective customer sites (VLANs).
15.1	Starting in Junos OS Release 15.1, Junos OS supports aliasing in an EVPN.
14.1x53-D30	Starting with Junos OS Release 14.1x53-D30 for QFX5100 switches and Junos OS Release 18.2R1 for EX4600 switches, these switches support the active-active mode of operation for EVPN multihoming.
14.1R4	Starting with Junos OS Release 14.1R4, 14.2, 15.1F6, and 16.1R1, Junos OS supports the active-active mode for EVPN multihoming on MX Series routers.

RELATED DOCUMENTATION

[Configuring Dynamic List Next Hop | 510](#)

[Example: Configuring EVPN-MPLS Active-Standby Multihoming | 308](#)

[Example: Configuring EVPN Active-Active Multihoming | 375](#)

EVPN Multihoming Designated Forwarder Election

SUMMARY

The designated forwarder (DF) manages broadcast, unknown unicast, and multicast (BUM) traffic to prevent loops and ensure efficient traffic distribution.

IN THIS SECTION

- [DF Election Overview | 187](#)
- [DF Election Roles | 188](#)
- [DF Election Trigger | 189](#)
- [DF Election Procedure \(RFC 7432\) | 189](#)
- [NTP-Based DF Election | 190](#)

- [Preference-Based DF Election | 192](#)
- [DF Verification | 198](#)
- [DF Election for Virtual Switch | 199](#)
- [Handling Failover | 199](#)

DF Election Overview

IN THIS SECTION

- [Benefits of Designated Forwarder \(DF\) Election | 188](#)

Depending on the multihoming mode of operation, traffic to a multihomed customer edge (CE) device uses one or all the multihomed provider edge (PE) devices to reach the customer site. The designated forwarder (DF) election procedure ensures only one endpoint, the DF handles the broadcast, unknown unicast, and multicast (BUM) traffic for a given Ethernet segment, thereby preventing forwarding loops and optimizing network performance.

The DF election process dynamically responds to various network events such as configuration changes, BGP session transitions, or link state changes. This adaptability enables the network to maintain efficient traffic forwarding without manual intervention. When a triggering event occurs, the DF election mechanism re-evaluates and potentially reassigns the DF role, maintaining optimal traffic handling across the network.

The DF election hold timer prevents the election process from starting prematurely. This ensures the network has time to stabilize before the election procedure begins. The timer defaults to 3 seconds. However, you can modify it with the *designated-forwarder-election-hold-time* statement to suit your network's stability and performance needs. This timer value must be the same across all the PE routers connected to the same Ethernet segment.

The default DF election procedure (as specified in RFC 7432) uses IP addresses and service carving to elect a DF for each EVPN instance (EVI). This election procedure is revertive, so when the elected DF fails and then recovers from that failure it will preempt existing DF.

The NTP-based Designated Forwarder (DF) election mechanism ensures the DF election is performed simultaneously at a predetermined Service Carving Time (SCT) within an Ethernet Segment. This synchronization minimizes the risks of loops, duplicate traffic, and traffic discarding associated with traditional timer-based elections. This mechanism ensures that DF roles are consistently applied across all PEs, improving robustness and reliability in scenarios such as the addition of new PEs or recovery

from failures. Key features include skew handling to prevent duplicate traffic, coordinated recovery processes for multiple PE failures, and compatibility with existing DF election algorithms. Enhancements also allow for per EVPN instance (*per-ev*) DF election and integration with both IPv4 and IPv6 underlay networks.

The preference-based DF election procedure uses manually configured preference values, the Don't Preempt (DP) bit, and router ID or loopback address to elect the DF. Starting in Junos OS Release 24.2, the *preference* statement includes a *non-revertive* option that prevents the preemption of the existing DF after a failure. This *non-revertive* option avoids a service impact if the old DF recovers after failing. The default behavior is revertive.

Benefits of Designated Forwarder (DF) Election

- **Prevents forwarding loops**—By electing a single multihomed Ethernet segment to handle BUM traffic, the DF election ensures that only one endpoint forwards traffic, significantly reducing the risk of forwarding loops and enhancing network stability.
- **Adapts dynamically to network changes**—The DF election process responds dynamically to network changes, such as new interface configurations or recovery from link failures, maintaining network stability and operational efficiency.
- **Ensures efficient failover**—The presence of a backup DF (BDF), which remains in a blocking state until it is needed to take over, ensures smooth failover and continuous network operation with minimal traffic disruption, thereby improving overall network resilience.
- **Reduces route processing overhead**—Utilizing the ES-Import extended community for route filtering ensures that only relevant routes are imported by PEs connected to the same Ethernet segment, which reduces unnecessary route processing and maintains efficient route management.
- **Maintains network consistency**—The mass withdrawal mechanism, triggered by the withdrawal of Ethernet autodiscovery routes after a link failure, invalidates stale MAC addresses on remote PEs, ensuring that the network state remains consistent and preventing issues caused by outdated MAC address information.
- **Distributes traffic efficiently**—The DF election process balances the load across multiple PEs, ensuring that no single segment is overwhelmed with traffic, which optimizes network performance and resource utilization.

DF Election Roles

The designated forwarder (DF) election process involves selecting a forwarding role as follows:

- **Designated forwarder (DF)**—The PE router that announces the MAC advertisement route for the customer site's MAC address. This PE router is the primary PE router that forwards BUM traffic to the multihomed CE device and is called the designated forwarder (DF) PE router.

- **Backup designated forwarder (BDF)**—Each router in the set of PE routers that advertise the Ethernet autodiscovery route for the same ESI and serve as the backup path in case the DF fails, is called a backup designated forwarder (BDF). A BDF is also called a non-DF router.

The DF election process elects a local PE router as the BDF, which then puts the multihomed interface connecting to the customer site into a blocking state for the active-standby mode. The interface stays in the blocking state until the BDF is elected as the DF for the Ethernet segment.

- **Non-designated forwarder (non-DF)**—Other PE routers not selected as the DF. The BDF is also considered to be a non-DF.

DF Election Trigger

In general, the following conditions will trigger the DF election process:

- When you configure an interface with a nonzero ESI, or when the PE router transitions from an isolated-from-the-core (no BGP session) state to a connected-to-the-core (has established BGP session) state. These conditions also trigger the hold timer. By default, the PE puts the interface into a blocking state until the router is elected as the DF.
- After completing a DF election process, a PE router receives a new Ethernet segment route or detects the withdrawal of an existing Ethernet segment route. Neither of these trigger the hold timer.
- When an interface of a non-DF PE router recovers from a link failure. In this case the PE router has no knowledge of the hold time imposed by other PE routers. As a result, the recovered PE router does not trigger a hold timer.

DF Election Procedure (RFC 7432)

Service carving refers to the default procedure for DF election at the granularity of the ESI and EVI. With *service carving*, it is possible to elect multiple DFs per Ethernet segment (one per EVI) to perform load-balancing of multi-destination traffic for a given Ethernet segment. The load-balancing procedures carve up the EVI space among the PE nodes evenly, in such a way that every PE is the DF for a disjoint set of EVIs.

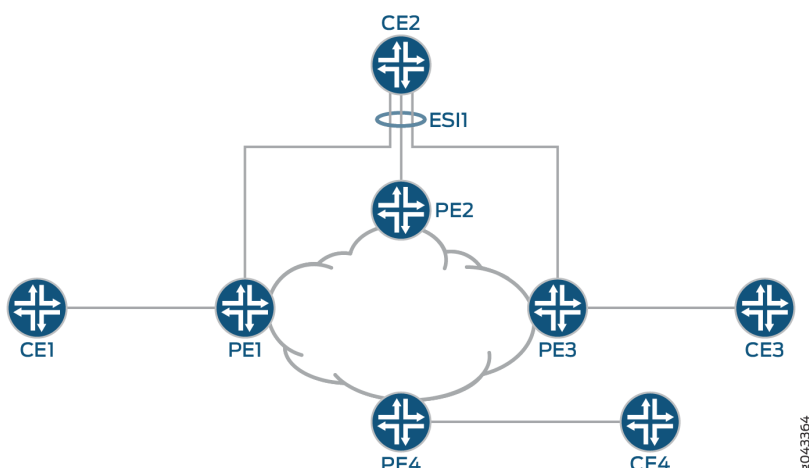
The service carving procedure is as follows:

1. When a PE router discovers the ESI of the attached Ethernet segment, it advertises an autodiscovery route per Ethernet segment with the associated ES-import extended community attribute.
2. The PE router starts a hold timer (default value of 3 seconds) in order to receive the autodiscovery routes from other PE nodes connected to the same Ethernet segment. This timer value must be the same across all the PE routers connected to the same Ethernet segment.

You can overwrite the default hold timer using the *designated-forwarder-election-hold-time* configuration statement.

3. When the hold timer expires, each PE router builds an ordered list of the IP addresses of all the PE nodes connected to the Ethernet segment (including itself), in increasing numeric order. The system assigns every PE router an ordinal indicating its position in the ordered list, starting with 0 for the PE with the numerically lowest IP address. The DF for a given EVI is then determined by the ordinal matching the VLAN ID modulo the number of PEs, providing a deterministic and predictable method for DF selection. For example, if the VLAN ID is 10 and there are three PEs, the DF would be the PE with the ordinal number that corresponds to 10 modulo 3, ($10 \bmod 3 = 1$).
4. The PE router elected as the DF for a given EVI unblocks traffic for the Ethernet tags associated with that EVI. The DF PE unblocks multi-destination traffic in the egress direction toward the Ethernet segment. All the non-DF PE routers continue to drop multi-destination traffic (for the associated EVIs) in the egress direction toward the Ethernet segment.

Figure 15: Active-Active EVPN Multihoming



In [Figure 15 on page 190](#), Routers PE1, PE2, and PE3 perform a DF election for active-active multihoming. Each router can become the DF for a particular VLAN from a range of VLANs configured on ESI1 and a non-DF for other VLANs. Each DF forwards BUM traffic on the ESI and VLAN it serves. The non-DF PE routers block the BUM traffic on those particular Ethernet segments.

NTP-Based DF Election

IN THIS SECTION

- [Overview | 191](#)
- [Benefits of NTP-Based DF Election | 191](#)

- Integration with Existing Algorithms | 192
- Skew Handling and Concurrent Recoveries | 192

Overview

The NTP-based DF election mechanism leverages NTP or PTP to synchronize the DF election on all peering PE devices within an EVPN segment. When a new PE device joins the segment or a previously failed PE recovers, it announces a "Service Carving Time" (SCT) to its peers. This SCT is an absolute timestamp at which all PEs simultaneously execute the DF election process, ensuring that the DF role is assigned consistently across the network. This precise synchronization mitigates the risks of traffic loops, duplicates, and discarding that can result from asynchronous DF elections.

The NTP-based DF election introduces additional enhancements to improve network stability. Key features include skew handling to prevent duplicate traffic, coordinated recovery processes for multiple PE failures, and compatibility with existing DF election algorithms.

To configure the NTP-based DF election, you enable the feature using the `df-election-ntp` command under the global `protocols evpn` configuration. Additionally, the `designated-forwarder-election-hold-time` command specifies the delay time for the election in seconds, with a range of 1-1800 seconds and a default of 3 seconds. For monitoring, the `show route table` extensive command displays the SCT extended community on Type 4 routes, including columns and fields to indicate the NTP-based DF election capability and SCT timestamps. These commands facilitate precise configuration and monitoring, ensuring that your network infrastructure maintains synchronized and reliable DF elections.

Benefits of NTP-Based DF Election

- Minimizes the risk of traffic loops and discarding by ensuring that all PE devices apply the DF election results simultaneously.
- Enhances network stability through precise timing coordination, reducing the chances of duplicate traffic during the DF election process.
- Supports seamless integration with existing DF election algorithms and both IPv4 and IPv6 underlay networks, ensuring compatibility and flexibility with current network setups.
- Improves recovery efficiency by coordinating the DF election process during simultaneous PE recoveries, preventing redundant election executions and maintaining a single synchronized event.
- Allows for per EVPN instance DF election, enhancing granularity and control over DF role assignments, which is particularly beneficial for VLAN-based services.

Integration with Existing Algorithms

The NTP-based DF election mechanism is designed to integrate seamlessly with existing DF election algorithms. It supports both MOD-based and preference-based algorithms, providing flexibility in how DF roles are assigned within the network. This compatibility ensures that the NTP-based mechanism can be adopted without requiring significant changes to your current DF election configurations. Additionally, the mechanism supports both IPv4 and IPv6 underlay networks, enabling you to deploy the feature across diverse network environments.

For VLAN-based services, the NTP-based DF election supports per EVPN instance (*per-evi*) DF election. This feature ensures that DF role switchovers occur only if all access interfaces under the EVPN instance are down, providing finer granularity and control over DF assignments. This granularity is particularly beneficial in scenarios where VLAN-based services require precise control over traffic forwarding decisions. Overall, the NTP-based DF election mechanism enhances the stability and reliability of your EVPN segments, ensuring consistent and synchronized DF elections across your network.

Skew Handling and Concurrent Recoveries

The NTP-based DF election introduces a skew handling mechanism to prevent duplicate traffic. This mechanism introduces a slight delay (default -10 milliseconds) to ensure that previously active PEs perform the service carving just before the newly inserted PE. This prevents duplicate traffic by ensuring that all PEs have a current and consistent view of the network state before the election. Furthermore, in scenarios where multiple PEs recover simultaneously, the DF election process is triggered only once, based on the largest SCT value received. This single coordinated election event eliminates the potential for redundant DF elections and ensures network consistency.

If any PE does not support the NTP-based DF election, the system reverts to the default timer-based DF election as specified in RFC 7432. This fallback mechanism ensures compatibility and continuity across the network.

Preference-Based DF Election

IN THIS SECTION

- [Overview | 193](#)
- [Benefits of Preference-Based Designated Forwarder \(DF\) Election | 193](#)
- [Preference-Based DF Election Procedure | 194](#)
- [DF Election Algorithm Mismatch | 194](#)
- [DF Election Algorithm Migration | 195](#)

- [Changing Preference for Maintenance | 195](#)
- [Non-Revertive Mode | 195](#)
- [Load Balancing with Preference-Based DF Election | 196](#)

Overview

The DF election based on RFC 7432 fails to meet the operational requirements of some service providers. To address this issue, we introduced the preference-based DF election feature that enables control of the DF election based on administrative preference values set on interfaces.

The preference-based DF election feature offers network operators the flexibility to manage DF roles with preference values configured on interfaces. In scenarios where the primary link must handle most of the traffic, this strategy optimizes both throughput and resource allocation.

Starting in Junos OS Release 24.2, we provide more configuration options to customize the preference-based DF election process:

- The *preference non-revertive* option improves network stability by ensuring a previously designated DF does not preempt the current DF when it comes back online after a failure.
- The *preference least*, and the *evpn designated-forwarder-preference-highest* and *evpn designated-forwarder-preference-least* statements enable you to select whether the election process uses the highest or lowest preference values at the ESI and EVI levels.

Please refer to [Feature Explorer](#) for a complete list of the products that support these features.

- [Preference-based DF election for EVPN and PBB-EVPN](#)
- [EVPN\(MAC-VRF\) Multi-Homing Preference based DF Election](#)

Benefits of Preference-Based Designated Forwarder (DF) Election

- **Optimized traffic flow**—Configuring the DF based on interface attributes like bandwidth ensures optimal link selection. This results in more efficient traffic distribution and better use of network resources.
- **Enhanced operational control**—Manually configuring preference values gives you greater control over the DF election process and ensure the most suitable link is used.
- **Enhanced network stability**—The *preference non-revertive* option prevents a returning DF from preempting the current DF. This eliminates unnecessary disruptions, and ensures continuous service stability.

- **Granular load balancing control**—You can configure DF election preferences. You can also select the DF based on the highest or lowest preference values at the ESI and EVI levels. This enables you to effectively distribute traffic across multiple links, leading to improved network performance.
- **Maintenance flexibility**—You can adjust preference values to switch the DF role during maintenance activities, facilitating operational flexibility and reducing the impact of maintenance on service continuity.

Preference-Based DF Election Procedure

The preference-based DF election uses manually configured interface preference values when electing a DF. Manual configuration of preference values gives you enhanced control over the DF election process. You can set specific preferences on interfaces to influence which node acts as the DF.

The preference-based DF election proceeds as follows:

1. Configure the DF election type *preference value* under an ESI.
2. The multihoming PE devices advertise the configured preference value and DP bit using the DF election extended community in the EVPN Type 4 routes.
3. After receiving the EVPN Type 4 route, the PE devices build the list of candidate DF devices, in the order of the preference value, DP bit, and IP address.
4. When the DF timer expires, the PE devices elect the DF.

By default, the DF election is based on the highest preference value. However, you can configure the preference-based DF election process to choose the DF based on the lowest preference value using the *preference least* or *evpn designated-forwarder-preference-least* statements.



NOTE: The EVPN configuration for *evpn designated-forwarder-preference-highest* or *designated-forwarder-preference-least* should be the same on the competing multihoming EVIs; otherwise the election process might elect two DFs, which can cause traffic loss or traffic loops.

5. If multiple DF candidates have the same preference value, then the PE device selects the DF based on the DP bit. When those DF candidates have the same DP bit value, the process elects the DF based on the lowest IP address.

DF Election Algorithm Mismatch

When there is a mismatch between a locally configured DF election algorithm and a remote PE device's DF election algorithm, then all the PE devices should fall back to the default DF election as specified in RFC 7432.

DF Election Algorithm Migration

Migration from the traditional modulo-based DF election to the new preference-based method requires careful planning. Typically, this involves a maintenance window where interfaces with the same ESI on non-DF PEs are brought down. You then configure the new DF election algorithm on the DF PE before applying it to the other multihoming PEs. This structured approach ensures a smooth transition with minimal service impact.

Perform the migration using the following steps:

1. Bring down all the interfaces with the same ESI on the non-DF PE devices.
2. Configure the current DF PE with the preference-based DF election options.
3. Configure the preference-based DF election options on the non-DF PE devices.
4. Bring up all the interfaces on the non-DF PE devices.

After reconfiguring and bringing the interfaces back online, verify that the DF election process is functioning as intended. Monitor the network to ensure that the designated forwarders are correctly elected based on the configured preferences. This step is crucial to confirm that the new settings are correctly applied and that the network operates smoothly with the enhanced DF election mechanism.

Changing Preference for Maintenance

The ability to change preference values during maintenance activities enhances operational flexibility. You can switch DF roles as needed by simply changing the configured preference value on a selected device.

Change the DF for a given ESI by performing one of the following steps:

1. Change the preference value to a higher value on a current non-DF device.
2. Change the preference value to a lower value on the current DF device.



NOTE: Changing the preference value for an ESI can lead to some traffic loss during the short duration required to integrate the delay in the updated BGP route propagation with the new preference value.

Non-Revertive Mode

Beginning with Junos OS Release 24.2R1, you can enable the `non-revertive` option under the `[edit interfaces name esi df-election-type preference]` hierarchy, which helps to ensure that your network remains stable across link failures and recoveries. When you configure this option per ESI, it provides granular control and ensures each segment adheres to the desired operational behavior.

The non-revertive option ensures that once a DF is elected, it will not be preempted by the previously designated DF coming back online after a failure. This non-revertive mode is key in maintaining a stable network environment and avoiding unnecessary service disruptions.

Please refer to [Feature Explorer](#) for a complete list of the products that support this feature.

- [Non-revertive preference-based DF election in EVPN-MPLS networks](#)



NOTE: The non-revertive preference-based DF election option doesn't work if:

- You configure the `no-core-isolation` option under the `[edit protocols evpn]` hierarchy, and any of the following events occur:
 - You reboot the device.
 - You run the `restart routing` command.
 - You run the `clear bgp neighbor` command.
- You have the graceful restart (GR) feature enabled (`set routing-options graceful-restart`), and the device goes through a graceful restart.

Load Balancing with Preference-Based DF Election

The preference-based DF election enables load balancing by selecting DFs based on the highest or lowest preference values. By default, the DF is selected based on the highest preference value. You can configure DF election type `preference least` on the interface (ESI level) to choose the DF based on lowest preference value.

```
[edit interfaces interface-name]
esi {
  XX:XX:XX:XX:XX:XX:XX:XX:XX;
  df-election-type {
    preference {
      value value;
      least;
    }
  }
}
```

You can also configure *evpn* *designated-forwarder-preference-highest* or *designated-forwarder-preference-least* at the EVI level.

```
[edit routing-instances]
  instance-name {
    instance-type evpn;
    protocols {
      evpn {
        (designated-forwarder-preference-highest | designated-forwarder-preference-
least);
      }
    }
  }
}
```

The EVI level configurations override the ESI level configurations when both are used as shown in [Table 13 on page 197](#) below.

You can use these configurations to manage load balancing in different scenarios as in the following examples.

Single ESI Under Multiple EVIs

Configure load balancing for a single ESI under multiple EVIs using various combinations of the following statements:

- DF election type *preference* *least* at the ESI level.
- *evpn* *designated-forwarder-preference-least* or *designated-forwarder-preference-highest* at the EVI level.

Table 13: Load balancing DF selection with one ESI and multiple EVIs

Case No.	preference least configured on ESI	designated- forwarder- preference-least configured on EVI-1	designated- forwarder- preference- highest configured on EVI-2	Result on EVI-1	Result on EVI-2
1	No	No	No	Highest	Highest

Table 13: Load balancing DF selection with one ESI and multiple EVIs (Continued)

Case No.	preference least configured on ESI	designated-forwarder-preference-least configured on EVI-1	designated-forwarder-preference-highest configured on EVI-2	Result on EVI-1	Result on EVI-2
2	No	Yes	Yes (optional command)	Lowest	Highest
3	Yes	No	No	Lowest	Lowest
4	Yes	No	Yes	Lowest	Highest

Multiple ESIs Under a Single EVI

When configuring load balancing for multiple ESIs under a single EVI, use the ESI default setting to select the highest preference or configure *preference* least on the ESI to select the lowest preference.

Do not configure the *evpn* designated-forwarder-preference-least or designated-forwarder-preference-highest statements at the EVI level because they will override the ESI level configurations.

Table 14: Load balancing DF selection with multiple ESIs under a single EVI

preference least configured on ESI-1	preference least on ESI-2	Result for ESI-1 on EVI-1	Result for ESI-2 on EVI-1
No	No	Highest	Highest
Yes	No	Lowest	Highest

DF Verification

The following show commands offer detailed insights into DF election preferences and statuses, aiding in effective troubleshooting and monitoring:

- [show evpn instance extensive](#)

- `show evpn instance instance-name extensive`
- `show evpn instance esi-info esi value`

These commands display detailed information about the EVPN instance, including DF election preferences and the current DF status. The ESI info command provides insights into the current DF and backup forwarders along with their preference values and non-revertive status.

By mastering these commands, you can effectively implement and manage the DF Election feature, ensuring a robust and efficient network environment.

DF Election for Virtual Switch

The virtual switch permits multiple bridge domains in one EVPN instance (EVI). It also accommodates both trunk and access ports. You can configure flexible Ethernet services on the port, enabling different VLANs on a single port to become part of different EVIs.

See the following for more information:

- ["Understanding Flexible Ethernet Services Support With EVPN-VXLAN" on page 654](#)
- *encapsulation (interfaces)*

The DF election for virtual switch depends on the following:

- Port mode—Sub-interface, trunk interface, and access port
- EVI mode—Virtual switch with EVPN and EVPN-EVI

In the virtual switch, multiple Ethernet tags can be associated with a single EVI, wherein the numerically lowest Ethernet tag value in the EVI is used for the DF election.

Handling Failover

IN THIS SECTION

- [Unicast Traffic | 200](#)
- [BUM Traffic | 200](#)

A failover can occur when:

- The DF PE router loses its DF role.
- There is a link or port failure on the DF PE router.

On losing the DF role, the PE router puts the customer-facing interface on the DF into the blocking state.

A link or port failure triggers a DF election process, which results in the BDF PE router's election as the DF. At that time, unicast traffic and BUM flow of traffic will be affected as follows:

Unicast Traffic

- **CE to Core**—The CE device continues to flood traffic on all the links. The previous BDF PE router changes the EVPN multihomed status of the interface from the blocking state to the forwarding state, and traffic is learned and forwarded through this PE router.
- **Core to CE**—The failed DF PE router withdraws the Ethernet autodiscovery route per Ethernet segment and the locally-learned MAC routes, causing the remote PE routers to redirect traffic to the BDF.



NOTE: The transition of the BDF PE router to the DF role can take some time, causing the EVPN multihomed status of the interface to continue to be in the blocking state, resulting in traffic loss.

BUM Traffic

- **CE to Core**—All the traffic is routed toward the BDF.
- **Core to CE**—The remote PE routers flood the BUM traffic in the core.

RELATED DOCUMENTATION

[EVPN Multihoming Overview](#)

Understanding Automatically Generated ESIs in EVPN Networks

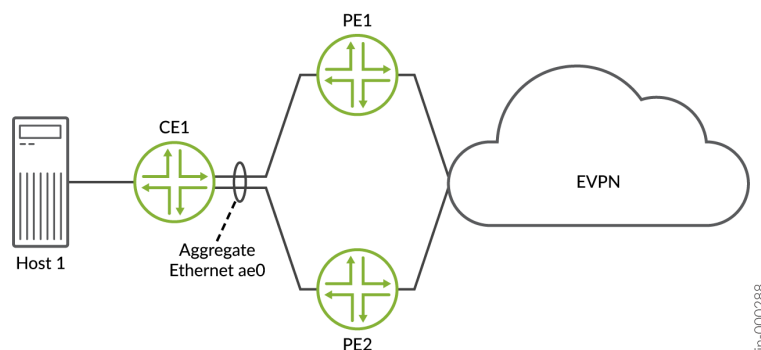
IN THIS SECTION

- [Benefits | 202](#)
- [Automatic ESI Configuration | 202](#)
- [Method 1 Sample Configuration—Automatic ESI on An Aggregated Ethernet Interface | 204](#)

- Method 2 Sample Configuration—Automatic ESI On Aggregated Ethernet Logical Interfaces | 205
- Method 3 Sample Configuration—Manual ESI on Aggregated Ethernet Interface and Automatic ESI on Logical Interfaces | 206
- Other Methods to Auto-Derive the ESI | 208
- ESI Value Format | 208
- Configuring Type 1 and Type 3 Automatic Derivation | 212

Starting with Junos OS Release 18.4R1, you can configure aggregated Ethernet interfaces and aggregated Ethernet logical interfaces to automatically derive Ethernet segment identifiers (ESIs) from the Link Aggregation Control Protocol (LACP) configuration. [Figure 16 on page 201](#) shows a simple multihomed network with a customer edge (CE) device multihomed to two provider edge (PE) devices with Link Aggregation Control Protocol (LACP). This feature automatically derives the ESI from the system ID and the administrative key on the local PE device in the LACP link (actor). Starting in Junos OS Release 22.2R1, we support other automatic ESI derivation methods. See ["Other Methods to Auto-Derive the ESI" on page 208](#).

Figure 16: Simple Multihomed Topology



We support this feature on multihomed devices:

- In all-active mode in an EVPN-VXLAN overlay network.
- In active-standby or all-active mode in an EVPN-MPLS overlay network.

This topic includes the following information:

Benefits

- Frees you from manually configuring ESIs in large EVPN-VXLAN and EVPN-MPLS overlay networks,
- Eliminates the possibility of inadvertently configuring the same ESI for multiple Ethernet segments.

Automatic ESI Configuration

In general, you can configure ESIs on aggregated Ethernet interfaces and aggregated Ethernet logical interfaces using the following methods:

- **Method 1**—You can configure automatic ESI on an aggregated Ethernet interface on which LACP is enabled. In this case, an ESI is generated, and that particular ESI is assigned to all logical interfaces on the aggregated Ethernet interface.
- **Method 2**—You can configure automatic ESI on one or more logical interfaces of an aggregated Ethernet interface on which LACP is configured. In this case, an ESI is generated for each logical interface on which the feature is enabled and assigned to that particular logical interface.
- **Method 3**—On an aggregated Ethernet interface on which LACP is enabled, you can manually configure an ESI using the `esi identifier` configuration statement at the `[edit interfaces aeX]` hierarchy level. On one or more logical interfaces on that particular aggregated Ethernet interface, you can configure automatic ESI. In this case, an ESI is generated for each logical interface on which the feature is enabled and assigned to that particular logical interface.

[Table 15 on page 202](#) outlines the automatic ESI configuration options, how to configure each option, and how the ESI is derived for each option.

Table 15: Automatic ESI Configuration Options

Configuration Options	How to Configure Automatic ESI	How ESI Is Derived
Configure automatic ESI on an aggregated Ethernet interface on which LACP is enabled.	Include the auto-derive configuration statement at the <code>[edit interfaces aeX esi]</code> hierarchy level, with the <code>lacp</code> option under the auto-derive statement.	The ESI is derived from the configured values for the <code>system-id</code> and <code>admin-key</code> configuration statements at the <code>[edit interfaces aeX aggregated-ether-options lacp]</code> hierarchy level.

Table 15: Automatic ESI Configuration Options (*Continued*)

Configuration Options	How to Configure Automatic ESI	How ESI Is Derived
Configure automatic ESI on an aggregated Ethernet logical interface. LACP is enabled on the parent aggregated Ethernet interface.	Include the auto-derive configuration statement at the [edit interfaces aeX unit <i>logical-unit-number</i> esi] hierarchy level, with the lacp option under the auto-derive statement.	The ESI is derived from the configured values for the system-id configuration statement at the [edit interfaces aeX aggregated-ether-options lacp] hierarchy level and the vlan-id configuration statement at the [edit interfaces aeX unit <i>logical-unit-number</i>] hierarchy level. If a logical interface is configured as a trunk interface (interface-mode trunk) and has a VLAN ID list associated with it, the lowest VLAN ID value is used.

**NOTE:**

Starting in Junos OS Release 22.2R1, the lacp configuration statement at the [edit interfaces *interface-name* esi auto-derive] hierarchy level and the [edit interfaces *interface-name* unit *logical-unit-number* esi auto-derive] hierarchy level has been renamed. The new statement name at those hierarchy levels is lacp-pe-system-id-and-admin-key. The CLI aliases the old name to the new name, so you can still commit configurations that use the old statement name lacp, although you only see the new name in the CLI.

When implementing the automatic ESI feature, keep the following in mind:

- In your EVPN-VXLAN or EVPN-MPLS overlay network, you can configure automatic ESI using a mix of method 1, 2, and 3 configuration use cases.
- If a local device is multihomed to two remote devices, we recommend that the aggregated Ethernet and aggregated Ethernet logical interfaces by which the three devices are multihomed have the automatic ESI feature enabled. If the automatic ESI feature is not enabled on one of the interfaces, that interface is not considered during the designated forwarder (DF) election process.
- The automatically generated ESI is supported in both modulo operation- and preference-based DF election processes.
- If you enable the automatic ESI feature and manually configure an ESI on a particular aggregated Ethernet interface or aggregated Ethernet logical interface, you will receive an error when you try to commit the configuration.

- If you enable the automatic ESI feature on an aggregated Ethernet interface and one or more of the logical interfaces on that particular aggregated Ethernet interface, you will receive an error when you try to commit the configuration.

Method 1 Sample Configuration—Automatic ESI on An Aggregated Ethernet Interface

The following example shows the configuration of automatic ESI on aggregated Ethernet interface ae0, which is multihomed in all-active mode. This configuration results in an ESI that is automatically generated based on the LACP configuration and assigned to logical interfaces ae0.0, ae0.100, ae0.101, and ae0.102.

```
user@mx240> show configuration interfaces
ae0
flexible-vlan-tagging;
encapsulation flexible-ethernet-services;
esi {
    auto-derive { ### Automatic ESI configuration.###
        lacp; ### Automatic ESI configuration.###
    }
    all-active;
}
aggregated-ether-options {
    lacp {
        active;
        system-id 00:00:00:00:11:01; ### ESI derived from this value.###
        admin-key 40; ### ESI derived from this value.###
    }
}
unit 0 {
    encapsulation vlan-bridge;
    vlan-id 10;
}
unit 100 {
    family bridge {
        interface-mode trunk;
        vlan-id-list 100;
    }
}
unit 101 {
    family bridge {
        interface-mode trunk;
        vlan-id-list 101;
```

```

    }
}
unit 102 {
    family bridge {
        interface-mode trunk;
        vlan-id-list 102;
    }
}
...

```

Method 2 Sample Configuration—Automatic ESI On Aggregated Ethernet Logical Interfaces

The following example shows the configuration of automatic ESI on aggregated Ethernet logical interfaces ae0.0, ae0.100, ae0.101, and ae0.102, all of which are multihomed in all-active mode. This configuration results in ESIs that are automatically generated based on the LACP and VLAN ID configurations and assigned to each respective logical interface.

```

user@mx240> show configuration interfaces
ae0
flexible-vlan-tagging;
encapsulation flexible-ethernet-services;
aggregated-ether-options {
    lacp {
        active;
        system-id 00:00:00:00:11:01; ### ESI derived from this value.###
    }
}
unit 0 {
    encapsulation vlan-bridge;
    vlan-id 10; ### ESI derived from this value.###
    esi {
        auto-derive { ### Automatic ESI configuration.###
            lacp; ### Automatic ESI configuration.###
        }
        all-active;
    }
}
unit 100 {
    esi {
        auto-derive { ### Automatic ESI configuration.###

```

```

        lacp; ### Automatic ESI configuration.###
    }
    all-active;
}
family bridge {
    interface-mode trunk;
    vlan-id-list 100; ### ESI derived from this value.###
}
}
unit 101 {
    esi {
        auto-derive { ### Automatic ESI configuration.###
            lacp; ### Automatic ESI configuration.###
        }
        all-active;
    }
    family bridge {
        interface-mode trunk;
        vlan-id-list 101; ### ESI derived from this value.###
    }
}
unit 102 {
    esi {
        auto-derive { ### Automatic ESI configuration.###
            lacp; ### Automatic ESI configuration.###
        }
        all-active;
    }
    family bridge {
        interface-mode trunk;
        vlan-id-list 102; ### ESI derived from this value.###
    }
}
...

```

Method 3 Sample Configuration—Manual ESI on Aggregated Ethernet Interface and Automatic ESI on Logical Interfaces

The following example shows the manual configuration of an ESI on aggregated Ethernet interface ae0, and the configuration of automatic ESI on logical interfaces ae0.0, ae0.100, ae0.101, and ae0.102. All interfaces are multihomed in all-active mode. This configuration results in ESI

00:11:22:33:44:55:66:77:88:99 being assigned to ae0, and ESIs that are automatically generated based on the LACP and VLAN ID configurations and assigned to the respective logical interfaces.

```

user@mx240> show configuration interfaces
ae0
flexible-vlan-tagging;
encapsulation flexible-ethernet-services;
esi 00:11:22:33:44:55:66:77:88:99; ### Manual ESI configuration.###
aggregated-ether-options {
    lacp {
        active;
        system-id 00:00:00:00:11:01; ### Logical interface ESI derived from this value.###
    }
}
unit 0 {
    encapsulation vlan-bridge;
    vlan-id 10; ### Logical interface ESI derived from this value.###
    esi {
        auto-derive { ### Automatic ESI configuration.###
            lacp; ### Automatic ESI configuration.###
        }
        all-active;
    }
}
unit 100 {
    esi {
        auto-derive { ### Automatic ESI configuration.###
            lacp; ### Automatic ESI configuration.###
        }
        all-active;
    }
    family bridge {
        interface-mode trunk;
        vlan-id-list 100; ### Logical interface ESI derived from this value.###
    }
}
unit 101 {
    esi {
        auto-derive { ### Automatic ESI configuration.###
            lacp; ### Automatic ESI configuration.###
        }
        all-active;
    }
}

```



```

    }
    family bridge {
        interface-mode trunk;
        vlan-id-list 101; ### Logical interface ESI derived from this value.###
    }
}
unit 102 {
    esi {
        auto-derive { ### Automatic ESI configuration.###
            lacp; ### Automatic ESI configuration.###
        }
        all-active;
    }
    family bridge {
        interface-mode trunk;
        vlan-id-list 102; ### Logical interface ESI derived from this value.###
    }
}
...

```

Other Methods to Auto-Derive the ESI

Network operators might not be managing all the devices in an EVPN network but they must still ensure that the ESI is unique. Juniper provides other methods to automatically derive an ESI.

Include the following statements at the `[edit interfaces aeX aggregated-ether-options lacp]` hierarchy level.

- `type-1-lacp`—Type 1 uses the system ID and the administrative key on the remote CE device in the LACP link (partner).
- `type-3-system-mac`—Type 3 uses the mac and local-discriminator values that are configured on the PE device.

We support configuring type 1 and type 3 auto-derived ESI on multihomed devices in the all-active mode in both EVPN-VXLAN and EVPN-MPLS networks.

ESI Value Format

When the automatic ESI feature is configured, the aggregated Ethernet and aggregated Ethernet logical interfaces derive the ESIs from various configurations on the aggregated Ethernet interface. The 10-byte ESI value for the different auto-derived ESI options is shown in [Figure 17 on page 209](#) and described in [Table 16 on page 210](#).

Figure 17: ESI Value Format

	ESI Type	ESI Value		
	1 octet	6 octets	2 octets	1 octet
lacp-pe-system-id-and-admin-key	T	PE LACP System MAC address	PE Port Key	0
type-1-lacp	T	CE LACP System MAC address	CE Port Key	0
type-3-system-mac	T	System MAC	Local Discriminator	

jn-000289

Table 16: ESI Value Format

Auto-Derived Option	T (ESI Type)	ESI Value
lacp-pe-system-id-and-admin-key	Type 1—The first octet is encoded as 0x01.	<p>The next eight octets are generated from the LACP configuration on the local PE device (actor):</p> <ul style="list-style-type: none"> • The LACP system MAC address includes the system-id configuration statement at the [edit interfaces aeX aggregated-ether-options lacp] hierarchy level. • The Port Key field consists of a two octet field from either: <ul style="list-style-type: none"> • The value of the admin-key configuration statement at the [edit interfaces aeX aggregated-ether-options lacp] hierarchy level. • The value of the vlan-id configuration statement at the [edit interfaces aeX unit logical-unit-number] hierarchy level. <p>The last octet is encoded as 0x00.</p>

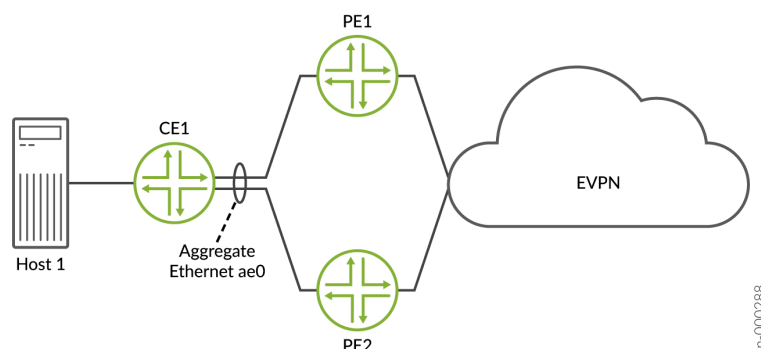
Table 16: ESI Value Format (*Continued*)

Auto-Derived Option	T (ESI Type)	ESI Value
type-1-lacp	Type 1—The first octet is encoded as 0x01.	<p>The next eight octets are generated from the LACP configuration on the remote CE device (partner):</p> <ul style="list-style-type: none"> • The LACP system MAC address includes the system-id configuration statement at the [edit interfaces aeX aggregated-ether-options lacp] hierarchy level. • The Port Key field consists of a two octet field from either: <ul style="list-style-type: none"> • The value of the admin-key configuration statement at the [edit interfaces aeX aggregated-ether-options lacp] hierarchy level. • The value of the vlan-id configuration statement at the [edit interfaces aeX unit <i>logical-unit-number</i>] hierarchy level. <p>NOTE: The system-id and admin-key are optional on the CE device. If you do not configure these options, the device will derive the ESI from the interface of the CE device and a system generated number.</p> <p>The last octet is encoded as 0x00.</p>
type-3-system-mac	Type 3—The first octet is encoded as 0x03.	<p>The remaining octets are generated from the values of the mac and local-discriminator values at the [edit interfaces aeX esi auto-derive] hierarchy level.</p>

Configuring Type 1 and Type 3 Automatic Derivation

Figure 18 on page 212 show a simple multihomed topology with PE1 and PE2 multihomed to CE1.

Figure 18: Multihomed Topology



Configure Automatic Derivation Using Type 1

The following example shows a type 1 configuration on the aggregated Ethernet interfaces on CE1 and one PE device. The result is an ESI that is derived from the LACP configuration on CE1.

```
user@pe1 show configuration interfaces ae0
esi {
  auto-derive {
    type-1-lacp;
  }
  all-active;
}
aggregated-ether-options {
  minimum-links 1;
  lacp {
    active;
    system-id 00:11:22:33:44:55; ### system-id must be identical on PE2
    admin-key 1234; ### admin-key must be identical on PE2
  }
}
unit 0 {
  family ethernet-switching {
    interface-mode trunk;
```

```

        vlan {
            members [ v100 v200 ];
        }
    }
}
...

```

```

user@ce1# show interfaces ae0
flexible-vlan-tagging;
encapsulation flexible-ethernet-services;
aggregated-ether-options {
    lacp {
        active;
        system-id aa:bb:cc:dd:ee:22; ### ESI derived from this value
        admin-key 1234;   ### ESI derived from this value
    }
}
unit 1 {
    vlan-id 100;
    family inet {
        address 10.1.1.200/24;
    }
}
unit 2 {
    vlan-id 200;
    family inet {
        address 10.1.2.200/24;
    }
}
}

```

Configure Automatic Derivation Using Type 3

The following example shows the configuration of automatic ESI on the aggregated Ethernet interface that is using the locally configure system mac and local discriminator options on the PE1 device.

```

user@pe1# show interfaces ae0
esi {
    auto-derive {
        type-3-system-mac {
            mac 00:aa:bb:cc:dd:ee; ### ESI derived from this value
        }
    }
}

```

```
        local-discriminator 1000; ### ESI derived from this value
    }
}
all-active;
}
aggregated-ether-options {
    minimum-links 1;
    lacp {
        system-id 00:11:22:33:44:55;
        admin-key 1234;
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ v100 v200 ];
        }
    }
}
```

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
22.2R1	Starting in Junos OS Release 22.2R1, you can configure the ESI to be derived from the system ID and the administrative key on the remote CE device in the LACP link or from the mac and local discriminator values on the PE device.
18.4R1	Starting with Junos OS Release 18.4R1, you can configure aggregated Ethernet interfaces and aggregated Ethernet logical interfaces to automatically derive Ethernet segment identifiers (ESIs) from the Link Aggregation Control Protocol (LACP) configuration.

RELATED DOCUMENTATION

EVPN Multihoming Overview 162
<i>auto-derive</i>

Easy EVPN LAG (EZ-LAG) Configuration

SUMMARY

With the EZ-LAG configuration feature, you can easily configure a small Ethernet virtual private network (EVPN) for a pair of peer provider edge (PE) devices that have attached multihomed or single-homed servers. You use a simplified Junos OS CLI statement hierarchy, and a built-in commit script generates the full configuration.

IN THIS SECTION

- [Benefits of Using Easy EVPN LAG Configuration | 216](#)
- [Easy EVPN LAG Configuration Overview | 217](#)
- [Simplified CLI Statements and Parameters to Generate the Configuration | 220](#)
- [Platform-Specific Behavior for EZ-LAG Generated Configurations | 255](#)
- [Easy EVPN LAG Configuration with Multihomed Servers | 256](#)
- [Add Configuration for a New Multihomed Server | 268](#)
- [Add a New VLAN and IRB Interfaces | 270](#)
- [Add a New Single-homed Server | 273](#)
- [Use OSPF for the Underlay Configuration | 274](#)
- [Use IBGP for the Overlay Configuration | 275](#)

EVPN fabric PE devices reliably handle traffic to and from attached multihomed end devices by grouping the multihoming links into an EVPN Ethernet segment with an identifier (ESI). You configure the links participating in the Ethernet segment into a link aggregation group (LAG), so we call the set of multihomed links an *ESI LAG*. In this document, we call the PE devices *peer PE devices* when they link to multihomed end devices with the same ESI. The PE devices might also have one or more links in a LAG to single-homed end devices, although an EVPN PE device doesn't need to handle single-homed links as an Ethernet segment.

The end devices might be hosts or servers directly attached to the PE devices, or might be customer edge (CE) devices with attached end hosts or servers. For simplicity, in this document we refer to the end devices collectively as *servers*.

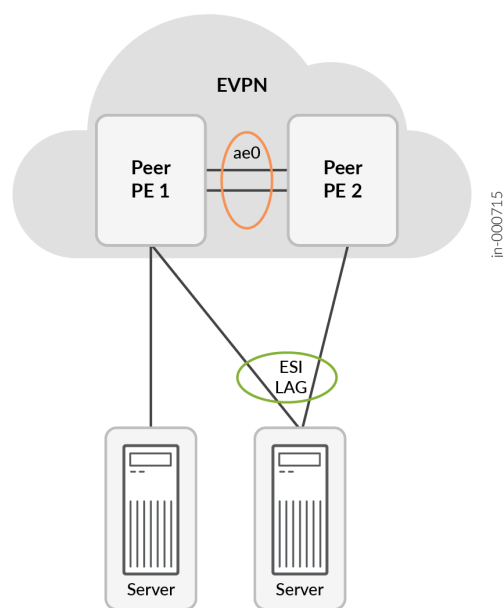
EVPN Multihoming provides redundancy and load balancing between the two switches, and provides a loop-free Layer 2 (L2) network without running STP. However, configuring EVPN multihoming can be complex, and involves configuring many statements correctly across the PE devices in the fabric. With this easy EVPN LAG configuration feature (also called the EZ-LAG feature), we provide a simplified

configuration statement hierarchy and a built-in commit script that you can use to set up EVPN multihoming. This feature makes it easy to migrate a multichassis link aggregation group (MC-LAG) topology to a standards-based EVPN-VXLAN multihoming model.

See [Figure 19 on page 216](#). The supported EVPN topology includes:

- Two peer PE devices connected back-to-back with an aggregated Ethernet interface bundle
- An EVPN fabric with VXLAN encapsulation interconnecting the peer PE devices
- ESI LAG configurations between the peer PE devices and one or more attached multihomed servers
- Connections between a peer PE device and one or more attached single-homed servers

Figure 19: Setups Supported for Easy EVPN LAG Configuration



This feature requires a software license. See the [Juniper Licensing User Guide](#) for details on the Juniper Flex Software License model and Juniper Agile Licensing, and the available licenses for the EZ-LAG feature on supported platforms.

Benefits of Using Easy EVPN LAG Configuration

- **Automated configuration:** You can easily set up a small EVPN fabric, including ESI-LAGs for multihomed attached servers, without using a network controller. You provide some essential parameters through the CLI, and the device's built-in commit script transforms them into a complete EVPN fabric configuration.

- **Configuration flexibility:** You have the flexibility to use the basic and default configuration options, and also customize the generated configuration. You can override many of the commit script default behaviors, and manually configure those elements.
- **Simplified topology migration:** You can use this feature to easily migrate a multichassis link aggregation group (MC-LAG) topology to a simple EVPN-VXLAN fabric.

Easy EVPN LAG Configuration Overview

IN THIS SECTION

- Overview of the `[edit services evpn]` Statement Hierarchy | 217
- Built-in Commit Script | 219

The easy EVPN LAG configuration feature operates at configuration commit time. It consists of:

- A set of configuration statements at the `[edit services evpn]` hierarchy level with which you provide the parameters to set up the prescribed EVPN fabric.
- A built-in commit script that processes the simplified configuration elements at commit time (before the standard Junos OS configuration validity checks) and generates a corresponding EVPN fabric configuration. See ["Built-in Commit Script" on page 219](#) for how to enable the EZ-LAG commit script.

You configure a few `[edit services evpn]` statements to provide the minimal, required set of parameters to configure the EVPN fabric and the links to the attached servers.

When you commit the `[edit services evpn]` configuration statements, the device invokes the commit script that generates the corresponding EVPN configuration using the parameters you provided. The commit script derives some configuration statements specific to the device on which you commit the simplified configuration. The commit process validates the configuration and generates warning messages for missing required parameters or misconfigured parameters.

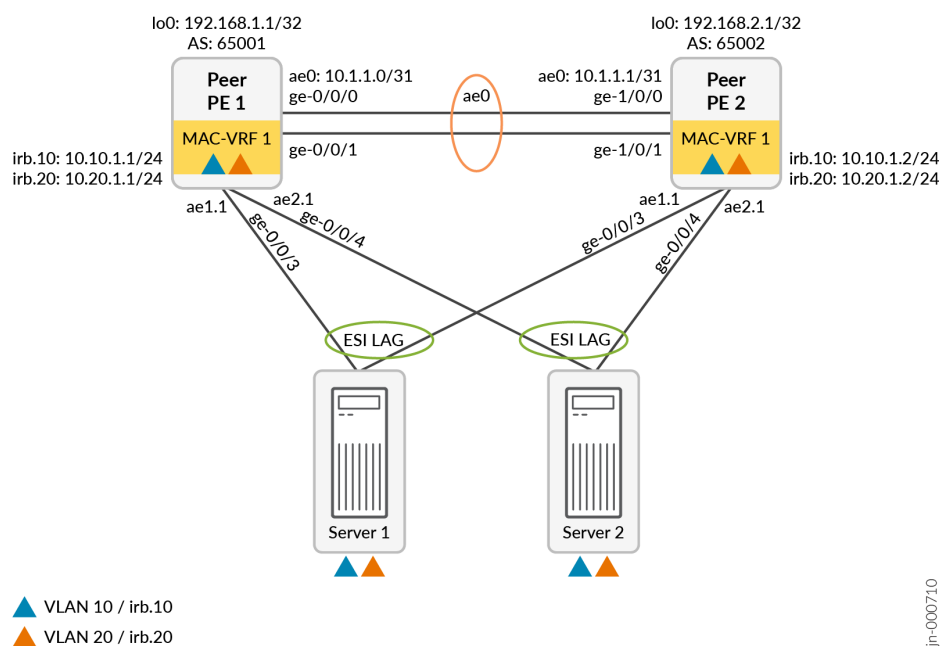
See *Commit Script Overview* and *How Commit Scripts Work* for more on commit scripts.

At any time, you can easily include a few more `[edit services evpn]` statements to add elements to the existing configuration, such as new VLANs or connections to new servers.

Overview of the `[edit services evpn]` Statement Hierarchy

You can specify the minimal set of parameters in just a few configuration statements. For example, consider the following small EVPN-VXLAN fabric with two connected multihomed servers that each host two VLANs:

Figure 20: Small Example EVPN Topology



The following few easy EVPN LAG configuration commands provide the minimal required parameters to configure the EVPN fabric from [Figure 20 on page 218](#) on device Peer PE 1. When you commit these configuration commands, the commit script generates a corresponding full default configuration that might contain 80 to 100 configuration commands:

```
set services evpn device-attribute peer-id 1 system-id 10:14:15:16:17:10 peer-to-peer peer-
subnet inet 10.1.1.0/31 interface-name [ ge-0/0/0 ge-0/0/1 ]
set services evpn device-attribute loopback peer1-subnet 192.168.1.1/32 peer2-subnet
192.168.2.1/32
set services evpn evpn-vxlan irb IRB_10 vlan-id 10 subnet-address inet 10.10.1.1/24
set services evpn evpn-vxlan irb IRB_20 vlan-id 20 subnet-address inet 10.20.1.1/24
set services evpn evpn-vxlan server SERVER_1 esi-lag-id 1 vlan-id-list [ 10 20 ] interface
ge-0/0/3
set services evpn evpn-vxlan server SERVER_2 esi-lag-id 2 vlan-id-list [ 10 20 ] interface
ge-0/0/4
```

See ["Easy EVPN LAG Configuration with Multihomed Servers"](#) on page 256 for the full configuration the commit script generates.

The `[edit services evpn]` command hierarchy has many additional options that give you flexibility to customize the default generated configuration as needed. The commit script uses some of the elements you provide in the simplified configuration to automatically derive other parameters in the generated

configuration. By default, the commit script also generates configuration commands for other common features in an EVPN fabric, such as loop detection and storm control on the interfaces from PE devices to servers.

You can include options to:

- Tell the commit script not to automatically derive some parameters in the generated configuration.
- Override some of the commit script default values and behaviors.
- Apply configuration groups to statements the commit script generates.

With any of those options, if the commit script requires any such elements to provide a valid configuration, you must configure those elements manually.

See ["Simplified CLI Statements and Parameters to Generate the Configuration" on page 220](#) for details on the parameters each statement and option provide to the commit script, and how those parameters affect the generated configuration.

Built-in Commit Script

The commit script for this feature, `services_evpn_commit_script.py`, is enabled by default on supported platforms.

Because the commit script applies transient configuration changes, for the commit script to work, you must also set the system scripts *allow-transients* option, as follows:

- In releases before Junos OS Release 24.2R1, you can set the *allow-transients* option only at the global level, and the option will apply to any configured commit scripts, as follows:

```
set system scripts commit allow-transients
```

- Starting in Junos OS Release 24.2R1, if needed, you can alternatively set the *allow-transients* option at the individual script level so the option applies only to the EZ-LAG commit script, as follows:

```
set system scripts commit file services_evpn_commit_script.py allow-transients
```

- Starting in Junos OS Release 24.4R1, we include the *allow-transients* option at the EZ-LAG commit script level in the default configuration, so you don't need to explicitly set this option anymore.

Simplified CLI Statements and Parameters to Generate the Configuration

IN THIS SECTION

- [Default Generated Configuration Overview | 220](#)
- [Easy EVPN LAG Configuration Statements and Options | 221](#)
- [How the Commit Script Uses Easy EVPN LAG Configuration Elements in the Generated Configuration | 241](#)

This section gives details about the set of configuration statements at the `[edit services evpn]` hierarchy level. We also show how the commit script maps the simplified configuration parameters to generate the full configuration.

Default Generated Configuration Overview

The default generated configuration includes:

- Underlay and overlay peering using external BGP (EBGP).
- A default MAC-VRF EVPN instance named `__SERVICES_EVPN_EVPN_VXLAN_MAC-VRF_1`, with VLAN-aware service type and VXLAN encapsulation, which hosts the VLANs you specify.

You can also configure additional MAC-VRF instances and their member VLANs.

- IRB interfaces with the specified IPv4 (or IPv6) addresses for routing between VLANs.

By default, the commit script derives a virtual gateway address for each IRB interface as the highest configurable address in the specified IRB subnet address range.

The commit script also assigns a default virtual gateway MAC address of `00:00:5e:00:01:01` if you don't specify that parameter.

- ESI LAG connections to multihomed servers, and LAG links to single-homed servers.

The commit script assigns aggregated Ethernet interfaces to those links as trunk interfaces with VLAN tagging, flexible Ethernet services encapsulation, and LACP enabled.

- Lightweight loop detection on the server-facing aggregated Ethernet interfaces associated with the EVPN MAC-VRF instance, with loop detection action `interface down`. See ["EVPN-VXLAN Lightweight Leaf to Server Loop Detection" on page 657](#) for more on this feature.
- Storm control on server-facing interfaces.

The commit script generates a default storm control profile named `__SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL`, and assigns it to the each server-facing interface. See ["MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment" on page 687](#) for more on the storm control feature.

- Other options commonly included in EVPN network configurations to improve routing table convergence if the aggregated Ethernet interface connections become unstable, such as:
 - `no-core-isolation` at the `[edit (routing-instances name) protocols evpn]` hierarchy level . See ["Understanding When to Disable EVPN-VXLAN Core Isolation" on page 514](#).
 - `no-mac-flush-on-aa-ae-down` at the `[edit protocols l2-learning platform-parameters]` hierarchy level.

See an example easy LAG configuration and the corresponding generated configuration here: ["Easy EVPN LAG Configuration with Multihomed Servers" on page 256](#).

The next sections describe the configuration elements in detail.

Easy EVPN LAG Configuration Statements and Options

[Table 17 on page 221](#) shows the statements and options at the `[edit services evpn]` configuration statement hierarchy level. The commit script requires a minimum set of statements to have enough information to generate a working configuration. The table indicates which statements are mandatory. You can optionally include other statements to change the commit script behavior and override default values as needed for your topology and configuration preferences.

The commit script uses some of the elements you provide in the simplified configuration to automatically derive other parameters in the generated configuration. By default, the commit script also generates configuration statements for supporting common features in an EVPN fabric, such as loop detection and storm control on links to servers.

Table 17: `[edit services evpn]` Statements and Purpose

Statement and Child Statements	Options	Purpose	Mandatory?
defaults-override If you configure any of these options, the script will not automatically derive those parameters from the related <code>[edit services evpn]</code> statements you provided, or it will skip generating configuration for the related element. You must manually configure those elements instead.			

Table 17: [edit services evpn] Statements and Purpose *(Continued)*

Statement and Child Statements	Options	Purpose	Mandatory?
	no-aggregate-device-count-config	Don't generate the default aggregated Ethernet (ae <i>n</i>) interface count configuration statement.	No
	no-loop-detect-config	Don't generate lightweight loop detection configuration statements.	No
	no-platform-defaults-config	Don't generate statements for default platform-specific options.	No

Table 17: [edit services evpn] Statements and Purpose *(Continued)*

Statement and Child Statements	Options	Purpose	Mandatory?
	no-policy-and-routing-options-config	<p>Don't generate default routing and policy options configuration statements.</p> <p>If you specify this option, you must manually provide configuration for a policy-statement named EXPORT-LOO at the [edit policy-options policy-statement] hierarchy level. The commit script assigns a policy by that name in the default generated EBGp underlay configuration.</p> <p>Or, you must include the no-underlay-config option to override generating the default underlay configuration, and manually configure the underlay peering statements you want instead.</p>	No
	no-storm-control-config	Don't generate configuration for storm control.	No
	no-overlay-bgp-config	Don't generate BGP configuration for the overlay.	No
	no-underlay-config	Don't generate configuration for the underlay peering.	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
<i>device-attribute</i> With these statements, you define parameters to configure the peer PE devices, including the peer interconnection interfaces, underlay peering, and overlay peering between them to form the EVPN fabric. The commit script configures underlay and overlay peering with external BGP (EBGP) by default. You can optionally specify that the commit script generate configuration for one of the other available underlay or overlay protocols. Or, you can alternatively set the no-overlay-bgp-config option or the no-underlay-config option (at the [edit services evpn defaults-override] hierarchy level), and manually configure the overlay BGP peering statements or underlay peering statements you want.			
	peer-id <i>peer-id</i>	Number (1-2) that identifies the PE device (compared to its peer PE device) in the fabric. The commit script uses this value to derive some values that must be unique across the peer PE devices in the generated configuration.	Yes
	system-id <i>system-id</i>	Peer PE device's system ID (MAC address format). The commit script needs this value to derive configuration to set up LACP on ESI LAG interfaces.	Yes

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
(device-attribute) loopback	peer1-subnet <i>peer1-subnet</i> peer2-subnet <i>peer2-subnet</i>	PE device loopback IPv4 subnet addresses for peer PE 1 and peer PE 2 From this parameter, the commit script configures the loopback interface subnet addresses for the peer PE devices, and the router ID for each device.	Yes
(device-attribute) <i>peer-to-peer</i>	<i>peer-subnet</i> (inet inet6) <i>subnet-address</i>	IPv4 or IPv6 subnet address for the interfaces that connect the PE device to its peer PE device. You can specify this option with either the inet or the inet6 option, or both.	Yes
	<i>peer-subnet</i> interface-name [<i>interface-name</i> ...]	Name or names of the interfaces that connect the PE device to its peer PE device.	Yes
	overlay-connectivity { ibgp }	Generate a prescribed configuration for overlay peering using internal BGP (IBGP) instead of the default protocol, EBGp.	No
	underlay-connectivity { ospf }	Generate a prescribed configuration for underlay peering using OSPF instead of the default protocol, EBGp.	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
<p><i>evpn-vxlan</i></p> <p>With these statements, you define parameters to configure the peer PE devices to run EVPN with VXLAN encapsulation using one or more VLAN-aware MAC-VRF instances.</p> <p>By default, the commit script enables EVPN-VXLAN in a single MAC-VRF instance named <code>__SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1</code>. If you configure the <code>mac-vrf-instance instance-id</code> option at the <code>[edit services evpn vxlan irb irb-instance instance]</code> hierarchy level, the commit script generates configuration using that <i>instance-id</i> instead. The commit script appends the <i>instance-id</i> to the base string to configure unique MAC-VRF instance names in the generated configuration: <code>__SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_instance-id</code>.</p> <p>If needed, you can configure more than one MAC-VRF instance for tenant traffic separation at L2.</p> <p>These statements also provide the parameters to set up the ESI-LAGs to multihomed servers or LAGs to single-homed servers, including the IRB interfaces and the associated VLANs. You can optionally specify to configure DHCP relay for the IRB instances under this stanza as well.</p>			
<i>dhcp-relay</i>	<i>name</i>	DHCP relay group name.	No
	<code>dhcp-server-address</code> <i>dhcp-server-address</i>	IP address of the DHCP server to enable the DHCP server relay group.	No
	<code>relay-source</code> <i>DHCP-relay-source-interface</i>	IP address of the DHCP relay source loopback interface. The commit script generates DHCP overrides <code>relay-source</code> statements that use this source address.	No

Table 17: [edit services evpn] Statements and Purpose *(Continued)*

Statement and Child Statements	Options	Purpose	Mandatory?
	<code>vrf-instance</code> <i>vrf-instance-id</i>	<p>VRF instance identifier in this configuration for which you want to configure DHCP relay.</p> <p>If you don't specify a <i>vrf-instance-id</i>, the commit script uses the default VRF instance (inet.0 routing table). Otherwise, the commit script generates configuration for a VRF instance named <code>__SERVICES_EVPN_EVPN_VXLAN_VRF_vrf-instance-id</code>.</p>	No
<i>irb</i>	<i>irb-instance</i>	<p>String to identify an IRB instance and its associated parameters for commit script processing.</p> <p>We recommend choosing IDs that match the parameters of the IRB instance in the generated configuration. For example, use "irb_10" for the IRB instance associated with VLAN 10, which will be irb.10 in the generated configuration.</p>	Yes
	<code>apply-config-groups</code> <i>config-groups</i>	Apply the specified configuration groups to the configuration the commit script generates from the statements at this hierarchy level.	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
	no-dhcp-relay	<p>Exclude this IRB instance from the DHCP relay configuration.</p> <p>This option applies only if you specified to generate configuration for DHCP relay with the dhcp-relay option at the [edit services evpn expn-vxlan] hierarchy level.</p>	No
	use-anycast-address	<p>Configure the IRB instance with an anycast gateway address.</p> <p>With this option, you must include the anycast-mac <i>mac-address</i> at the [edit services evpn global-parameters] hierarchy level to specify the anycast MAC address to use.</p>	No
	vlan-id <i>vlan-num</i>	<p>VLAN ID associated with the IRB instance.</p> <p>The commit script derives the IRB interface name with a matching logical unit number. For example, the commit script configures irb.10 for <i>vlan-num</i> 10.</p>	Yes

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
(irb <i>irb-instance</i>) instance	mac-vrf-instance <i>instance-id</i>	<p>The MAC-VRF instance ID to which the IRB instance belongs.</p> <p>The commit script generates MAC-VRF instance configuration statements using MAC-VRF instance name <code>__SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_<i>instance-id</i></code>.</p>	<p>No*</p> <p>*If you don't configure this option, the commit script creates a MAC-VRF instance using default <i>instance-id</i> 1.</p>
	vrf-instance <i>instance-id</i>	<p>The Layer 3 (L3) virtual routing and forwarding (VRF) instance to which this IRB instance belongs.</p> <p>If you specify this option, the commit script generates VRF instance configuration statements for VRF instance name <code>__SERVICES_EVPN_EVPN_VXLAN_VRF_<i>instance-id</i></code>.</p>	<p>No*</p> <p>*If you don't configure this option, the commit script uses the default VRF instance (which corresponds to the <code>inet.0</code> routing table).</p>

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
(<i>irb irb-instance</i>) <i>subnet-address</i>	<i>inet (ipv4-subnet-address</i> <i> [ipv4-addr1 ipv4-addr 2</i> <i>...])</i>	<p>IPv4 subnet address or list of subnet addresses for the IRB interface.</p> <p>For simplicity, you can configure the same IRB subnet address on both peer PE devices. By default, the commit script uses the PE peer ID to derive different IRB subnet addresses for each peer device based on the configured <i>ipv4-subnet-address</i> value.</p> <p>Alternatively, you can set the subnet addresses you want on each peer PE device, and include the <i>no-irb-address-auto-derive</i> option so the commit script uses those exact addresses.</p> <p>Also, see <i>subnet-address</i> for details on additional default router-advertisement configuration statements the commit script generates for the IRB interfaces.</p>	<p>Yes*</p> <p>*You must configure at least one subnet address for the IRB instance using either the <i>inet</i> option or the <i>inet6</i> option.</p>

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
	inet6 (<i>ipv6-subnet-address</i> [<i>ipv6-addr1</i> <i>ipv6-addr 2</i> ...])	<p>IPv6 subnet address for the IRB interface.</p> <p>The commit script has the same default behavior with this option as with the inet option above. You can similarly include the no-irb-address-auto-derive option and specify the exact IRB interface IPv6 subnet addresses you want on each device. Also, see <i>subnet-address</i> for details on the additional router-advertisement configuration statements the commit script generates by default when you specify a link-local address with this inet6 option.</p>	<p>Yes*</p> <p>*You must configure at least one subnet address for the IRB instance using either the inet option or the inet6 option.</p>

Table 17: [edit services evpn] Statements and Purpose *(Continued)*

Statement and Child Statements	Options	Purpose	Mandatory?
	no-irb-address-auto-derive	<p>Don't derive IRB subnet addresses in the generated configuration for this IRB instance.</p> <p>You must specify this option if you don't want the commit script default behavior that uses the PE peer ID to derive different IRB subnet addresses for each peer device from the same configured (inet inet6) address.</p> <p>With this option, the commit script uses the exact inet and inet6 subnet addresses you provide.</p> <p>This option is also available at the <code>global-parameters</code> hierarchy level that applies to all IRB instances.</p>	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
	virtual-gateway-v4-address <i>virtual-gateway-v4-address</i>	<p>Virtual gateway IPv4 address for the IRB interface.</p> <p>If you don't include this option, by default the commit script derives a virtual gateway IPv4 address from the inet <i>ipv4-subnet-address</i> value. The commit script uses the highest configurable IPv4 address in that subnet range. For example, subnet address 10.1.1.1/24 generates virtual gateway address 10.1.1.254.</p>	No
	virtual-gateway-v6-address <i>virtual-gateway-v6-address</i>	<p>Virtual gateway IPv6 address for the IRB interface.</p> <p>If you don't include this option, the commit script has the same default behavior to derive the virtual-gateway-v6-address as it does for the virtual-gateway-v4-address above. For example, IPv6 subnet address 2001:db8::10:1:1:1/112 generates virtual gateway IPv6 address 2001:db8::10:1:1:fffe.</p>	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
<i>server</i>	<i>name</i>	<p>A unique name to identify a server connected to the peer PE device, such as SERVER_1 or HostA.</p> <p>This value is used internally to associate server-related parameters with the statements to generate for that server. You might see this server name in description parameters for related statements in the generated configuration.</p>	Yes
	apply-config-groups <i>config-groups</i>	Apply the specified configuration groups to the configuration the commit script generates from the statements at this hierarchy level.	No
	enable-pxe-boot	<p>Enable this connected server to use a preboot execution environment (PXE) boot process.</p> <p>The commit script configures LACP force-up status on the link to the server to ensure the link is UP when the server boots up. The PXE boot process requires this status.</p>	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
	esi-lag-id <i>esi-lag-id</i>	<p>ESI LAG connection ID for this server.</p> <p>The commit script assigns an aggregated Ethernet interface for the connection to this server by adding <i>esi-lag-id</i> to the base interface name ae0 (so for example, <i>esi-lag-id</i> 1 uses ae1).</p>	<p>Yes*</p> <p>*You must specify at least one esi-lag-id <i>OR</i> single-home-id for each server. See the single-home-id option in the next row.)</p>
	single-home-id <i>single-home-id</i>	<p>Single-homed connection ID for the named server.</p> <p>The commit script assigns an aggregated Ethernet interface for the connection to this server by adding <i>single-home-id</i> to the base interface name ae1024 (so for example, <i>single-home-id</i> 1 uses ae1025).</p>	<p>Yes*</p> <p>*You must specify at least one esi-lag-id <i>OR</i> single-home-id for each server. (See the esi-lag-id option in the previous row.)</p>
	interface (<i>interface-name</i> [<i>interface-name</i> ...])	Interface name or list of interface names for the physical links from the peer PE device to this server.	Yes

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
	vlan-id-list [<i>vlan-id-list</i>]	List of VLAN IDs hosted by this MAC-VRF instance and server.	Yes* *You must configure at least one VLAN ID list (with at least one VLAN ID in it) for each server, either at this server <i>name</i> level or at the mac-vrf-instance <i>instance-id</i> level (see next row).
(server <i>name</i>) mac-vrf-instance	<i>instance-id</i>	Identifier for a MAC-VRF instance that hosts this server. If you don't specify this option, the commit script generates configuration that associates these server parameters with the default MAC-VRF instance (named __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1).	No
	vlan-id-list [<i>vlan-id-list</i>]	List of VLAN IDs hosted by this MAC-VRF instance and server.	Yes* *You must configure at least one VLAN ID list (with at least one VLAN ID in it) for each server, either at this MAC-VRF instance level or at the server <i>name</i> level.

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
<i>global-parameters</i> With these statements, you specify parameters for configuration elements that are common across the peer PE devices in the EVPN fabric, such as the default VLAN and anycast gateway addresses. The commit script has default values it uses if you don't configure these options.			
	anycast-mac <i>anycast-mac</i>	Globally use the specified anycast virtual gateway MAC address in the generated configuration.	No
	default-vlan <i>vlan-id</i>	Use this VLAN ID (1-4094) as the default VLAN in the generated configuration. The commit script uses the system default VLAN ID if you don't configure this.	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
	no-irb-address-auto-derive	<p>Don't derive IRB interface subnet addresses in the generated configuration for all IRB instances. Instead, use the exact IPv4 or IPv6 subnet addresses you specify with the subnet-address statement at the [edit services evpn evpn-vxlan irb <i>irb-instance</i>] hierarchy level.</p> <p>Otherwise, by default, the commit script derives the IRB subnet address from the subnet-address (inet inet6) statement based on the PE device peer-id.</p> <p>You can alternatively set this option at the irb <i>irb-instance</i> hierarchy level to apply only to a specific IRB instance.</p>	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
	start-aggregate-ethernet-index <i>num</i>	<p>Use this number as the starting index in aggregated Ethernet interface names for ESI LAG links to attached servers.</p> <p>The commit script starts at ae1 by default for ESI LAG links to servers, and reserves ae0 interface names only for the links between the peer PEs.</p> <p>NOTE: This option does not affect generated interface names for links to single-homed servers. The commit script uses aggregated Ethernet interface names starting at ae1025 for single-homed server links.</p>	No
	virtual-gateway-mac <i>virtual-gateway-mac</i>	<p>Use this virtual gateway MAC address for both IPv4 and IPv6 traffic in the generated configuration.</p> <p>Specify the virtual-gateway option (later in this table) instead of this option if you want to assign different virtual gateway MAC addresses for IPv4 and IPv6 traffic.</p>	No

Table 17: [edit services evpn] Statements and Purpose *(Continued)*

Statement and Child Statements	Options	Purpose	Mandatory?
(global-parameters) mtu	overlay <i>overlay-mtu</i>	Specify the maximum transmission unit (MTU) in bytes to set in the generated configuration for the interfaces used in the overlay peering, instead of the default MTU value. The default MTU value varies by platform and type of interface (or protocol) for which you configure the value.	No
	underlay <i>underlay-mtu</i>	Specify the MTU in bytes to set in the generated configuration for the interfaces used in the overlay peering, instead of the default MTU value. The default MTU value varies by platform and type of interface (or protocol) for which you configure the value.	No
(global-parameters) virtual-gateway	v4-mac <i>v4-mac</i>	Globally use this virtual gateway MAC address for IPv4 traffic in the generated configuration. The commit script uses 00:00:5e:00:01:01 by default if you don't configure this option or the virtual-gateway-mac option (above).	No

Table 17: [edit services evpn] Statements and Purpose (*Continued*)

Statement and Child Statements	Options	Purpose	Mandatory?
	<code>v6-mac v6-mac</code>	<p>Globally use this virtual gateway MAC address for IPv6 traffic in the generated configuration.</p> <p>The commit script uses 00:00:5e:00:02:01 by default if you don't configure this option or the <code>virtual-gateway-mac</code> option (see that row earlier in this table).</p>	No

How the Commit Script Uses Easy EVPN LAG Configuration Elements in the Generated Configuration

Table 18 on page 241 lists configuration elements and default values that the commit script uses or derives from the simplified configuration you provide.

Table 18: Derived and Default Configuration Values

Configuration Element	Default or Derived Value
General or Shared Elements	
Number of aggregated Ethernet interface connected device for set <code>chassis aggregated-devices ethernet device-count num</code> in generated configuration	<p><code>num</code> is 255 by default.</p> <p>(Some platforms might use a different default platform-specific value.)</p>
Peer PE identifier, to associate specified or derived parameters with the corresponding peer PE device	Parameter <code>peer-id</code> is required for each peer PE device (there is no default value).
From set <code>services evpn device-attribute peer-id peer-id</code> .	The <code>peer-id</code> can have value 1 or 2.

Table 18: Derived and Default Configuration Values *(Continued)*

Configuration Element	Default or Derived Value
Peer PE device loopback interface (lo0) address From set services evpn device-attribute : <ul style="list-style-type: none"> • peer-id <i>peer-id</i> • loopback peer1-subnet <i>peer1-subnet</i> or loopback peer2-subnet <i>peer2-subnet</i> 	Use the loopback interface address on peer PE device <i>peer-id</i> from the provided loopback subnet address for that peer ID.
Router ID, for set routing-options router-id <i>router-id</i> in generated configuration From set services evpn device-attribute : <ul style="list-style-type: none"> • peer-id <i>peer-id</i> • loopback peer1-subnet <i>peer1-subnet</i> or loopback peer2-subnet <i>peer2-subnet</i> 	Assigned the same value as the device's loopback interface address.
Load balancing routing options policy for per-flow load balancing	<pre>set policy-options policy-statement pp1b then load-balance per-packet set routing-options forwarding-table export pp1b</pre>
Storm control configuration for each server-facing interface	Storm control configuration is platform-dependent. See "Platform-Specific Behavior for EZ-LAG Generated Configurations" on page 255 for the differences in the generated configuration per platform.
Peer to Peer PE Device Connections You specify a peer PE device ID used internally to track the parameters that apply to that configured EVPN peer PE device as follows: <pre>set services evpn device-attribute peer-id <i>peer-id</i></pre>	
Aggregated Ethernet interface name	ae0

Table 18: Derived and Default Configuration Values (*Continued*)

Configuration Element	Default or Derived Value
<p>Aggregated Ethernet interface address</p> <p>From set services evpn device-attribute:</p> <ul style="list-style-type: none"> • peer-id <i>peer-id</i> • peer-to-peer peer-subnet (inet inet6) <i>subnet-address</i> 	<p>For configuration simplicity, you configure the same <i>subnet-address</i> on both peer PE devices.</p> <p>The commit script derives the interface address based on <i>peer-id</i> as follows:</p> <ul style="list-style-type: none"> • <i>peer-id</i> 1: ae0 address is <i>subnet-address</i>. • Otherwise, ae0 address is <i>subnet-address</i> + 1 (add 1 to the low-order address segment of the subnet range). <p>For example, if the IPv4 <i>subnet-address</i> is 10.0.1.0/31:</p> <ul style="list-style-type: none"> • On peer-id 1, ae0 = 10.0.1.0 • Otherwise (on peer-id 2) ae0 = 10.0.1.1
<p>Aggregated Ethernet interface member physical interface or interfaces</p> <p>From set services evpn device-attribute peer-to-peer peer-subnet interface-name [<i>interface-name ...</i>].</p>	<p>Parameter [<i>interface-name ...</i>] is required; set those interfaces as member links in ae0.</p>
<p>Peer PE Device to Server Connections</p> <p>You specify server names and assign IDs that represent the links to those servers (either multihomed ESI LAG connections or single-homed connections), as follows:</p> <ul style="list-style-type: none"> • set services evpn evpn-vxlan server <i>server-name</i> : <ul style="list-style-type: none"> • esi-lag-id <i>esi-lag-id</i> • single-home-id <i>single-home-id</i> <p>The commit script uses the server names internally to associate configured elements with an attached server, but also displays these names in the description element of related generated configuration statements.</p>	

Table 18: Derived and Default Configuration Values (*Continued*)

Configuration Element	Default or Derived Value
Starting index for assigning aggregated Ethernet interface names for ESI LAG links to multihomed servers	<p>Default is 1 (ae1)</p> <p>To override the default value and assign a different starting index (<i>num</i>), configure:</p> <pre>set service evpn global-parameters start-aggregate-ethernet-index <i>num</i> .</pre> <p>This option affects only the starting index for aggregated Ethernet interfaces to multihomed servers (esi-lag-id option). The commit script uses ae(<i>num</i> + esi-lag-id).</p> <p>This option doesn't affect how the commit script allocates single-homed server links (single-home-id option), which is ae(1024 + single-home-id).</p>
<p>Name of physical interface or interfaces from peer PE device to server (single interface or list of interfaces)</p> <p>From set services evpn evpn-vxlan server <i>server-name</i> interface [<i>interface-name</i> ...]</p>	Required parameter per server, no default
<p>Aggregated Ethernet interface name (ae<i>index</i>) and logical unit (unit <i>num</i>)</p> <p>From set services evpn evpn-vxlan server <i>server-name</i> :</p> <ul style="list-style-type: none"> esi-lag-id <i>esi-lag-id</i> or single-home-id <i>single-home-id</i> mac-vrf-instance <i>instance-id</i> 	<p>Derive as follows based on server link ID and configured MAC-VRF instance ID (or default MAC-VRF instance-id 1):</p> <ul style="list-style-type: none"> For multihomed servers: ae(<i>esi-lag-id</i>) unit <i>instance-id</i> For single-homed servers: ae(1024 + <i>single-home-id</i>) unit <i>instance-id</i>

Table 18: Derived and Default Configuration Values *(Continued)*

Configuration Element	Default or Derived Value
<p>Aggregated Ethernet interface options and member VLANs</p> <p>From set services evpn evpn-vxlan server <i>server-name</i> :</p> <ul style="list-style-type: none"> • esi-lag-id <i>esi-lag-id</i> or single-home-id <i>single-home-id</i> • mac-vrf-instance <i>instance-id</i> • vlan-id-list [<i>vlan-id</i> ...] 	<p>Enable VLAN tagging, flexible Ethernet services encapsulation, and trunk interface mode.</p> <p>Set member VLANs from configured server vlan-id-list</p> <p>For example, for <i>esi-lag-id</i> 1 (interface name ae1) in the default MAC-VRF <i>instance-id</i> 1 with vlan-id-list [10 20], the generated configuration is:</p> <pre> set interfaces ae1 vlan-tagging set interfaces ae1 encapsulation flexible-ethernet-services set interfaces ae1 unit 1 family ethernet-switching interface-mode trunk set interfaces ae1 unit 1 family ethernet-switching vlan members 10 set interfaces ae1 unit 1 family ethernet-switching vlan members 20 </pre>

Table 18: Derived and Default Configuration Values *(Continued)*

Configuration Element	Default or Derived Value
<p>System ID for LACP on PE device to server links (MAC address format)</p> <p>From:</p> <ul style="list-style-type: none"> • <code>set services evpn device-attribute system-id <i>system-id</i></code> • <code>set services evpn evpn-vxlan server <i>server-name</i> :</code> <ul style="list-style-type: none"> • <code>esi-lag-id <i>esi-lag-id</i></code> • <code>single-home-id <i>single-home-id</i></code> 	<p>Derive from <i>system-id</i> per server link by adding the server link ID as follows:</p> <p><i>system-id</i> + <i>esi-lag-id</i></p> <p>OR</p> <p><i>system-id</i> + <i>single-home-id</i></p> <p>For example, if <i>system-id</i> is 10:11:12:13:14:15, then for <i>esi-lag-id</i> 1 (ae1), LACP system ID is 10:11:12:13:14:16 in the generated LACP configuration for that link:</p> <pre>set interfaces ae1 aggregated-ether-options lacp active set interfaces ae1 aggregated-ether-options lacp system-id 10:11:12:13:14:16</pre> <p>NOTE: To skip configuring LACP for a peer-server link, include the following option:</p> <pre>set services evpn evpn-vxlan server <i>server-name</i> no- lacp</pre>
<p>Multihomed server interface ESI LAG configuration</p> <p>From <code>set services evpn evpn-vxlan server <i>server-name</i> esi-lag-id <i>esi-lag-id</i></code></p>	<p>Automatically derive the ESI:</p> <pre>set interfaces ae<i>esi-lag-id</i> esi auto-derive type-1- lacp set interfaces ae<i>esi-lag-id</i> all-active</pre>

Table 18: Derived and Default Configuration Values (Continued)

Configuration Element	Default or Derived Value
<p>Lightweight loop detection configuration on server-facing logical aggregated Ethernet interfaces</p> <p>From <code>set services evpn evpn-vxlan server <i>server-name</i> :</code></p> <ul style="list-style-type: none"> • <code>esi-lag-id <i>esi-lag-id</i> or single-home-id <i>single-home-id</i></code> • <code>mac-vrf-instance <i>instance-id</i></code> • <code>vlan-id-list [<i>vlan-id</i> ...]</code> 	<p>Generates loop-detect enhanced statements with action <code>interface-down</code> for the aggregated Ethernet interfaces connected to each of the configured servers, with the logical unit based on the configured MAC-VRF instance ID or default <i>instance-id</i> 1:</p> <pre>set protocols loop-detect enhanced interface aesi-lag-id.<i>instance-id</i> vlan-id <i>vlan-id</i> set protocols loop-detect enhanced interface aesi-lag-id.<i>instance-id</i> loop-detect-action interface-down set protocols loop-detect enhanced interface aesi-lag-id.<i>instance-id</i> transmit-interval 1s set protocols loop-detect enhanced interface aesi-lag-id.<i>instance-id</i> revert-interval 60</pre> <p>For more on this loop detection feature, see "EVPN-VXLAN Lightweight Leaf to Server Loop Detection" on page 657.</p> <p>NOTE: The commit script generates the loop detection configuration only on platforms that support the EVPN-VXLAN lightweight loop detection feature (see Feature Explorer).</p>
<p>EVPN, VXLAN, and IRB Interface Elements</p> <p>Supported EVPN instance parameters in generated configuration: MAC-VRF instance type, VLAN-aware service type, and VXLAN encapsulation</p>	
<p>MAC-VRF EVPN instance name</p> <p>From:</p> <ul style="list-style-type: none"> • <code>set services evpn evpn-vxlan irb <i>irb-instance</i> instance mac-vrf-instance <i>instance-id</i>.</code> • <code>set services evpn evpn-vxlan server <i>server-name</i> mac-vrf-instance <i>instance-id</i></code> 	<p>If you don't configure one or more <code>mac-vrf-instance <i>instance-id</i></code> statements:</p> <ul style="list-style-type: none"> • The default MAC-VRF instance <i>instance-id</i> is 1. • The default MAC-VRF instance name is <code>__SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1</code>. <p>Otherwise, for each configured <i>instance-id</i>, the MAC-VRF instance name is <code>__SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_<i>instance-id</i></code>.</p>

Table 18: Derived and Default Configuration Values *(Continued)*

Configuration Element	Default or Derived Value
<p>Aggregated Ethernet interface logical unit per MAC-VRF instance</p> <p>From <code>set services evpn evpn-vxlan server <i>server-name</i></code> :</p> <ul style="list-style-type: none"> • <code>mac-vrf-instance <i>instance-id</i></code> • <code>esi-lag-id <i>esi-lag-id</i></code> or <code>single-home-id <i>single-home-id</i></code> 	<p>Derive and configure associated logical interface as follows:</p> <p><code>set routing-instances</code> <code>__SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_<i>instance-id</i></code> <code>interface ae(<i>esi-lag-id</i> <i>single-home-id</i>).<i>instance-id</i></code></p>
<p>Route distinguisher (RD) for EVPN instance</p> <p>From :</p> <ul style="list-style-type: none"> • <code>set services evpn device-attribute :</code> <ul style="list-style-type: none"> • <code>peer-id <i>peer-id</i></code> • <code>loopback peer1-subnet <i>peer1-subnet</i></code> • <code>loopback peer2-subnet <i>peer2-subnet</i></code> • <code>set services evpn evpn-vxlan server <i>server-name</i></code> <code>mac-vrf-instance <i>instance-id</i></code> 	<p>Route distinguisher for each MAC-VRF routing instance is derived from the peer PE device loopback subnet address and the MAC-VRF <i>instance-id</i> as follows:</p> <p><code>peer<<i>peer-id</i>>-subnet:<i>instance-id</i></code></p>
<p>Target extended community for EVPN instance</p> <p>From <code>mac-vrf-instance <i>instance-id</i></code></p>	<p><code>vrf-target:1:<i>instance-id</i></code></p>
<p>VLANs (bridge domains) hosted in the EVPN instance</p> <p>From:</p> <ul style="list-style-type: none"> • <code>set services evpn evpn-vxlan server <i>server-name</i></code> <code>vlan-id-list [<i>vlan-id</i> ...]</code> • <code>set services evpn evpn-vxlan irb <i>irb-instance</i></code> <code>vlan-id <i>vlan-id</i></code> 	<p>VLAN names are <code>SERVICES_EVPN_EVPN_VXLAN_VLAN_<i>vlan-id</i></code></p>

Table 18: Derived and Default Configuration Values *(Continued)*

Configuration Element	Default or Derived Value
<p>IRB interface logical units corresponding to configured server VLANs</p> <p>From:</p> <ul style="list-style-type: none"> • <code>set services evpn evpn-vxlan irb <i>irb-instance</i> vlan-id <i>vlan-id</i></code> • <code>set services evpn evpn-vxlan server <i>server-name</i> vlan-id-list [<i>vlan-id</i> ...]</code> 	<p>IRB interface logical unit is <i>vlan-id</i> in generated configuration for:</p> <ul style="list-style-type: none"> • <code>set interfaces irb unit <i>vlan-id</i> family inet address <i>subnet-address</i> ...</code> • <code>set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_<i>instance-id</i> vlans SERVICES_EVPN_EVPN_VXLAN_VLAN_<i>vlan-id</i> 13-interface irb.<i>vlan-id</i></code>

Table 18: Derived and Default Configuration Values (*Continued*)

Configuration Element	Default or Derived Value
<p>IRB interface address (to configure set interfaces irb unit <i>unit</i> family inet address <i>subnet-address</i> ...)</p> <p>From set services evpn evpn-vxlan irb <i>irb-instance</i> subnet-address (inet inet6) <i>subnet-address</i></p> <p>If you want the generated configuration to have different IRB interface <i>subnet-address</i> values that you explicitly set per peer PE device (instead of the default addresses derived from a common configured <i>subnet-address</i> value), then configure the following option:</p> <ul style="list-style-type: none"> Per-IRB instance: <pre>set services evpn evpn-vxlan irb <i>irb-instance</i> no-irb-instance-auto-derive</pre> Globally (for all IRB instances): <pre>set services evpn global-parameters no-irb-instance-auto-derive</pre> 	<p>Derive different addresses per peer PE device from IPv4 or IPv6 <i>subnet-address</i>.</p> <p>Parameter <i>subnet-address</i> is required, and for configuration simplicity, you configure the same <i>subnet-address</i> on both peer PE devices.</p> <p>The commit script derives different IRB interface addresses per peer PE device based on <i>subnet-address</i> and <i>peer-id</i>, as follows:</p> <ul style="list-style-type: none"> <i>peer-id</i> 1: IRB interface address is <i>subnet-address</i>. Otherwise, IRB interface address is <i>subnet-address</i> + 1 (add 1 to low-order address segment of the subnet address). <p>For example, if IPv4 <i>subnet-address</i> is 10.10.1.0/24:</p> <ul style="list-style-type: none"> On <i>peer-id</i> 1, address is 10.10.1.0/24 On <i>peer-id</i> 2, address is 10.10.1.1/24 <p>NOTE: The commit process doesn't do a commit check to enforce setting the same subnet address across both devices. By default, the commit script will derive the IRB interface subnet address based on the configured <i>subnet-address</i> at this hierarchy level.</p> <p>As a result, we strongly recommend you use the same subnet address for this parameter on both peer PE devices, and only set the <i>peer-id</i> to be different on each device. This way the default address derivation works as expected for the IRB interfaces on each peer PE device.</p>

Table 18: Derived and Default Configuration Values (Continued)

Configuration Element	Default or Derived Value
<p>IRB interface virtual gateway address</p> <p>From <code>set services evpn evpn-vxlan irb <i>irb-instance</i> subnet-address (inet inet6) <i>subnet-address</i></code></p> <p>Include the following options at <code>[edit services evpn evpn-vxlan irb <i>irb-instance</i>]</code> to specify the virtual gateway IPv4 or IPv6 addresses instead of using the default derived virtual gateway addresses:</p> <ul style="list-style-type: none"> <code>virtual-gateway-v4-address <i>ipv4-virtual-gateway-address</i></code> <code>virtual-gateway-v6-address <i>ipv6-virtual-gateway-address</i></code> 	<p>Derive the virtual gateway address for the IRB instance as the highest configurable address in the IPv4 or IPv6 <i>subnet-address</i> subnet range.</p> <p>For example, if the IPv4 (<code>subnet-address inet</code>) <i>subnet-address</i> is 10.1.1.1/24, the derived virtual gateway IPv4 address is 10.1.1.254.</p> <p>Similarly, for example, if the IPv6 (<code>subnet-address inet6</code>) <i>subnet-address</i> is 2001:db8::10:1:1:1/112, the derived virtual gateway IPv6 address is 2001:db8::10:1:1:fffe.</p>
<p>IRB interface virtual gateway MAC address</p> <p>Include the following options at <code>[edit services evpn global-parameters]</code> to override the default virtual gateway IPv4 or IPv6 addresses:</p> <ul style="list-style-type: none"> <code>virtual-gateway (v4-mac v6-mac) <i>virtual-gateway-mac-address</i></code> —To set different virtual gateway MAC addresses for IPv4 traffic and IPv6 traffic <code>virtual-gateway-mac <i>virtual-gateway-mac-address</i></code> — To set same virtual gateway MAC address for both IPv4 traffic and IPv6 traffic 	<p>IPv4 virtual gateway MAC address: 00:00:5e:00:01:01</p> <p>IPv6 virtual gateway MAC address: 00:00:5e:00:02:01</p>
<p>VXLAN network identifier (VNI) mapping</p> <p>From:</p> <ul style="list-style-type: none"> <code>set services evpn evpn-vxlan server <i>server-name</i> vlan-id-list [<i>vlan-id</i> ...]</code> <code>set services evpn evpn-vxlan irb <i>irb-instance</i> vlan-id <i>vlan-id</i></code> 	<p>Add <i>vlan-id</i> to a base value of 10000</p>

Table 18: Derived and Default Configuration Values *(Continued)*

Configuration Element	Default or Derived Value
Underlay Peering for PE Devices—Default Protocol: EBGp BGP group name: <code>__SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY</code>	
Autonomous system (AS) number	$65000 + \text{peer-id}$
Local AS number and peer AS number for BGP group configuration (local-as <i>local-as</i> , neighbor peer-as <i>peer-as</i>) From set services evpn device-attribute peer-id <i>peer-id</i> .	On peer-id 1, local AS is $65000 + 4$ and peer AS is $65000 + 3$. The AS numbers are reversed on the other peer PE device peer-id 2: local AS is $65000 + 3$ and peer AS is $65000 + 4$.
Local device address and peer neighbor address for BGP group configuration (local-address, neighbor neighbor <i>neighbor-address</i> in generated configuration) From set services evpn device-attribute peer-to-peer peer-subnet (inet inet6) <i>subnet-address</i> .	On peer PE device peer-id 1, local address is peer-subnet <i>subnet-address</i> and neighbor address is <i>subnet-address</i> + 1. The local address and neighbor address are reversed on the other peer PE device peer-id 2: local address is peer-subnet <i>subnet-address</i> + 1 and neighbor address is <i>subnet-address</i> .
MTU size for underlay or overlay peering interfaces	Use the overlay-mtu or underlay-mtu sizes you configure at the [edit services evpn global-parameters mtu] hierarchy level instead of using the default MTU size. The default MTU size varies based on platform and type of interface. See <i>Media MTU and Protocol MTU</i> for more on MTU settings.

Table 18: Derived and Default Configuration Values *(Continued)*

Configuration Element	Default or Derived Value
Routing policy to advertise loopback interface to EBGp peer PE device	<pre> set policy-options policy-statement EXPORT-LO0 term LOOPBACK from interface lo0.0 set policy-options policy-statement EXPORT-LO0 term LOOPBACK then accept set policy-options policy-statement EXPORT-LO0 term REJECT then reject set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGp_UNDERLAY export EXPORT-LO </pre>
Overlay Peering for PE Devices—Default Protocol: EBGp BGP group name: __SERVICES_EVPN_EVPN_VXLAN_EBGp_OVERLAY	
Autonomous system (AS) number (for routing-options autonomous-system <i>num</i>) From set services evpn device-attribute peer-id <i>peer-id</i> .	65000 + <i>peer-id</i>
Local device address and peer neighbor address for generated BGP group configuration (local-address <i>local-address</i> and neighbor <i>neighbor-address</i>) From set services evpn device-attribute : <ul style="list-style-type: none"> peer-id <i>peer-id</i> loopback peer1-subnet <i>peer1-subnet</i> loopback peer2-subnet <i>peer2-subnet</i> 	Each peer device uses the provided loopback peer <i>peer-id</i> -subnet addresses for the local address and its neighbor address.
Other Underlay Peering Options for PE Devices—OSPF If you configure set services evpn device-attribute peer-to-peer underlay-connectivity ospf:	
Aggregated Ethernet interface logical unit with OSPF underlay peering	ae0.0

Table 18: Derived and Default Configuration Values (*Continued*)

Configuration Element	Default or Derived Value
OSPF area	0.0.0.0
Other Overlay Peering Options for PE Devices—IBGP If you configure <code>set services evpn device-attribute peer-to-peer overlay-connectivity ibgp:</code>	
Autonomous system number for internal BGP peering (set <code>routing-options autonomous-system num</code> in generated configuration)	65000
Local device address and peer neighbor address for generated IBGP group configuration (local-address <i>local-address</i> and neighbor <i>neighbor-address</i>) From <code>set services evpn device-attribute :</code> <ul style="list-style-type: none"> • <code>peer-id peer-id</code> • <code>loopback peer1-subnet peer1-subnet</code> • <code>loopback peer2-subnet peer2-subnet</code> 	Same as with an EBGp overlay—each peer device uses the provided loopback <i>peer-peer-id</i> -subnet addresses for the local address and its neighbor address.
Option to Configure DHCP Relay for a VRF Routing Instance If you configure <code>set services evpn evpn-vxlan dhcp-relay name :</code>	

Table 18: Derived and Default Configuration Values *(Continued)*

Configuration Element	Default or Derived Value
DHCP Relay group configuration From set services evpn evpn-vxlan dhcp-relay <i>name</i> : <ul style="list-style-type: none"> • dhcp-server-address <i>dhcp-server-address</i> • relay-source <i>relay-source-interface</i> • vrf-instance <i>instance-id</i> 	Derived DHCP relay group name is SERVICES_EVPN_EVPN_VXLAN_VRF <i>name</i> and default configuration is: <pre> [edit routing-instances __SERVICES_EVPN_EVPN_VXLAN_VRF_<i>instance-id</i> forwarding-options] set dhcp-relay forward-only set dhcp-relay server-group SERVICES_EVPN_EVPN_VXLAN_VRF<i>named</i>dhcp-server-address [edit routing-instances __SERVICES_EVPN_EVPN_VXLAN_VRF_<i>instance-id</i> forwarding-options dhcp-relay group SERVICES_EVPN_EVPN_VXLAN_VRF<i>name</i>] set active-server-group SERVICES_EVPN_EVPN_VXLAN_VRF<i>name</i> set overrides relay-source <i>relay-source-interface</i> set relay-option-82 server-id-override </pre>

Platform-Specific Behavior for EZ-LAG Generated Configurations

See the [Simplified configuration for ESI LAGS with EVPN dual homing \(EZ-LAG\)](#) entry in Feature Explorer to review the platforms and releases in which we support the EZ-LAG feature.

Use the following table to review platform-specific differences in the generated configuration on different platforms that support the EZ-LAG feature:

Platform	Difference
EX9204, EX9208, and EX9214 routers	Storm control default generated configuration: <pre>set forwarding-options storm-control-profiles __SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL all bandwidth-percentage 1 set interfaces <i>interface-name</i> family ethernet- switching storm-control __SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL</pre>
Other supported platforms besides EX9204, EX9208, and EX9214 routers	Storm control default generated configuration: <pre>set forwarding-options storm-control-profiles __SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL all bandwidth-percentage 1 set interfaces <i>interface-name</i> ether-options ethernet- switch-profile storm-control __SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL</pre>

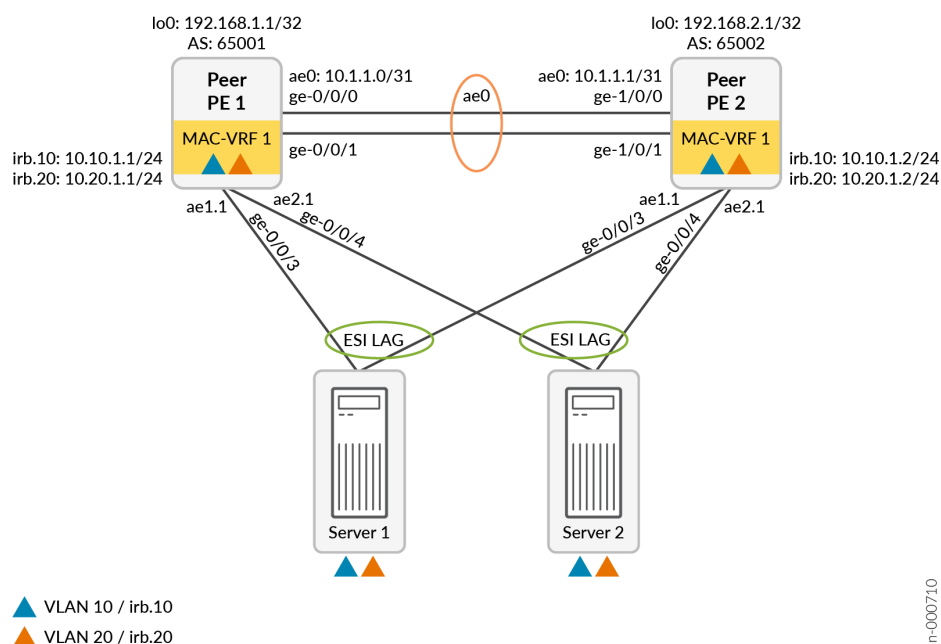
Easy EVPN LAG Configuration with Multihomed Servers

IN THIS SECTION

- Simplified Configuration with Two Multihomed Servers | 257
- Generated Configuration with Two Multihomed Servers | 261
- Derived Values in Generated Configuration | 265

Figure 21 on page 257 shows a topology with two multihomed servers connected to two peer PE devices. The peer PE devices are connected back-to-back in a small EVPN fabric that hosts two server VLANs (VLAN IDs 10 and 20).

Figure 21: Easy EVPN LAG Configuration with Two Multihomed Servers



Simplified Configuration with Two Multihomed Servers

This section shows an example of a minimal easy EVPN LAG configuration for the topology in [Figure 21 on page 257](#). You use statements in the `[edit services evpn]` configuration stanza. This configuration provides all of the required parameters for the commit script to generate a corresponding EVPN-VXLAN configuration. You configure only a few elements that are specific to each peer PE device. Otherwise, most of the configuration is the same on both devices.



NOTE: In this example, both peer PE devices host the same VLANs and use the same physical interface names to link to the multihomed servers, so those required configuration parameters can all be part of the common configuration across the peer PE devices. However, because physical interface allocations per device are usually different in actual customer deployments, we include the easy EVPN LAG configuration statements for those parameters separately below for each peer PE device.

The commit script uses default values for some elements. It also automatically derives other values for the generated configuration from the parameters in the simplified configuration, as mentioned earlier.

In this configuration, note that we configure the following common subnet address parameters across both peer PE devices. We do this because the commit script uses or automatically derives values on each peer PE device based on the *peer-id* on which the script is running:

- The device loopback subnet addresses for each device:

```
set services evpn device-attribute loopback peer1-subnet peer1-subnet peer2-subnet peer2-subnet
```

The commit script uses the provided loopback subnet address as the local address for the *peer-id* on which it runs. It uses the other *peer-id* loopback subnet address as its peer PE device neighbor address.

- The peer PE device peer-to-peer link subnet addresses:

```
set services evpn device-attribute peer-to-peer peer-subnet inet subnet-address
```

The commit script uses the aggregated Ethernet interface subnet address for *peer-id* 1 as the configured *subnet-address*, and derives the address for *peer-id* 2 as that *subnet-address* plus 1 in the low-order byte of the address subnet range.

- The IRB interface subnet address for each IRB interface that serves each VLAN:

```
set services evpn evpn-vxlan irb irb-instance subnet-address inet subnet-address
```

The commit script derives an IRB interface address in the same way as for the peer-to-peer aggregated Ethernet addresses above—it uses the configured *subnet-address* for *peer-id* 1, and adds 1 to the low-order byte of that *subnet-address* address range for *peer-id* 2.



NOTE: The commit process doesn't do a commit check to enforce setting the same subnet address across both devices. By default, the commit script will automatically derive these addresses based on the provided *subnet-address* *peer-id* 1 values. As a result, we strongly recommend that you make sure to set these subnet addresses to the same base value on both peer PE devices in an easy EVPN LAG configuration. That way the default address derivation works as expected for the loopback device, loopback peer link, and IRB interface subnet addresses on each peer PE device.

See "[Derived Values in Generated Configuration](#)" on page 265 for details on all of the values that the commit script derives for the generated configuration.

Easy EVPN LAG Configuration for Two Multihomed Servers

Peer PE 1:

```
set services evpn device-attribute peer-id 1
set services evpn device-attribute peer-to-peer peer-subnet interface-name ge-0/0/0
set services evpn device-attribute peer-to-peer peer-subnet interface-name ge-0/0/1
set services evpn evpn-vxlan server SERVER_1 interface ge-0/0/3
set services evpn evpn-vxlan server SERVER_2 interface ge-0/0/4
```

Peer PE 2:

```
set services evpn device-attribute peer-id 2
set services evpn device-attribute peer-to-peer peer-subnet interface-name ge-1/0/0
set services evpn device-attribute peer-to-peer peer-subnet interface-name ge-1/0/1
set services evpn evpn-vxlan server SERVER_1 interface ge-0/0/3
set services evpn evpn-vxlan server SERVER_2 interface ge-0/0/4
```

Common Configuration on Both Peer PE Devices:

```
set services evpn device-attribute loopback peer1-subnet 192.168.1.1/32 peer2-subnet
192.168.2.1/32
set services evpn device-attribute system-id 10:11:12:13:14:10
set services evpn device-attribute peer-to-peer peer-subnet inet 10.1.1.0/31
set services evpn evpn-vxlan irb irb_10 vlan-id 10
set services evpn evpn-vxlan irb irb_10 subnet-address inet 10.10.1.1/24
set services evpn evpn-vxlan irb irb_20 vlan-id 20
set services evpn evpn-vxlan irb irb_20 subnet-address inet 10.20.1.1/24
set services evpn evpn-vxlan server SERVER_1 esi-lag-id 1
set services evpn evpn-vxlan server SERVER_1 vlan-id-list [ 10 20 ]
set services evpn evpn-vxlan server SERVER_2 esi-lag-id 2
set services evpn evpn-vxlan server SERVER_2 vlan-id-list [ 10 20 ]
```

Default Parameters Not Specified in the Simplified Configuration

The commit script uses the following default elements that you don't specify in this simplified configuration:

Table 19: Default Elements Used in Generated Configuration

Configuration Element	Default Value
Aggregated Ethernet device count	255
Peer to peer PE device links	ae0
Overlay peering autonomous system number base value	65000
Underlay and overly peering protocol	EBGP
Underlay BGP group name	__SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY
Underlay export policy name and policy statement	EXPORT-LO0
Overlay BGP group name	__SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY
EVPN-VXLAN MAC-VRF instance	<i>instance-id</i> 1 Name: __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 service-type vlan-aware encapsulation vxlan vtep-source-interface lo0.0
Virtual gateway MAC address (for IPv4)	00:00:5e:00:01:01
VLAN name	SERVICES_EVPN_EVPN_VXLAN_VLAN_ <i>vlan-id</i>
Storm control profile name	__SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL

Generated Configuration with Two Multihomed Servers

With the easy EVPN LAG configuration commit script enabled, when you commit the configuration in ["Simplified Configuration with Two Multihomed Servers" on page 257](#), the commit script generates the following EVPN-VXLAN configuration. By default, the commit script configures storm control and lightweight loop detection on the server-facing interfaces.

See ["Derived Values in Generated Configuration" on page 265](#) for all the values the commit script derives in this generated configuration from the simplified configuration.

Peer PE 1:

```
set chassis aggregated-devices ethernet device-count 255
set chassis network-services enhanced-ip
set interfaces ge-0/0/0 ether-options 802.3ad ae0
set interfaces ge-0/0/1 ether-options 802.3ad ae0
set interfaces ge-0/0/3 ether-options ethernet-switch-profile storm-control
__SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL
set interfaces ge-0/0/3 ether-options 802.3ad ae1
set interfaces ge-0/0/4 ether-options ethernet-switch-profile storm-control
__SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL
set interfaces ge-0/0/4 ether-options 802.3ad ae2
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 unit 0 family inet address 10.1.1.0/31
set interfaces ae1 vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 esi auto-derive type-1-lacp
set interfaces ae1 esi all-active
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp system-id 10:11:12:13:14:11
set interfaces ae1 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 1 family ethernet-switching vlan members 10
set interfaces ae1 unit 1 family ethernet-switching vlan members 20
set interfaces ae2 vlan-tagging
set interfaces ae2 encapsulation flexible-ethernet-services
set interfaces ae2 esi auto-derive type-1-lacp
set interfaces ae2 esi all-active
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp system-id 10:11:12:13:14:12
set interfaces ae2 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae2 unit 1 family ethernet-switching vlan members 10
set interfaces ae2 unit 1 family ethernet-switching vlan members 20
set interfaces irb unit 10 virtual-gateway-accept-data
```

```

set interfaces irb unit 10 family inet address 10.10.1.1/24 virtual-gateway-address 10.10.1.254
set interfaces irb unit 10 virtual-gateway-v4-mac 00:00:5e:00:01:01
set interfaces irb unit 20 virtual-gateway-accept-data
set interfaces irb unit 20 family inet address 10.20.1.1/24 virtual-gateway-address 10.20.1.254
set interfaces irb unit 20 virtual-gateway-v4-mac 00:00:5e:00:01:01
set interfaces lo0 unit 0 family inet address 192.168.1.1/32 primary
set interfaces lo0 unit 0 family inet address 192.168.1.1/32 preferred
set forwarding-options storm-control-profiles __SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL all
bandwidth-percentage 1
set policy-options policy-statement EXPORT-LO0 term LOOPBACK from interface lo0.0
set policy-options policy-statement EXPORT-LO0 term LOOPBACK then accept
set policy-options policy-statement EXPORT-LO0 term REJECT then reject
set policy-options policy-statement pplb then load-balance per-packet
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 instance-type mac-vrf
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 protocols evpn encapsulation vxlan
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 protocols evpn default-gateway no-
gateway-community
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vtep-source-interface lo0.0
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 service-type vlan-aware
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 interface ae1.1
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 interface ae2.1
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 route-distinguisher 192.168.1.1:1
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vrf-target target:1:1
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_10 vlan-id 10
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_10 l3-interface irb.10
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_10 vxlan vni 10010
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_20 vlan-id 20
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_20 l3-interface irb.20
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_20 vxlan vni 10020
set routing-options router-id 192.168.1.1
set routing-options autonomous-system 65001
set routing-options forwarding-table export pplb
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY type external
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY local-address 10.1.1.0
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY export EXPORT-LO0
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY local-as 65004
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY multipath multiple-as

```

```

set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY neighbor 10.1.1.1 peer-as 65003
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY type external
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY multihop ttl 2
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY multihop no-nexthop-change
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY local-address 192.168.1.1
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY family evpn signaling
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY neighbor 192.168.2.1 peer-as
65002
set protocols evpn no-core-isolation
set protocols l2-learning platform-parameters no-mac-flush-on-aa-ae-down
set protocols loop-detect enhanced interface ae1.1 vlan-id 10
set protocols loop-detect enhanced interface ae1.1 loop-detect-action interface-down
set protocols loop-detect enhanced interface ae1.1 transmit-interval 1s
set protocols loop-detect enhanced interface ae1.1 revert-interval 60
set protocols loop-detect enhanced interface ae2.1 vlan-id 10
set protocols loop-detect enhanced interface ae2.1 loop-detect-action interface-down
set protocols loop-detect enhanced interface ae2.1 transmit-interval 1s
set protocols loop-detect enhanced interface ae2.1 revert-interval 60

```

Peer PE 2:

```

set chassis aggregated-devices ethernet device-count 255
set chassis network-services enhanced-ip
set interfaces ge-0/0/0 ether-options 802.3ad ae0
set interfaces ge-0/0/1 ether-options 802.3ad ae0
set interfaces ge-0/0/3 ether-options ethernet-switch-profile storm-control
__SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL
set interfaces ge-0/0/3 ether-options 802.3ad ae1
set interfaces ge-0/0/4 ether-options ethernet-switch-profile storm-control
__SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL
set interfaces ge-0/0/4 ether-options 802.3ad ae2
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 unit 0 family inet address 10.1.1.0/31
set interfaces ae1 vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 esi auto-derive type-1-lacp
set interfaces ae1 esi all-active
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp system-id 10:11:12:13:14:11
set interfaces ae1 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 1 family ethernet-switching vlan members 10
set interfaces ae1 unit 1 family ethernet-switching vlan members 20

```



```

set interfaces ae2 vlan-tagging
set interfaces ae2 encapsulation flexible-ethernet-services
set interfaces ae2 esi auto-derive type-1-lacp
set interfaces ae2 esi all-active
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp system-id 10:11:12:13:14:12
set interfaces ae2 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae2 unit 1 family ethernet-switching vlan members 10
set interfaces ae2 unit 1 family ethernet-switching vlan members 20
set interfaces irb unit 10 virtual-gateway-accept-data
set interfaces irb unit 10 family inet address 10.10.1.2/24 virtual-gateway-address 10.10.1.254
set interfaces irb unit 10 virtual-gateway-v4-mac 00:00:5e:00:01:01
set interfaces irb unit 20 virtual-gateway-accept-data
set interfaces irb unit 20 family inet address 10.20.1.2/24 virtual-gateway-address 10.20.1.254
set interfaces irb unit 20 virtual-gateway-v4-mac 00:00:5e:00:01:01
set interfaces lo0 unit 0 family inet address 192.168.2.1/32 primary
set interfaces lo0 unit 0 family inet address 192.168.2.1/32 preferred
set forwarding-options storm-control-profiles __SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL all
bandwidth-percentage 1
set policy-options policy-statement EXPORT-LO0 term LOOPBACK from interface lo0.0
set policy-options policy-statement EXPORT-LO0 term LOOPBACK then accept
set policy-options policy-statement EXPORT-LO0 term REJECT then reject
set policy-options policy-statement pplb then load-balance per-packet
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 instance-type mac-vrf
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 protocols evpn encapsulation vxlan
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 protocols evpn default-gateway no-
gateway-community
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vtep-source-interface lo0.0
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 service-type vlan-aware
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 interface ae1.1
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 interface ae2.1
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 route-distinguisher 192.168.2.1:1
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vrf-target target:1:1
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_10 vlan-id 10
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_10 l3-interface irb.10
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_10 vxlan vni 10010
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_20 vlan-id 20
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_20 l3-interface irb.20

```

```

set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_20 vxlan vni 10020
set routing-options router-id 192.168.2.1
set routing-options autonomous-system 65002
set routing-options forwarding-table export pplb
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY type external
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY local-address 10.1.1.1
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY export EXPORT-L00
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY local-as 65003
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY multipath multiple-as
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY neighbor 10.1.1.0 peer-as 65004
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY type external
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY multihop ttl 2
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY multihop no-nexthop-change
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY local-address 192.168.2.1
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY family evpn signaling
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_EBGP_OVERLAY neighbor 192.168.1.1 peer-as
65001
set protocols evpn no-core-isolation
set protocols l2-learning platform-parameters no-mac-flush-on-aa-ae-down
set protocols loop-detect enhanced interface ae1.1 vlan-id 10
set protocols loop-detect enhanced interface ae1.1 loop-detect-action interface-down
set protocols loop-detect enhanced interface ae1.1 transmit-interval 1s
set protocols loop-detect enhanced interface ae1.1 revert-interval 60
set protocols loop-detect enhanced interface ae2.1 vlan-id 10
set protocols loop-detect enhanced interface ae2.1 loop-detect-action interface-down
set protocols loop-detect enhanced interface ae2.1 transmit-interval 1s
set protocols loop-detect enhanced interface ae2.1 revert-interval 60

```

Derived Values in Generated Configuration

In the generated configuration in ["Generated Configuration with Two Multihomed Servers" on page 261](#), the commit script derives the following values based on corresponding easy EVPN LAG configuration elements:

Table 20: Derived Values in Generated Configuration Example with Two Multihomed Servers

Configuration Element	Derived From [edit services evpn]	Derived Value on Peer <i>peer-id</i>	
		Peer 1	Peer 2
AS number	Default AS number base: 65000	Underlay AS number: 65004 Overlay AS number: 65001	Underlay AS number: 65003 Overlay AS number: 65002
Peer to peer ae0 address	device-attribute peer-to-peer peer-subnet inet 10.1.1.0/31	10.1.1.0/31	10.1.1.1/31
Underlay EBGp local address and neighbor address	device-attribute peer-to-peer peer-subnet inet 10.1.1.0/31	local-address: 10.1.1.0/31 neighbor: 10.1.1.1/31	local-address: 10.1.1.1/31 neighbor: 10.1.1.0/31
Overlay EBGp local address and neighbor address	device-attribute loopback peer1-subnet 192.168.1.1/32 peer2-subnet 192.168.2.1/32	local-address: 192.168.1.1/32 neighbor: 192.168.2.1/32	local-address: 192.168.2.1/32 neighbor: 192.168.1.1/32
<i>system-id</i> for LACP configuration per server ESI LAG link	device-attribute system-id 10:11:12:13:14:10	ae1: 10:11:12:13:14:11 ae2: 10:11:12:13:14:12	ae1: 10:11:12:13:14:11 ae2: 10:11:12:13:14:12
Server VLANs for: <ul style="list-style-type: none"> • aexVLAN members • MAC-VRF instance VLAN members • IRB interface logical units 	evpn-vxlan server SERVER_1 vlan-id-list [10 20] evpn-vxlan server SERVER_2 vlan-id-list [10 20]	vlan-id 10 vlan-id 20	vlan-id 10 vlan-id 20

Table 20: Derived Values in Generated Configuration Example with Two Multihomed Servers
(Continued)

Configuration Element	Derived From [edit services evpn]	Derived Value on Peer <i>peer-id</i>	
		Peer 1	Peer 2
IRB interface subnet address and virtual-gateway-address (VLAN 10)	evpn-vxlan irb irb_10 subnet-address inet 10.10.1.1/24	10.10.1.1/24 10.10.1.254/24	10.10.1.2/24 10.10.1.254/24
IRB interface subnet address and virtual-gateway-address (VLAN 20)	evpn-vxlan irb irb_20 subnet-address inet 10.20.1.1/24	10.20.1.1/24 10.20.1.254/24	10.20.1.2/24 10.20.1.254/24
MAC-VRF instance route distinguisher	Default MAC-VRF <i>instance-id</i> 1 device-attribute loopback peer1-subnet 192.168.1.1/32 peer2-subnet 192.168.2.1/32	192.168.1.1:1	192.168.2.1:1
MAC-VRF instance route target	Default MAC-VRF <i>instance-id</i> 1 vrf-target:1: <i>instance-id</i>	vrf-target:1:1	vrf-target:1:1
MAC-VRF instance aggregated Ethernet interface logical units per server ESI LAG link: ae[<i>esi-lag-id</i>].[<i>instance-id</i>]	Default MAC-VRF <i>instance-id</i> 1 evpn-vxlan server SERVER_1 esi-lag-id esi-lag-id 1 evpn-vxlan server SERVER_2 esi-lag-id esi-lag-id 2	Server 1: ae1.1 Server 2: ae2.1	Server 1: ae1.1 Server 2: ae2.1

Table 20: Derived Values in Generated Configuration Example with Two Multihomed Servers
(Continued)

Configuration Element	Derived From [edit services evpn]	Derived Value on Peer <i>peer-id</i>	
		Peer 1	Peer 2
IRB interface names (VLANs 10 and 20)	evpn-vxlan irb irb_10 vlan-id 10 evpn-vxlan irb irb_20 vlan-id 20 evpn-vxlan server SERVER_1 vlan-id-list [10 20]	irb.10 irb.20	irb.10 irb.20
VLAN to VNI mappings (VLANs 10 and 20)	Default VNI base value: 10000	For VLAN 10: 10010 For VLAN 20: 10020	for VLAN 10: 10010 for VLAN 20: 10020

Add Configuration for a New Multihomed Server

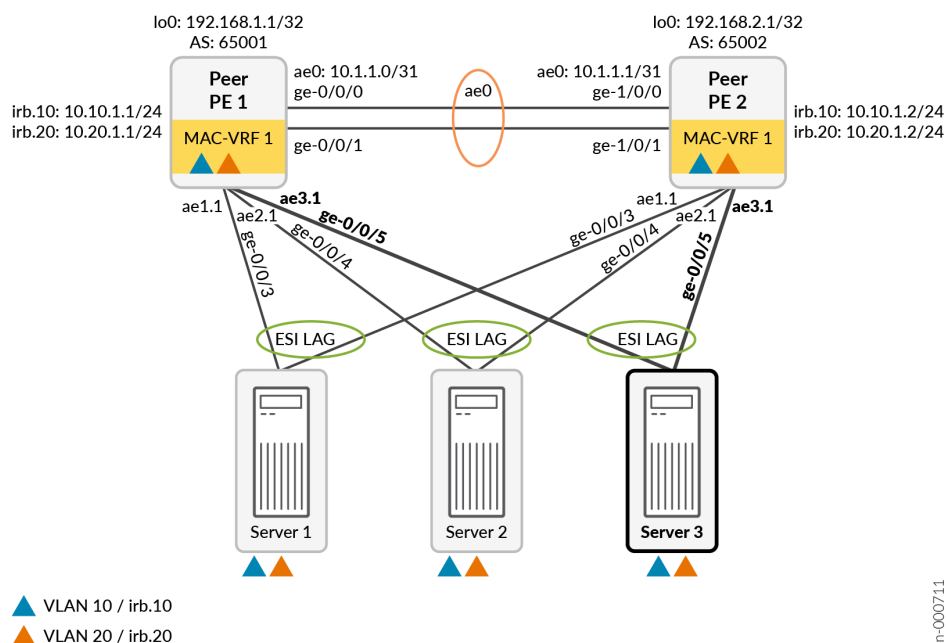
IN THIS SECTION

- [Simplified Configuration to Add a New Multihomed Server and ESI LAG | 269](#)
- [Additional Generated Configuration for a New Multihomed Server and ESI LAG | 270](#)

[Figure 22 on page 269](#) shows the same topology as [Figure 21 on page 257](#) with an additional multihomed server.

The example configuration here shows how to add the new multihomed server, Server 3, that hosts the two VLANs VLAN 10 and VLAN 20.

Figure 22: Add a New Multihomed Server to an Existing Easy EVPN LAG Configuration



Simplified Configuration to Add a New Multihomed Server and ESI LAG

In this example, both peer PE devices:

- Link to the new server using interface ge-0/0/5.
- Host the same VLANs, VLAN 10 and VLAN 20.
- Automatically derive the ES identifier by default (using the `set interfaces aex esi auto-derive type-1-lacp` command)

As a result, you can add the same additional easy EVPN LAG configuration statements on both devices, and the commit script generates the same additional configuration statements on both devices. You can also combine `server server-name` options at the `[edit services evpn evpn-vxlan]` hierarchy level into the same command, as this configuration example shows. You only need to add one configuration line item for this use case.

To add Server 3 with corresponding ESI LAG links, add the following single easy EVPN LAG configuration command to the existing simplified configuration on both Peer PE 1 and Peer PE 2:

```
set services evpn evpn-vxlan server SERVER_3 esi-lag-id 3 vlan-id-list [ 10 20 ] interface ge-0/0/5
```

Additional Generated Configuration for a New Multihomed Server and ESI LAG

The commit script generates the following additional configuration on both peer PE devices from the simplified configuration statements in ["Simplified Configuration to Add a New Multihomed Server and ESI LAG" on page 269](#):

```
set interfaces ge-0/0/5 ether-options ethernet-switch-profile storm-control
__SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL
set interfaces ge-0/0/5 ether-options 802.3ad ae3
set interfaces ae3 vlan-tagging
set interfaces ae3 encapsulation flexible-ethernet-services
set interfaces ae3 esi auto-derive type-1-lacp
set interfaces ae3 esi all-active
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp system-id 10:11:12:13:14:13
set interfaces ae3 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae3 unit 1 family ethernet-switching vlan members 10
set interfaces ae3 unit 1 family ethernet-switching vlan members 20
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 interface ae3.1
set protocols loop-detect enhanced interface ae3.1 vlan-id 10
set protocols loop-detect enhanced interface ae3.1 loop-detect-action interface-down
set protocols loop-detect enhanced interface ae3.1 transmit-interval 1s
set protocols loop-detect enhanced interface ae3.1 revert-interval 60
```

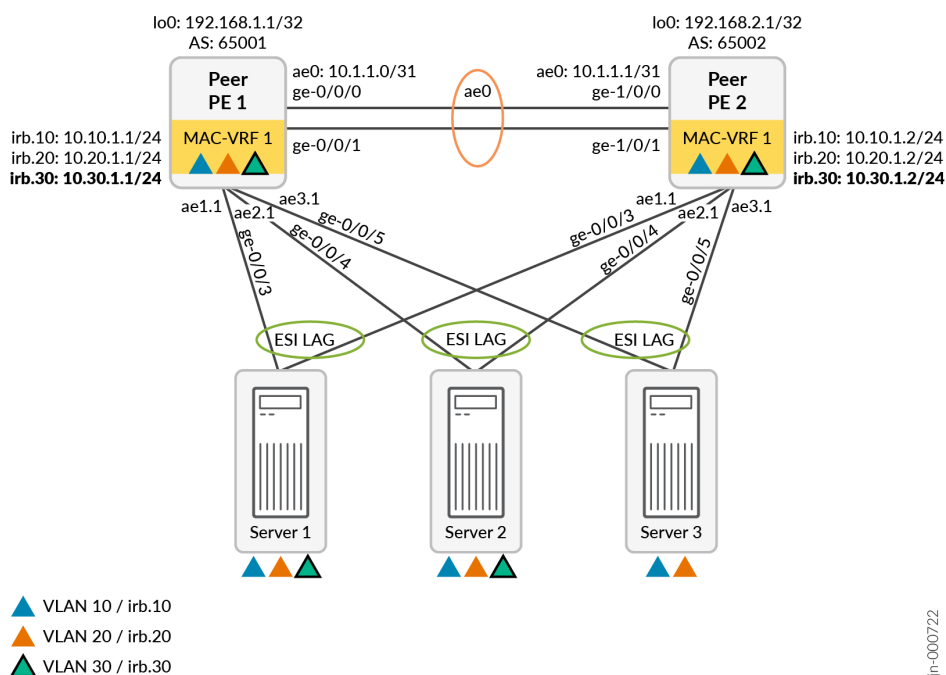
Add a New VLAN and IRB Interfaces

IN THIS SECTION

- [Simplified Configuration to Add a New VLAN | 271](#)
- [Additional Generated Configuration for a New VLAN | 272](#)

[Figure 23 on page 271](#) shows the same topology as [Figure 23 on page 271](#) with a new VLAN, VLAN 30, hosted only by Server 1 and Server 2.

Figure 23: Add a New VLAN and IRB Interfaces to an Existing Server Configuration



Simplified Configuration to Add a New VLAN

To add VLAN 30 for Server 1 and Server 2 to the original configuration in "[Easy EVPN LAG Configuration with Multihomed Servers](#)" on page 256), add the following easy EVPN LAG configuration commands:

On both Peer PE devices:

```
set services evpn evpn-vxlan irb IRB_30 vlan-id 30 subnet-address inet 10.30.1.1/24
set services evpn evpn-vxlan server SERVER_1 vlan-id-list [ 30 ]
set services evpn evpn-vxlan server SERVER_2 vlan-id-list [ 30 ]
```



NOTE: The simplified EVPN LAG configuration here is the same on both peer PE devices in this case because:

- The commit script derives unique IRB interface subnet addresses for you using the provided *subnet-address* parameter.
- We are adding the same VLAN on both devices.

Additional Generated Configuration for a New VLAN

The commit script generates the following additional configuration for VLAN 30:

Peer PE 1:

```
set interfaces ae1 unit 1 family ethernet-switching vlan members 30
set interfaces ae2 unit 1 family ethernet-switching vlan members 30
set interfaces irb unit 30 virtual-gateway-accept-data
set interfaces irb unit 30 family inet address 10.30.1.1/24 virtual-gateway-address 10.30.1.254
set interfaces irb unit 30 virtual-gateway-v4-mac 00:00:5e:00:01:01
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_30 vlan-id 30
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_30 l3-interface irb.30
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_30 vxlan vni 10030
```

Peer PE 2:

```
set interfaces ae1 unit 1 family ethernet-switching vlan members 30
set interfaces ae2 unit 1 family ethernet-switching vlan members 30
set interfaces irb unit 30 virtual-gateway-accept-data
set interfaces irb unit 30 family inet address 10.30.1.2/24 virtual-gateway-address 10.30.1.254
set interfaces irb unit 30 virtual-gateway-v4-mac 00:00:5e:00:01:01
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_30 vlan-id 30
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_30 l3-interface irb.30
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 vlans
SERVICES_EVPN_EVPN_VXLAN_VLAN_30 vxlan vni 10030
```

Add a New Single-homed Server

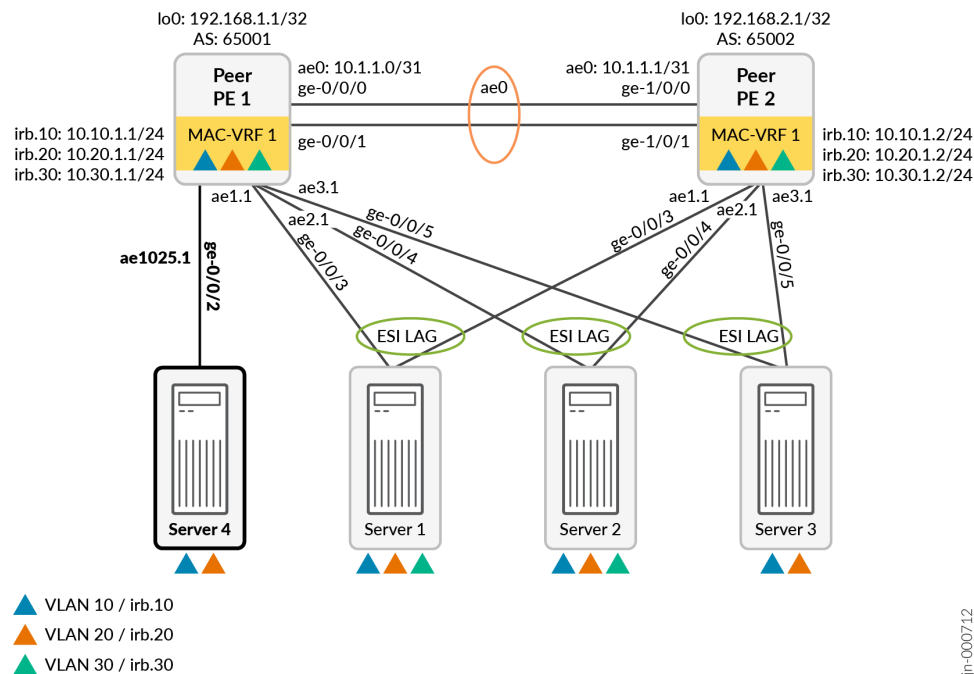
IN THIS SECTION

- Simplified Configuration to Add a Single-homed Server | 274
- Additional Generated Configuration for a Single-homed Server | 274

Figure 24 on page 273 adds a single-homed server, Server 4, that connects only to Peer PE 1. Server 4 hosts VLANs VLAN 10 and VLAN 20.

In the simplified configuration, you provide a *single-home-id* to identify single-homed servers links and the parameters related to that server. By default, the commit script uses aggregated Ethernet interface names starting with a base index of 1024 for links to single-homed servers. The commit script adds the *single-home-id* to that index, producing interface names that start with ae1025 (for *single-home-id* = 1).

Figure 24: Add Configuration for a Single-homed Server to an Existing Easy EVPN LAG Configuration



Simplified Configuration to Add a Single-homed Server

To add single-homed Server 4 to the original configuration in ["Easy EVPN LAG Configuration with Multihomed Servers" on page 256](#)), add the following easy EVPN LAG configuration commands on Peer PE 1:

```
set services evpn evpn-vxlan server SERVER_4 single-home-id 1 vlan-id-list [ 10 20 ] interface
ge-0/0/2
```



NOTE: You can combine server *server-name* options at the [edit services evpn evpn-vxlan] hierarchy level, as this configuration example shows. As a result, you only need to add one configuration line item for this use case.

Additional Generated Configuration for a Single-homed Server

The commit script generates the following additional configuration for Server 4 on Peer PE 1:

```
set interfaces ge-0/0/2 ether-options ethernet-switch-profile storm-control
__SERVICES_EVPN_EVPN_VXLAN_STORM_CONTROL
set interfaces ge-0/0/2 ether-options 802.3ad ae1025
set interfaces ae1025 vlan-tagging
set interfaces ae1025 encapsulation flexible-ethernet-services
set interfaces ae1025 aggregated-ether-options lacp active
set interfaces ae1025 aggregated-ether-options lacp system-id 10:11:12:13:18:19
set interfaces ae1025 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae1025 unit 1 family ethernet-switching vlan members 10
set interfaces ae1025 unit 1 family ethernet-switching vlan members 20
set routing-instances __SERVICES_EVPN_EVPN_VXLAN_MAC_VRF_1 interface ae1025.1
set protocols loop-detect enhanced interface ae1025.1 vlan-id 10
set protocols loop-detect enhanced interface ae1025.1 loop-detect-action interface-down
set protocols loop-detect enhanced interface ae1025.1 transmit-interval 1s
set protocols loop-detect enhanced interface ae1025.1 revert-interval 60
```

Use OSPF for the Underlay Configuration

By default, the easy EVPN LAG configuration commit script generates a configuration that uses EBGP for the underlay peering between the peer PE devices. See the `__SERVICES_EVPN_EVPN_VXLAN_EBGP_UNDERLAY` EBGP group configuration statements in the example in ["Generated Configuration with Two Multihomed Servers" on page 261](#).

If you want to use OSPF for the underlay peering instead, include the following option in your easy EVPN LAG configuration on both peer PE devices:

```
set services evpn device-attribute peer-to-peer underlay-connectivity ospf
```

With this option, in place of the default EBGp underlay configuration statements, the commit script generates an OSPF underlay peering configuration using the following default parameters (also refer to [Table 18 on page 241](#)):

- Aggregated Ethernet interface logical unit ae0.0
- OSPF area 0.0.0.0

The commit script generates the following default OSPF underlay peering configuration on both peer PE devices:

```
set protocols ospf area 0.0.0.0 interface ae0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
```



NOTE: If you don't want to use the default EBGp or OSPF underlay peering configurations, you can set the following easy EVPN LAG configuration option:

```
set services evpn defaults-override no-underlay-config
```

With this option set, the commit script will not generate any underlay peering configuration. In that case, you must manually configure the desired underlay peering.

Use IBGP for the Overlay Configuration

By default, the easy EVPN LAG configuration commit script generates a configuration that uses EBGp for the overlay peering between the peer PE devices. See the `__SERVICES_EVPN_EVPN_VXLAN_EBGp_OVERLAY` EBGp group configuration statements in the example in ["Generated Configuration with Two Multihomed Servers" on page 261](#).

If you want to use IBGP for the overlay peering instead, include the following option in your easy EVPN LAG configuration on both peer PE devices:

```
set services evpn device-attribute peer-to-peer overlay-connectivity ibgp
```

With this option, instead of the default EBGp overlay configuration, the commit script generates an IBGP overlay peering configuration using the following default or derived parameters (also refer to [Table 18 on page 241](#)):

- IBGP group name `__SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY`
- AS number 65000
- IBGP local address and peer neighbor address derived from `set services evpn device-attribute`:
 - peer-id *peer-id*
 - loopback peer1-subnet *peer1-subnet*
 - loopback peer2-subnet *peer2-subnet*

For the example topology in [Figure 21 on page 257](#), the generated configuration is:

Peer PE 1:

```
set routing-options autonomous-system 65000
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY type internal
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY local-address 192.168.1.1
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY family inet-vpn unicast
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY family evpn signaling
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY neighbor 192.168.2.1
```

Peer PE 2:

```
set routing-options autonomous-system 65000
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY type internal
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY local-address 192.168.2.1
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY family inet-vpn unicast
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY family evpn signaling
set protocols bgp group __SERVICES_EVPN_EVPN_VXLAN_IBGP_OVERLAY neighbor 192.168.1.1
```

RELATED DOCUMENTATION

[EVPN Multihoming Overview](#) | 162

[Understanding Automatically Generated ESIs in EVPN Networks](#) | 200

evpn (Easy EVPN LAG Configuration)

Configuring EVPN Active-Standby Multihoming to a Single PE Device

You can configure EVPN active-standby multihoming on a single PE device by configuring a standby (protect) interface to provide redundancy to an active (protected) interface. [Figure 25 on page 277](#) illustrates an EVPN topology with active-standby multihoming to a single PE device. A protect interface provides the benefit of a backup for the protected primary interface in case of failure. The PE device uses the primary interface for network traffic while the primary interface is functioning. If the primary interface fails, the protect interface becomes active and traffic switches to the protect interface. When the primary interface returns, the primary interface becomes the active interface once again.

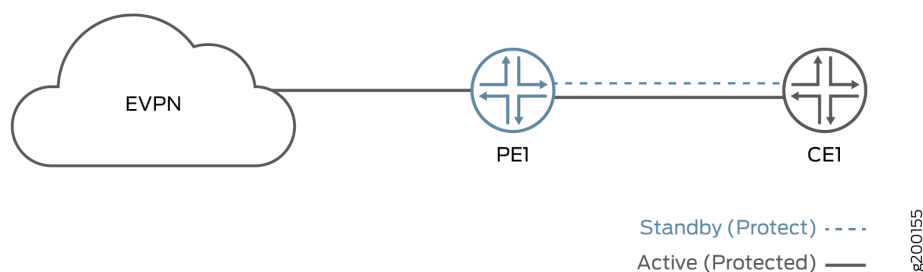


NOTE: If connectivity fault management (CFM) is enabled, the protect interface will trigger CFM to send Type, Length, and Value (TLV) or Remote Defect Indication (RDI) interface status to the customer edge device

Junos OS does not support the protect interface in the following cases:

- EVPN-VXLAN
- PBB-EVPN
- On an interface that has been configured as an ESI in EVPN multihoming

Figure 25: EVPN Active-Standby Multihoming on a Single PE Device



To configure the protect interface in a routing instance, configure both the protect interface and the primary interface in the routing instance for EVPN and include the protect-interface statement in the primary interface hierarchy.

```
routing-instances {
  routing-instance-name {
    instance-type type;
    interface primary-interface-name {
```

```

        protect-interface protect-interface-name
    }
    interface protect-interface-name
}
route-distinguisher (as-number:number / ip-address:number);
vrf-target community;
}

```

To configure the protect interface in a routing instance, configure both the protect interface and the primary interface in the routing instance for EVPN-VPWS and include the protect-interface statement in the evpn protocol hierarchy.

```

routing-instances {
    routing-instance-name {
        instance-type evpn-vpws;
        interface primary-interface-name ;
        interface protect-interface-name;
        route-distinguisher (as-number:number / ip-address:number);
        vrf-target community;
        protocols {
            evpn {
                interface primary-interface-name {
                    vpws-service-id {
                        local service-id;
                        remote service-id;
                    }
                    protect-interface protect-interface-name
                }
            }
        }
    }
}

```

To configure the protect interface in a bridge domain, configure both the protect interface and the primary interface in the bridge domain and include the protect-interface statement in the primary interface hierarchy.

```

bridge-domains {
    bridge-domain-name{
        domain-type bridge;
        vlan-id number;
    }
}

```

```

    interface primary-interface-name {
        protect-interface protect-interface-name
    }
    interface protect-interface-name
}
}

```

To display the protect interface, use the `show evpn instance extensive operational` command.

```

user@PE1> show evpn instance extensive
Instance: blue
Route Distinguisher: 10.255.255.1:100
Per-instance MAC route label: 299776
MAC database status Local Remote
MAC advertisements: 0 0
MAC+IP advertisements: 0 0
Default gateway MAC advertisements: 0 0
Number of local interfaces: 5 (5 up)
Interface name ESI Mode Status AC-Role
ae0.0 00:11:22:33:44:55:66:77:88:99 all-active Up Root
ge-0/0/3.0 00:00:00:00:00:00:00:00:00 single-homed Up Root
ge-0/0/4.0 00:11:11:11:44:55:66:77:88:99 all-active Up Root
ge-0/0/4.1 00:22:22:22:44:55:66:77:88:99 all-active Up Root
ge-0/0/4.50 00:00:00:00:00:00:00:00:00 single-homed Up Root
Number of IRB interfaces: 1 (0 up)
Interface name VLAN VNI Status L3 context
irb.1 25 Down vrf
Number of protect interfaces: 1
Interface name Protect Interface name Status
ge-0/0/3.1 ge-0/0/4.50 Protect-inactive

```

The `show interfaces detail` command shows that the protect interface has a CCC-DOWN status.

```

user@PE1> show interfaces ae0.0 detail
Logical interface ae80.0 (Index 399) (SNMP ifIndex 612) (Generation 223)
Flags: Up SNMP-Traps CCC-Down 0x20004000 VLAN-Tag [ 0x8100.10 ] Encapsulation: VLAN-Bridge
Statistics          Packets          pps          Bytes          bps
Bundle:
  Input :           0           0           0           0
  Output:           0           0           0           0
Adaptive Statistics:

```



```

Adaptive Adjusts:      0
Adaptive Scans   :      0
Adaptive Updates:      0
Link:
ge-0/1/8.0
Input :      0      0      0      0
Output:      0      0      0      0

Aggregate member links: 1

Marker Statistics:  Marker Rx      Resp Tx      Unknown Rx      Illegal Rx
ge-0/1/8.0          0          0          0          0
Protocol bridge, MTU: 1522, Generation: 263, Route table: 11, Mesh Group: __all_ces__

```

RELATED DOCUMENTATION

[EVPN Multihoming Overview | 162](#)

[Configuring EVPN-MPLS Active-Standby Multihoming | 280](#)

protect-interface

Configuring EVPN-MPLS Active-Standby Multihoming

You can configure multihoming in active-standby redundancy mode in an Ethernet VPN (EVPN) fabric with MPLS. This mode enables the device to autodiscover Ethernet segments, construct Ethernet segment routes, and assign Ethernet segment identifier (ESI) labels.



NOTE: We support active-standby multihoming in EVPN fabrics only with MPLS. Please refer to [EVPN-MPLS: Single-active multihoming support](#) for a complete list of the products that support this feature.

When configuring active-standby EVPN multihoming on supported devices, be aware of the following limitations:

- An interface or ESI can be attached to more than one EVPN instance (EVI), with a maximum limit of 200 EVIs per ESI.

- For an EVPN routing instance, only one logical interface per physical interface or ESI can be attached to an EVI.
- For a virtual switch routing instance, only one logical interface per physical interface or ESI can be configured under a bridge domain.

All the PE routers in the network topology that support this feature should be running Junos OS Release 14.1 or later releases, which are based on EVPN draft-ietf-l2vpn-evpn-03. Junos OS releases prior to 14.1 support the older version of the EVPN draft, causing interoperability issues when Junos OS Release 14.1 and a previous release are running.

Before you begin:

1. Configure the router interfaces.
2. Configure the router ID and autonomous system number for the device.
3. Configure OSPF or any other IGP protocol.
4. Configure a BGP internal group.
5. Include the EVPN signaling network layer reachability information (NLRI) to the internal BGP group.
6. Configure LDP.
7. Configure MPLS.
8. Configure RSVP MPLS LSP or GRE tunnels.

To configure the PE device:

1. Enable EVPN active-standby multihoming on the multihomed interfaces.

```
[edit interfaces]
user@PE1# set interface-name vlan-tagging
user@PE1# set interface-name encapsulation flexible-ethernet-services
user@PE1# set interface-name esi esi-value
user@PE1# set interface-name esi single-active
user@PE1# set interface-name unit 0 encapsulation vlan-bridge
user@PE1# set interface-name unit 0 vlan-id VLAN-ID
```

For example:

```
[edit interfaces]
user@PE1# set ge-0/0/4 vlan-tagging
user@PE1# set ge-0/0/4 encapsulation flexible-ethernet-services
```

```

user@PE1# set ge-0/0/4 esi 00:22:44:66:88:00:22:44:66:88
user@PE1# set ge-0/0/4 esi single-active
user@PE1# set ge-0/0/4 unit 0 encapsulation vlan-bridge
user@PE1# set ge-0/0/4 unit 0 vlan-id 300

```

2. Configure the routing instance for the active-standby mode of redundancy.

The active-standby multihoming can be configured under any EVPN routing-instance. We support evpn, virtual-switch, and mac-vrf instance types in active-standby EVPN multihoming. The vrf routing-instance is configured to illustrate the EVPN IRB functionality, in addition to multihoming, and is not mandatory for the active-standby EVPN multihoming feature to work. For example:

Virtual-switch Routing Instance

```

[edit routing-instances]
user@PE1# set virtual-switch-instance instance-type virtual-switch
user@PE1# set virtual-switch-instance protocols evpn extended-vlan-list VLAN-ID
user@PE1# set virtual-switch-instance bridge-domains bridge-domain-name domain-type bridge
user@PE1# set virtual-switch-instance bridge-domains bridge-domain-name vlan-id VLAN-ID
user@PE1# set virtual-switch-instance bridge-domains bridge-domain-name interface interface-name
user@PE1# set virtual-switch-instance bridge-domains bridge-domain-name routing-interface interface-name
user@PE1# set virtual-switch-instance route-distinguisher route-distinguisher-value
user@PE1# set virtual-switch-instance vrf-target vrf-target

```

OR

EVPN Routing Instance

```

[edit routing-instances]
user@PE1# set evpn-instance instance-type evpn
user@PE1# set evpn-instance vlan-id VLAN-ID
user@PE1# set evpn-instance interface interface-name
user@PE1# set evpn-instance routing-interface interface-name
user@PE1# set evpn-instance route-distinguisher route-distinguisher-value
user@PE1# set evpn-instance vrf-target vrf-target

```

OR

VRF Routing Instance

```

[edit routing-instances]
user@PE1# set vrf-instance instance-type vrf

```

```

user@PE1# set vrf-instance interface interface-name
user@PE1# set vrf-instance route-distinguisher route-distinguisher-value
user@PE1# set vrf-instance vrf-target vrf-target

```

3. Verify and commit the configuration.

For example:

```

[edit routing-instances]
user@PE1# set ALPHA instance-type virtual-switch
user@PE1# set ALPHA route-distinguisher 10.255.0.1:100
user@PE1# set ALPHA vrf-target target:100:100
user@PE1# set ALPHA protocols evpn extended-vlan-list 100
user@PE1# set ALPHA bridge-domains ONE domain-type bridge
user@PE1# set ALPHA bridge-domains ONE vlan-id 100
user@PE1# set ALPHA bridge-domains ONE interface ae0.0
user@PE1# set ALPHA bridge-domains ONE interface ge-0/0/2.0
user@PE1# set ALPHA bridge-domains ONE routing-interface irb.0
user@PE1# set BETA instance-type evpn
user@PE1# set BETA vlan-id 300
user@PE1# set BETA interface ge-0/0/4.0
user@PE1# set BETA interface ae1.0
user@PE1# set BETA routing-interface irb.1
user@PE1# set BETA route-distinguisher 10.255.0.1:300
user@PE1# set BETA vrf-target target:300:300
user@PE1# set DELTA instance-type vrf
user@PE1# set DELTA interface irb.0
user@PE1# set DELTA interface irb.1
user@PE1# set DELTA route-distinguisher 10.255.0.1:200
user@PE1# set DELTA vrf-target target:200:200
user@PE1# set DELTA vrf-table-label

```

```

[edit]
user@PE1# commit
commit complete

```

RELATED DOCUMENTATION

[EVPN Multihoming Overview](#) | 162

[Example: Configuring EVPN-MPLS Active-Standby Multihoming](#) | 308

Example: Configuring Basic EVPN-MPLS Active-Standby Multihoming

IN THIS SECTION

- Requirements | 284
- Overview and Topology | 284
- Configuration | 286
- Verification | 297

This example shows how to configure active-standby multihoming in an Ethernet VPN (EVPN) fabric with MPLS.

Requirements

This example uses the following hardware and software components:

- Four MX Series 5G Universal Routing Platforms running Junos OS Release 14.1 (or later), with MPC interfaces, acting as provider edge (PE) and provider (P) routers.
- Two customer edge (CE) devices.

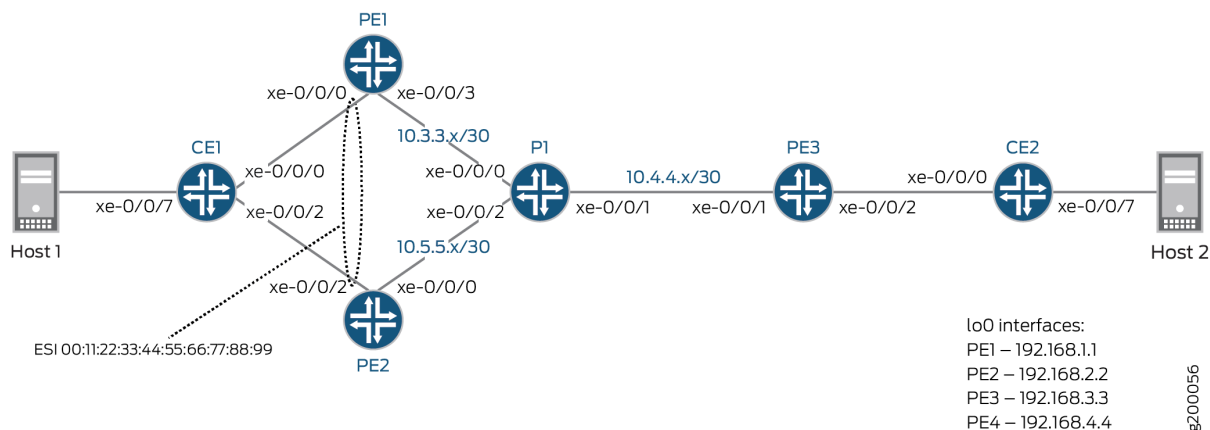


NOTE: We support active-standby multihoming in EVPN fabrics only with MPLS. Please refer to [EVPN-MPLS: Single-active multihoming support](#) for a complete list of the products that support this feature.

Overview and Topology

[Figure 26 on page 285](#) illustrates a simple EVPN topology. Routers PE1 and PE2 are provider edge (PE) routers connected to multihomed customer edge (CE) router CE1. An additional PE, router PE3, is a remote PE in the EVPN fabric connected to CE2, a single-homed CE router.

Figure 26: Simple EVPN Multihomed Topology



The network has the following characteristics:

- All PE and P routers are running OSPF.
- There is an IBGP mesh between all PE routers.
- MPLS (RSVP) LSPs are configured between all PE routers.
- On routers PE1 and PE2, each device's CE-facing interface uses the same Ethernet Segment Identifier (ESI).

For simplicity and consistency of configuration, this example configures the following elements on the three PE devices as part of setting up the EVPN:

- An EVPN instance (EVI) named EVPN-RI using instance-type virtual-switch. The example enables protocols evpn in the instance.



NOTE: You can use another instance type instead of virtual-switch that the device supports for EVPN instances, such as instance-type evpn.

- A route distinguisher for the EVI that is unique on each PE device.
- A route target extended community for the EVPN instance using the vrf-target statement.



NOTE: With this statement, the device automatically sets import and export routing policies based on the specified community. Because this configuration uses the same route target value on all of the PE devices, they can share routes using those implicit

routing policies. The example doesn't need to explicitly configure import and export policies for route sharing.

This example configures the following elements on the two multihoming peer PE devices PE1 and PE2 to which CE1 is multihomed:

- The interfaces that connect to the multihomed CE.
- An Ethernet Segment (ES) identifier (ESI) associated with those interfaces. The ESI values must match on the multihoming peer PE devices.
- Single-active mode for ES operation.

See ["EVPN Multihoming Overview" on page 162](#) and ["Configuring EVPN-MPLS Active-Standby Multihoming" on page 280](#) for more detail on the required configuration elements and steps.

Note that we support configuring the following elements on EVPN PEs, but we don't include them in this configuration because the example focuses only on the basic required elements:

- In addition to an EVPN instance, you usually configure Layer 3 (L3) VRF (virtual routing and forwarding) routing instances on the EVPN PE devices using the `vrf` instance type. L3 VRF instances enable separation or route sharing among multiple tenants at different sites supported by the PEs in the EVPN fabric.
- If needed, you can configure multiple EVPN instances to set up more than one EVPN with the same set of PEs. In contrast with L3 VRF instances, multiple EVPN instances work at Layer 2 (L2) to further separate how the traffic can be forwarded within or routed between particular VLANs or bridge domains that the EVPN fabric supports.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 286](#)

CLI Quick Configuration

The configurations for each device are as follows:

CE1

```

interfaces {
  xe-0/0/0 {
    description to-PE1;
    unit 0 {
      family bridge {
        interface-mode trunk;
        vlan-id-list 10;
      }
    }
  }
  xe-0/0/2 {
    description to-PE2;
    unit 0 {
      family bridge {
        interface-mode trunk;
        vlan-id-list 10;
      }
    }
  }
  xe-0/0/7 {
    description to-Host;
    unit 0 {
      family bridge {
        interface-mode access;
        vlan-id 10;
      }
    }
  }
}
bridge-domains {
  BD {
    vlan-id-list 10;
    bridge-options {
      no-mac-learning; ## Used with single-active PE configurations, ensures traffic is
always flooded to both PEs in case of a DF change.
    }
  }
}

```


CE2

```

interfaces {
  xe-0/0/0 {
    description to-PE3;
    unit 0 {
      family bridge {
        interface-mode trunk;
        vlan-id-list 10;
      }
    }
  }
  xe-0/0/7 {
    description to-Host;
    unit 0 {
      family bridge {
        interface-mode access;
        vlan-id 10;
      }
    }
  }
}
bridge-domains {
  BD {
    vlan-id-list 10;
  }
}

```

PE1

```

interfaces {
  xe-0/0/0 {
    description to-CE1;
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
      00:11:22:33:44:55:66:77:88:99;
      single-active;
    }
    unit 10 {
      family bridge {

```

```

        interface-mode trunk;
        vlan-id-list 10;
    }
}
xe-0/0/3 {
    description to-P;
    unit 0 {
        family inet {
            address 10.3.3.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.1.1/32;
        }
    }
}
}
routing-options {
    router-id 192.168.1.1;
    autonomous-system 65432;
    forwarding-table {
        export evpn-pplb;
    }
}
protocols {
    rsvp {
        interface xe-0/0/3.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE1-to-PE2 {
            to 192.168.2.2;
        }
        label-switched-path PE1-to-PE3 {
            to 192.168.3.3;
        }
        interface xe-0/0/3.0;
    }
}

```

```

bgp {
    group EVPN-PE {
        type internal;
        local-address 192.168.1.1;
        family evpn {
            signaling;
        }
        neighbor 192.168.2.2;
        neighbor 192.168.3.3;
    }
}

ospf {
    area 0.0.0.0 {
        interface xe-0/0/3.0;
        interface lo0.0;
    }
}

}

policy-options {
    policy-statement evpn-pplb {
        from protocol evpn;
        then {
            load-balance per-packet;
        }
    }
}

routing-instances {
    EVPN-RI {
        instance-type virtual-switch;
        interface xe-0/0/0.10;
        route-distinguisher 192.168.1.1:10;
        vrf-target target:65432:10;
        protocols {
            evpn {
                extended-vlan-list 10;
            }
        }
        bridge-domains {
            bd10 {
                domain-type bridge;
                vlan-id 10;
            }
        }
    }
}

```

```

    }
}

```

PE2

```

interfaces {
  xe-0/0/0 {
    description to-P;
    unit 0 {
      family inet {
        address 10.5.5.1/30;
      }
      family mpls;
    }
  }
  xe-0/0/2 {
    description to-CE1;
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
      00:11:22:33:44:55:66:77:88:99;
      single-active;
    }
    unit 10 {
      family bridge {
        interface-mode trunk;
        vlan-id-list 10;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 192.168.2.2/32;
      }
    }
  }
}
routing-options {
  router-id 192.168.2.2;
  autonomous-system 65432;
  forwarding-table {

```

```

        export evpn-pplb;
    }
}
protocols {
    rsvp {
        interface xe-0/0/0.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE2-to-PE1 {
            to 192.168.1.1;
        }
        label-switched-path PE2-to-PE3 {
            to 192.168.3.3;
        }
        interface xe-0/0/0.0;
    }
    bgp {
        group EVPN-PE {
            type internal;
            local-address 192.168.2.2;
            family evpn {
                signaling;
            }
            neighbor 192.168.1.1;
            neighbor 192.168.3.3;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface xe-0/0/0.0;
            interface lo0.0;
        }
    }
}
policy-options {
    policy-statement evpn-pplb {
        from protocol evpn;
        then {
            load-balance per-packet;
        }
    }
}
}

```

```

routing-instances {
  EVPN-RI {
    instance-type virtual-switch;
    interface xe-0/0/2.10;
    route-distinguisher 192.168.2.2:10;
    vrf-target target:65432:10;
    protocols {
      evpn {
        extended-vlan-list 10;
      }
    }
    bridge-domains {
      bd10 {
        domain-type bridge;
        vlan-id 10;
      }
    }
  }
}

```

PE3

```

interfaces {
  xe-0/0/1 {
    description to-P;
    unit 0 {
      family inet {
        address 10.4.4.1/30;
      }
      family mpls;
    }
  }
  xe-0/0/2 {
    description to-CE3;
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 10 {
      family bridge {
        interface-mode trunk;
        vlan-id-list 10;
      }
    }
  }
}

```

```

}
lo0 {
    unit 0 {
        family inet {
            address 192.168.3.3/32;
        }
    }
}
}
routing-options {
    router-id 192.168.3.3;
    autonomous-system 65432;
    forwarding-table {
        export evpn-pplb;
    }
}
protocols {
    rsvp {
        interface xe-0/0/1.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE3-to-PE1 {
            to 192.168.1.1;
        }
        label-switched-path PE3-to-PE2 {
            to 192.168.2.2;
        }
        interface xe-0/0/1.0;
    }
    bgp {
        group EVPN-PE {
            type internal;
            local-address 192.168.3.3;
            family evpn {
                signaling;
            }
            neighbor 192.168.1.1;
            neighbor 192.168.2.2;
        }
    }
    ospf {
        area 0.0.0.0 {

```

```

        interface xe-0/0/1.0;
        interface lo0.0;
    }
}
}
policy-options {
    policy-statement evpn-pplb {
        from protocol evpn;
        then {
            load-balance per-packet;
        }
    }
}
routing-instances {
    EVPN-RI {
        instance-type virtual-switch;
        interface xe-0/0/2.10;
        route-distinguisher 192.168.3.3:10;
        vrf-target target:65432:10;
        protocols {
            evpn {
                extended-vlan-list 10;
            }
        }
        bridge-domains {
            bd10 {
                domain-type bridge;
                vlan-id 10;
            }
        }
    }
}
}

```

P1

```

interfaces {
    xe-0/0/0 {
        unit 0 {
            family inet {
                address 10.3.3.2/30;
            }
            family mpls;
        }
    }
}

```



```

    }
}
xe-0/0/1 {
    unit 0 {
        family inet {
            address 10.4.4.2/30;
        }
        family mpls;
    }
}
xe-0/0/2 {
    unit 0 {
        family inet {
            address 10.5.5.2/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.4.4/32;
        }
    }
}
}
routing-options {
    router-id 192.168.4.4;
    autonomous-system 65432;
}
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}

```

```
ospf {
  area 0.0.0.0 {
    interface xe-0/0/0.0;
    interface xe-0/0/1.0;
    interface xe-0/0/2.0;
    interface lo0.0;
  }
}
```

Verification

IN THIS SECTION

Verifying OSPF | 297

Verifying BGP | 298

Verifying MPLS | 299

Verifying EVPN Configuration and Multihoming Status | 301

Verifying Route Exchange and ESI Autodiscovery | 304

Verifying Ethernet Segment (ES) Route Exchange | 307

Confirm that the configuration is working properly.

Verifying OSPF

Purpose

Verify that OSPF is working properly.

Action

Verify that Router P1 has adjacencies established with all PE devices.

```
user@P1> show ospf neighbor
```

Address	Interface	State	ID	Pri	Dead
10.3.3.1	xe-0/0/0.0	Full	192.168.1.1	128	33

10.4.4.1	xe-0/0/1.0	Full	192.168.3.3	128	38
10.5.5.1	xe-0/0/2.0	Full	192.168.2.2	128	37

Meaning

Adjacencies have been established with the PE devices.

Verifying BGP

Purpose

Verify that BGP is working properly.

Action

Verify that MP-IBGP peerings are established using EVPN signaling between all PE devices.

```
user@PE1> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State   Pending
bgp.evpn.0
                4         4         0         0         0         0
Peer          AS      InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.168.2.2    65432      89       55       0       1    22:18 Establ
  EVPN-RI.evpn.0: 2/2/2/0
  bgp.evpn.0: 3/3/3/0
  __default_evpn__.evpn.0: 1/1/1/0
192.168.3.3    65432      59       48       0       1    22:18 Establ
  EVPN-RI.evpn.0: 1/1/1/0
  bgp.evpn.0: 1/1/1/0
  __default_evpn__.evpn.0: 0/0/0/0
```

```
user@PE2> show bgp summary

Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State   Pending
bgp.evpn.0
                5         5         0         0         0         0
Peer          AS      InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
```

```
Received/Accepted/Damped...
192.168.1.1      65432      80      50      0      1      22:49 Establ
  bgp.evpn.0: 4/4/4/0
  EVPN-RI.evpn.0: 3/3/3/0
  __default_evpn__.evpn.0: 1/1/1/0
192.168.3.3      65432      73      87      0      0      27:26 Establ
  bgp.evpn.0: 1/1/1/0
  EVPN-RI.evpn.0: 1/1/1/0
  __default_evpn__.evpn.0: 0/0/0/0

user@PE3> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State   Pending
bgp.evpn.0
              5          5          0          0          0          0
Peer          AS      InPkt    OutPkt    OutQ    Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.168.1.1      65432      66      51      0      1      23:05 Establ
  bgp.evpn.0: 3/3/3/0
  EVPN-RI.evpn.0: 3/3/3/0
  __default_evpn__.evpn.0: 0/0/0/0
192.168.2.2      65432     104      64      0      0      27:42 Establ
  bgp.evpn.0: 2/2/2/0
  EVPN-RI.evpn.0: 2/2/2/0
  __default_evpn__.evpn.0: 0/0/0/0
```

Meaning

EVPN-signaled MP-IBGP peerings have been established between all PE devices.

Verifying MPLS

Purpose

Verify that MPLS is working properly.

Action

Verify that MPLS LSPs are established between all PE devices.

```
user@PE1> show mpls lsp
```

Ingress LSP: 2 sessions

To	From	State	Rt	P	ActivePath	LSPname
192.168.2.2	192.168.1.1	Up	0	*		PE1-to-PE2
192.168.3.3	192.168.1.1	Up	0	*		PE1-to-PE3

Total 2 displayed, Up 2, Down 0

Egress LSP: 2 sessions

To	From	State	Rt	Style	Labelin	Labelout	LSPname
192.168.1.1	192.168.2.2	Up	0	1 FF	3	-	PE2-to-PE1
192.168.1.1	192.168.3.3	Up	0	1 FF	3	-	PE3-to-PE1

Total 2 displayed, Up 2, Down 0

Transit LSP: 0 sessions

Total 0 displayed, Up 0, Down 0

```
user@PE2> show mpls lsp
```

Ingress LSP: 2 sessions

To	From	State	Rt	P	ActivePath	LSPname
192.168.1.1	192.168.2.2	Up	0	*		PE2-to-PE1
192.168.3.3	192.168.2.2	Up	0	*		PE2-to-PE3

Total 2 displayed, Up 2, Down 0

Egress LSP: 2 sessions

To	From	State	Rt	Style	Labelin	Labelout	LSPname
192.168.2.2	192.168.3.3	Up	0	1 FF	3	-	PE3-to-PE2
192.168.2.2	192.168.1.1	Up	0	1 FF	3	-	PE1-to-PE2

Total 2 displayed, Up 2, Down 0

Transit LSP: 0 sessions

Total 0 displayed, Up 0, Down 0

```
user@PE3> show mpls lsp
```

Ingress LSP: 2 sessions

To	From	State	Rt	P	ActivePath	LSPname
192.168.1.1	192.168.3.3	Up	0	*		PE3-to-PE1

```

192.168.2.2    192.168.3.3    Up    0 *                                PE3-to-PE2
Total 2 displayed, Up 2, Down 0

Egress LSP: 2 sessions
To            From            State   Rt  Style Labelin Labelout LSPname
192.168.3.3   192.168.1.1   Up      0  1 FF      3      - PE1-to-PE3
192.168.3.3   192.168.2.2   Up      0  1 FF      3      - PE2-to-PE3
Total 2 displayed, Up 2, Down 0

Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

```

Meaning

LSPs have been established between PE devices.

Verifying EVPN Configuration and Multihoming Status

Purpose

Verify that EVPN is configured properly.

Action

Verify that the EVPN routing instances and ESIs are configured and functioning correctly, and confirm that single-active multihoming is enabled.

```

user@PE1> show evpn instance EVPN-RI extensive
Instance: EVPN-RI
Route Distinguisher: 192.168.1.1:10
Per-instance MAC route label: 300128
MAC database status          Local  Remote
MAC advertisements:         0      0
MAC+IP advertisements:      0      0
Default gateway MAC advertisements: 0      0
Number of local interfaces: 1 (1 up)
Interface name  ESI                                Mode      Status    AC-Role
xe-0/0/0.10    00:11:22:33:44:55:66:77:88:99  single-active  Up        Root
Number of IRB interfaces: 0 (0 up)
Number of bridge domains: 1
VLAN  Domain ID  Intfs / up  IRB intf  Mode      MAC sync  IM route label

```

```

10          1    1          Extended    Enabled    300240
Number of neighbors: 2
  Address      MAC      MAC+IP      AD      IM      ES Leaf-label
  192.168.2.2    0        0          1        1        0
  192.168.3.3    0        0          0        1        0
Number of ethernet segments: 1
ESI: 00:11:22:33:44:55:66:77:88:99
  Status: Resolved by IFL xe-0/0/0.10
  Local interface: xe-0/0/0.10, Status: Up/Forwarding
  Number of remote PEs connected: 1
    Remote PE      MAC label  Aliasing label  Mode
    192.168.2.2    0          0              single-active

```

Designated forwarder: 192.168.1.1

```

Backup forwarder: 192.168.2.2
Last designated forwarder update: Jun 26 23:30:35
Advertised MAC label: 300224
Advertised aliasing label: 300224
Advertised split horizon label: 300256

```

user@PE2> **show evpn instance EVPN-RI extensive**

Instance: EVPN-RI

Route Distinguisher: 192.168.2.2:10

Per-instance MAC route label: 300384

```

MAC database status          Local  Remote
MAC advertisements:         0       0
MAC+IP advertisements:      0       0
Default gateway MAC advertisements: 0       0

```

Number of local interfaces: 1 (1 up)

```

Interface name  ESI          Mode      Status    AC-Role
xe-0/0/2.10    00:11:22:33:44:55:66:77:88:99 single-active  Up      Root

```

Number of IRB interfaces: 0 (0 up)

Number of bridge domains: 1

```

VLAN  Domain ID  Intfs / up  IRB intf  Mode      MAC sync  IM route label
10          1    1          Extended    Enabled    300608

```

Number of neighbors: 2

```

Address      MAC      MAC+IP      AD      IM      ES Leaf-label
192.168.1.1    0        0          2        1        0
192.168.3.3    0        0          0        1        0

```

Number of ethernet segments: 1

ESI: 00:11:22:33:44:55:66:77:88:99

```
Status: Resolved by NH 1048575
Local interface: xe-0/0/2.10, Status: Up/Blocking
Number of remote PEs connected: 1
  Remote PE      MAC label  Aliasing label  Mode
  192.168.1.1    0           300224          single-active
```

Designated forwarder: 192.168.1.1

```
Backup forwarder: 192.168.2.2
Last designated forwarder update: Jun 26 23:30:43
Advertised MAC label: 300544
Advertised aliasing label: 300544
Advertised split horizon label: 300320
```

user@PE3> **show evpn instance EVPN-RI extensive**

Instance: EVPN-RI

Route Distinguisher: 192.168.3.3:10

Per-instance MAC route label: 300272

MAC database status	Local	Remote
MAC advertisements:	0	0
MAC+IP advertisements:	0	0
Default gateway MAC advertisements:	0	0

Number of local interfaces: 1 (1 up)

Interface name	ESI	Mode	Status	AC-Role
xe-0/0/2.10	00:00:00:00:00:00:00:00:00:00	single-homed	Up	Root

Number of IRB interfaces: 0 (0 up)

Number of bridge domains: 1

VLAN	Domain ID	Intfs / up	IRB intf	Mode	MAC sync	IM route label
10		1 1		Extended	Enabled	300368

Number of neighbors: 2

Address	MAC	MAC+IP	AD	IM	ES	Leaf-label
192.168.1.1	0	0	2	1	0	
192.168.2.2	0	0	1	1	0	

Number of ethernet segments: 1

ESI: 00:11:22:33:44:55:66:77:88:99

Status: Resolved by NH 1048574

Number of remote PEs connected: 2

Remote PE	MAC label	Aliasing label	Mode
192.168.1.1	0	300224	single-active
192.168.2.2	0	0	single-active

Meaning

From the outputs above, the following can be determined:

- All three PE devices confirm that PE1 and PE2 are using single-active mode.
- PE1 and PE2 are using the same ESI.
- PE1 is elected as the designated forwarder (DF), and its CE-facing interface is put into a state of **Up/Forwarding**.
- PE2 is elected as the backup designated forwarder (BDF), and its CE-facing interface is put into a state of **Up/Blocking**.

Verifying Route Exchange and ESI Autodiscovery

Purpose

Verify that EVPN signaling is working properly.

Action

Verify that autodiscovery and other signaling information is being shared between PE devices.

```
user@PE1> show route table EVPN-RI.evpn.0

EVPN-RI.evpn.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:192.168.1.1:10::112233445566778899::0/304 AD/EVI
    *[EVPN/170] 00:19:27
    Indirect
1:192.168.2.2:0::112233445566778899::FFFF:FFFF/304 AD/ESI
    *[BGP/170] 00:18:20, localpref 100, from 192.168.2.2
    AS path: I, validation-state: unverified
    > to 10.3.3.2 via xe-0/0/3.0, label-switched-path PE1-to-PE2
3:192.168.1.1:10::10::192.168.1.1/304 IM
    *[EVPN/170] 00:19:31
    Indirect
```

```

3:192.168.2.2:10::10::192.168.2.2/304 IM
    *[BGP/170] 00:18:19, localpref 100, from 192.168.2.2
    AS path: I, validation-state: unverified
    > to 10.3.3.2 via xe-0/0/3.0, label-switched-path PE1-to-PE2
3:192.168.3.3:10::10::192.168.3.3/304 IM
    *[BGP/170] 00:18:13, localpref 100, from 192.168.3.3
    AS path: I, validation-state: unverified
    > to 10.3.3.2 via xe-0/0/3.0, label-switched-path PE1-to-PE3

```

user@PE2> **show route table EVPN-RI.evpn.0**

EVPN-RI.evpn.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)

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

```

1:192.168.1.1:10::112233445566778899::0/304 AD/EVI
    *[BGP/170] 00:18:51, localpref 100, from 192.168.1.1
    AS path: I, validation-state: unverified
    > to 10.5.5.2 via xe-0/0/0.0, label-switched-path PE2-to-PE1
1:192.168.1.1:0::112233445566778899::FFFF:FFFF/304 AD/ESI
    *[BGP/170] 00:18:51, localpref 100, from 192.168.1.1
    AS path: I, validation-state: unverified
    > to 10.5.5.2 via xe-0/0/0.0, label-switched-path PE2-to-PE1
3:192.168.1.1:10::10::192.168.1.1/304 IM
    *[BGP/170] 00:18:51, localpref 100, from 192.168.1.1
    AS path: I, validation-state: unverified
    > to 10.5.5.2 via xe-0/0/0.0, label-switched-path PE2-to-PE1
3:192.168.2.2:10::10::192.168.2.2/304 IM
    *[EVPN/170] 00:18:45
    Indirect
3:192.168.3.3:10::10::192.168.3.3/304 IM
    *[BGP/170] 00:18:40, localpref 100, from 192.168.3.3
    AS path: I, validation-state: unverified
    > to 10.5.5.2 via xe-0/0/0.0, label-switched-path PE2-to-PE3

```

user@PE3> **show route table EVPN-RI.evpn.0**

EVPN-RI.evpn.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)

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

```

1:192.168.1.1:10::112233445566778899::0/304 AD/EVI
    *[BGP/170] 00:18:54, localpref 100, from 192.168.1.1

```

```

        AS path: I, validation-state: unverified
        > to 10.4.4.2 via xe-0/0/1.0, label-switched-path PE3-to-PE1
1:192.168.1.1:0::112233445566778899::FFFF:FFFF/304 AD/ESI
        *[BGP/170] 00:18:54, localpref 100, from 192.168.1.1
        AS path: I, validation-state: unverified
        > to 10.4.4.2 via xe-0/0/1.0, label-switched-path PE3-to-PE1
1:192.168.2.2:0::112233445566778899::FFFF:FFFF/304 AD/ESI
        *[BGP/170] 00:18:54, localpref 100, from 192.168.2.2
        AS path: I, validation-state: unverified
        > to 10.4.4.2 via xe-0/0/1.0, label-switched-path PE3-to-PE2
3:192.168.1.1:10::10::192.168.1.1/304 IM
        *[BGP/170] 00:18:54, localpref 100, from 192.168.1.1
        AS path: I, validation-state: unverified
        > to 10.4.4.2 via xe-0/0/1.0, label-switched-path PE3-to-PE1
3:192.168.2.2:10::10::192.168.2.2/304 IM
        *[BGP/170] 00:18:54, localpref 100, from 192.168.2.2
        AS path: I, validation-state: unverified
        > to 10.4.4.2 via xe-0/0/1.0, label-switched-path PE3-to-PE2
3:192.168.3.3:10::10::192.168.3.3/304 IM
        *[EVPN/170] 00:18:53
        Indirect

```

Meaning

The outputs above show two EVPN route types:

- Route Type 1: Ethernet Auto-Discovery (AD) Route - These routes are advertised on a per-EVI and per-ESI basis. Ethernet AD routes are required when a CE device is multihomed. When a CE device is single-homed, the ESI will be zero.
- Route Type 3: Inclusive Multicast Ethernet Tag Route - This route sets up a path for broadcast, unknown unicast, and multicast (BUM) traffic from a PE device to the remote PE device on a per VLAN, per ESI basis.

The outputs above show the following information:

- 1:192.168.x.x:10::112233445566778899::0/304 AD/EVI - This is the per-EVI AD Type 1 EVPN route. As the DF (and active device), Router PE1 has advertised this route to Routers PE2 and PE3.
- 1:192.168.x.x:0::112233445566778899::FFFF:FFFF/304 AD/ESI - This is the per Ethernet segment AD Type 1 EVPN route. As the multihomed devices, Routers PE1 and PE2 have advertised this route to each other and to Router PE3.

- 3:192.168.x.x:10::10::192.168.x.x/304 IM - This is the route used to set up a path for BUM traffic. Each PE device has advertised this route to the other PE device.

Verifying Ethernet Segment (ES) Route Exchange

Purpose

Verify that ES route information is being shared correctly.

Action

Verify that the local and advertised autodiscovery routes per Ethernet segment and the Ethernet segment routes are received.

```

user@PE1> show route table __default_evpn__.evpn.0

__default_evpn__.evpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:192.168.1.1:0::112233445566778899::FFFF:FFFF/304 AD/ESI
    *[EVPN/170] 00:14:22
    Indirect
4:192.168.1.1:0::112233445566778899:192.168.1.1/304 ES
    *[EVPN/170] 00:14:23
    Indirect
4:192.168.2.2:0::112233445566778899:192.168.2.2/304 ES
    *[BGP/170] 00:14:14, localpref 100, from 192.168.2.2
    AS path: I, validation-state: unverified
    > to 10.3.3.2 via xe-0/0/3.0, label-switched-path PE1-to-PE2

user@PE2> show route table __default_evpn__.evpn.0

__default_evpn__.evpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:192.168.2.2:0::112233445566778899::FFFF:FFFF/304 AD/ESI
    *[EVPN/170] 00:14:25
    Indirect
4:192.168.1.1:0::112233445566778899:192.168.1.1/304 ES
    *[BGP/170] 00:14:24, localpref 100, from 192.168.1.1
    AS path: I, validation-state: unverified

```

```

> to 10.5.5.2 via xe-0/0/0.0, label-switched-path PE2-to-PE1
4:192.168.2.2:0::112233445566778899:192.168.2.2/304 ES
*[EVPN/170] 00:14:26
Indirect

```

Meaning

The outputs above show two EVPN route types:

- Route Type 1: Ethernet Auto-Discovery (AD) Route - These routes are advertised on a per-EVI and per-ESI basis. Ethernet AD routes are required when a CE device is multihomed. When a CE device is single-homed, the ESI will be zero.
- Route Type 4: Ethernet Segment Route - PE devices that are connected to the same Ethernet Segment will discover each other through the ES route.

The outputs above show the following information:

- 1:192.168.x.x:0::112233445566778899::FFFF:FFFF/304 AD/ESI - This is the per Ethernet segment AD Type 1 EVPN route. In the outputs above, each PE device shows its own route.
- 4:192.168.x.x:0::112233445566778899:192.168.x.x/304 ES - This is the ES route for the local ESI. In the outputs above, each PE device shows both its own route and the one advertised by the other PE device.

RELATED DOCUMENTATION

[EVPN Multihoming Overview | 162](#)

[Configuring EVPN-MPLS Active-Standby Multihoming | 280](#)

[Example: Configuring Basic EVPN Active-Active Multihoming | 362](#)

[Example: Configuring EVPN Active-Active Multihoming | 375](#)

Example: Configuring EVPN-MPLS Active-Standby Multihoming

IN THIS SECTION

- [Requirements | 309](#)

- Overview and Topology | 310
- Configuration | 312
- Verification | 330

This example shows how to configure Ethernet VPN (EVPN) with MPLS for multihomed customer edge (CE) devices in active-standby redundancy mode. The steps in this example set up:

- An EVPN routing instance using the virtual-switch routing instance type
- Another EVPN routing instance using the evpn routing instance type
- A virtual routing and forwarding (VRF) routing instance using the vrf routing instance type as part of configuring integrated routing and bridging (IRB) interfaces.



NOTE: We support active-standby multihoming in EVPN fabrics only with MPLS. Please refer to [EVPN-MPLS: Single-active multihoming support](#) for a complete list of the products that support this feature.

Requirements

This example uses the following hardware and software components:

- Four MX Series 5G Universal Routing Platforms with MPC interfaces only, where:
 - Two devices are configured as provider edge (PE) routers connected to a common multihomed customer site.
 - One device is configured as a remote PE router connected to a single-homed customer site.
- Eight customer edge (CE) devices, where:
 - Two CE devices are multihomed.
 - Two CE devices are single-homed for each of the PE routers.
- Junos OS Release 14.1 or later running on all the PE routers.



NOTE: Junos OS Release 14.1 and later releases are based on the EVPN draft-ietf-12vpn-evpn-03. Releases prior to 14.1, support the older version of the EVPN draft,

causing interoperability issues when Junos OS Release 14.1 and a previous release are running.

Before you begin:

1. Configure the router interfaces.
2. Configure OSPF or any other IGP protocol.
3. Configure BGP.
4. Configure LDP.
5. Configure MPLS.
6. Configure RSVP MPLS LSP or GRE tunnels.

Overview and Topology

This EVPN solution on MX Series routers with MPC interfaces is extended to provide multihoming functionality with active-standby mode of operation. The multihoming functions include autodiscovery of Ethernet segments, Ethernet segment route construction, and Ethernet segment identifier (ESI) label assignment.



NOTE: In some older Junos releases, the EVPN functionality support on MX Series Routers was limited to routers using MPC and MIC interfaces only. See [Feature Explorer](#) for platform and release support for any Junos OS feature.

The DPC support for EVPN is provided with the following considerations:

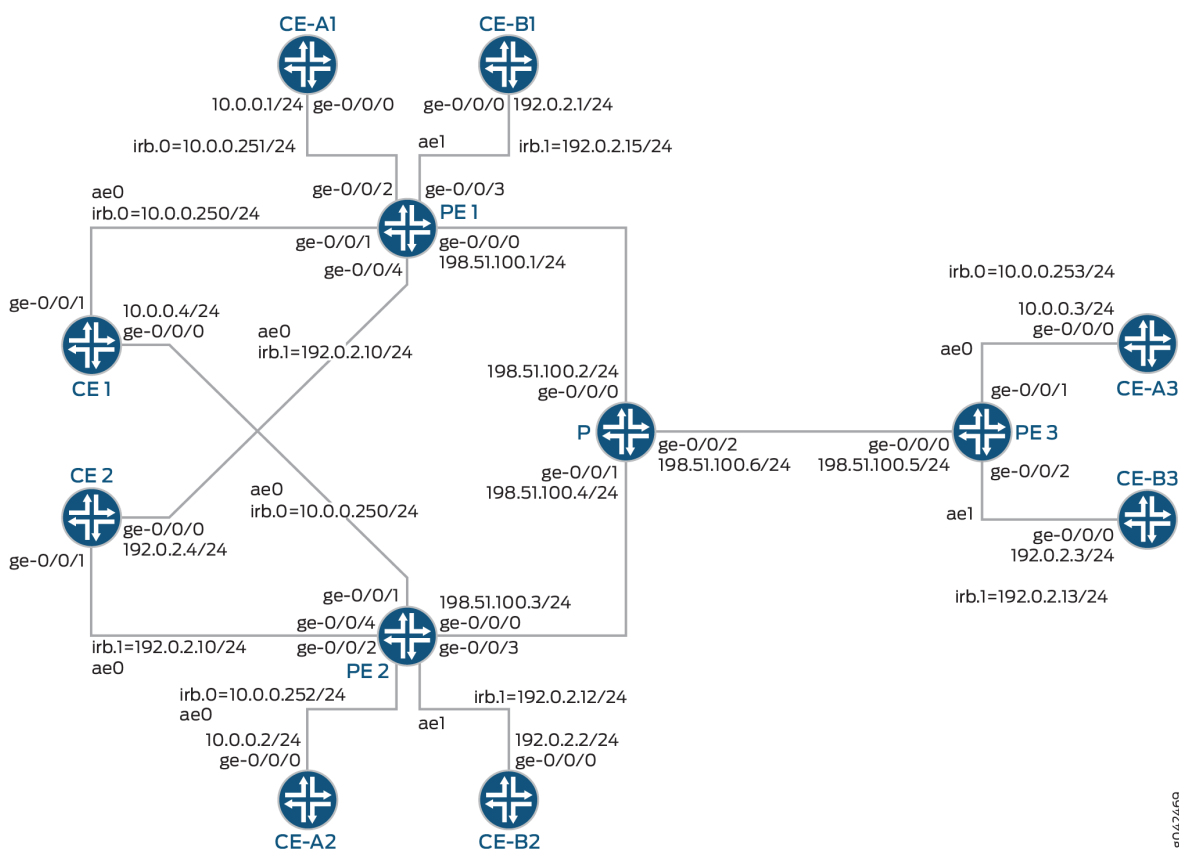
- DPCs provide support for EVPN in the active-standby mode of operation including support for the following:
 1. EVPN instance (EVI)
 2. Virtual switch (VS)
 3. Integrated routing and bridging (IRB) interfaces
- DPCs intended for providing the EVPN active-standby support should be the CE device-facing line card. The PE device interfaces in the EVPN domain should use only MPC and MIC interfaces.



NOTE: When configuring active-standby EVPN multihoming, be aware of the following limitations:

- An interface or ESI can be attached to more than one EVI, with a maximum limit of 200 EVIs per ESI.
- For an EVPN routing instance, only one logical interface per physical interface or ESI can be attached to an EVI.
- For a virtual switch routing instance, only one logical interface per physical interface or ESI can be configured under a bridge domain.
- All the PE routers in the network topology should be running Junos OS Release 14.1 or later, which are based on the EVPN draft-ietf-l2vpn-evpn-03. Junos OS releases prior to 14.1 support the older version of the EVPN draft, causing interoperability issues when Junos OS Release 14.1 and a previous release are running.

Figure 27: EVPN Active-Standby Multihoming



In Figure 27 on page 311:

- Routers PE1 and PE2 are provider edge (PE) routers connected to multihomed customer edge (CE) devices “Device CE1” and “Device CE1.”
- Router PE3 is a remote PE router connected to a single-homed customer site.
- Router P is the provider router connected to Routers PE1, PE2, and PE3.
- There are three routing instances running in the topology—ALPHA, BETA, and DELTA—with the virtual switch, EVPN, and VRF type of routing instance, respectively.
- All the PE routers are connected to one single-homed CE device each for the ALPHA and BETA routing instances.
- Device CE1 belongs to the ALPHA routing instance, and Device CE2 belongs to the BETA routing instance.

For Router PE1, Device CE-A1 and Device CE-B1 are the single-homed CE devices for the routing instances ALPHA and BETA, respectively. In the same way, Device CE-A2 and Device CE-A3 belong to the ALPHA routing instance, and Device CE-B2 and Device CE-B3 belong to the BETA routing instances connected to Routers PE2 and PE3, respectively.



NOTE: You can configure active-standby multihoming under any EVPN routing-instance. We support both `evpn` and `virtual-switch` instance types in active-standby EVPN multihoming. We include configuration for the tenant VRF routing instance of type `vrf` to illustrate the EVPN IRB functionality in addition to multihoming. The DELTA routing instance configuration is not required for the active-standby EVPN multihoming feature to work.

For simplicity and consistency of configuration, in this example we configure a route target extended community for the EVPN instances and VRF instance using the `vrf-target` statement at the `[edit routing-instances name]` hierarchy level. With this statement, the device automatically sets import and export routing policies based on the specified community. On each of the PE devices, we configure corresponding routing instances with route target values that match, so the PE devices can share routes using those implicit routing policies. We don't need to explicitly configure import and export policies for route sharing.

Configuration

IN THIS SECTION



CLI Quick Configuration | 313

- [Procedure | 320](#)
- [Results | 326](#)

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 chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ge-0/0/1 gigether-options 802.3ad ae0
set interfaces ae0 vlan-tagging
set interfaces ae0 mac 00:00:00:00:00:04
set interfaces ae0 unit 0 vlan-id 100
set interfaces ae0 unit 0 family inet address 10.0.0.4/24
set routing-options static route 0.0.0.0/0 next-hop 10.0.0.250
```

CE-A1

```
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ae0 vlan-tagging
set interfaces ae0 mac 00:00:00:00:00:01
set interfaces ae0 unit 0 vlan-id 100
set interfaces ae0 unit 0 family inet address 10.0.0.1/24
set routing-options static route 0.0.0.0/0 next-hop 10.0.0.251
```

CE-A2

```
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ae0 vlan-tagging
set interfaces ae0 mac 00:00:00:00:00:02
```

```
set interfaces ae0 unit 0 vlan-id 100
set interfaces ae0 unit 0 family inet address 10.0.0.2/24
set routing-options static route 0.0.0.0/0 next-hop 10.0.0.252
```

CE-A3

```
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ae0 vlan-tagging
set interfaces ae0 mac 00:00:00:00:00:03
set interfaces ae0 unit 0 vlan-id 100
set interfaces ae0 unit 0 family inet address 10.0.0.3/24
set routing-options static route 0.0.0.0/0 next-hop 10.0.0.253
```

CE2

```
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ge-0/0/1 gigether-options 802.3ad ae0
set interfaces ae0 vlan-tagging
set interfaces ae0 mac 00:00:00:00:00:04
set interfaces ae0 unit 0 vlan-id 300
set interfaces ae0 unit 0 family inet address 192.0.2.4/24
set routing-options static route 0.0.0.0/0 next-hop 192.0.2.10
```

CE-B1

```
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ae0 vlan-tagging
set interfaces ae0 mac 00:00:00:00:00:01
set interfaces ae0 unit 0 vlan-id 300
set interfaces ae0 unit 0 family inet address 192.0.2.1/24
set routing-options static route 0.0.0.0/0 next-hop 192.0.2.15
```

CE-B2

```
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ae0 vlan-tagging
```

```

set interfaces ae0 mac 00:00:00:00:00:02
set interfaces ae0 unit 0 vlan-id 300
set interfaces ae0 unit 0 family inet address 192.0.2.4/24
set routing-options static route 0.0.0.0/0 next-hop 192.0.2.12

```

CE-B3

```

set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ae0 vlan-tagging
set interfaces ae0 mac 00:00:00:00:00:03
set interfaces ae0 unit 0 vlan-id 300
set interfaces ae0 unit 0 family inet address 192.0.2.3/24
set routing-options static route 0.0.0.0/0 next-hop 192.0.2.13

```

PE1

```

set interfaces ge-0/0/0 unit 0 family inet address 198.51.100.1/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 gigether-options 802.3ad ae0
set interfaces ge-0/0/2 vlan-tagging
set interfaces ge-0/0/2 encapsulation flexible-ethernet-services
set interfaces ge-0/0/2 unit 0 encapsulation vlan-bridge
set interfaces ge-0/0/2 unit 0 vlan-id 100
set interfaces ge-0/0/3 gigether-options 802.3ad ae1
set interfaces ge-0/0/4 vlan-tagging
set interfaces ge-0/0/4 encapsulation flexible-ethernet-services
set interfaces ge-0/0/4 esi 00:22:44:66:88:00:22:44:66:88
set interfaces ge-0/0/4 esi single-active
set interfaces ge-0/0/4 unit 0 encapsulation vlan-bridge
set interfaces ge-0/0/4 unit 0 vlan-id 300
set interfaces ae0 vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 esi single-active
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 100
set interfaces ae1 vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 unit 0 encapsulation vlan-bridge
set interfaces ae1 unit 0 vlan-id 300

```

```

set interfaces irb unit 0 family inet address 10.0.0.250/24
set interfaces irb unit 0 family inet address 10.0.0.251/24
set interfaces irb unit 0 mac 00:11:22:33:44:55
set interfaces irb unit 1 family inet address 192.0.2.10/24
set interfaces irb unit 1 family inet address 192.0.2.15/24
set interfaces irb unit 1 mac 00:22:44:66:88:00
set interfaces lo0 unit 0 family inet address 10.255.0.1/32 primary
set interfaces lo0 unit 0 family inet address 10.255.0.1/32 preferred
set routing-options router-id 10.255.0.1
set routing-options autonomous-system 100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols mpls interface lo0.0
set protocols mpls interface ge-0/0/0.0
set protocols bgp group EVPN-PE type internal
set protocols bgp group EVPN-PE local-address 10.255.0.1
set protocols bgp group EVPN-PE family evpn signaling
set protocols bgp group EVPN-PE neighbor 10.255.0.2
set protocols bgp group EVPN-PE neighbor 10.255.0.3
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0
set routing-instances ALPHA instance-type virtual-switch
set routing-instances ALPHA route-distinguisher 10.255.0.1:100
set routing-instances ALPHA vrf-target target:100:100
set routing-instances ALPHA protocols evpn extended-vlan-list 100
set routing-instances ALPHA bridge-domains ONE domain-type bridge
set routing-instances ALPHA bridge-domains ONE vlan-id 100
set routing-instances ALPHA bridge-domains ONE interface ae0.0
set routing-instances ALPHA bridge-domains ONE interface ge-0/0/2.0
set routing-instances ALPHA bridge-domains ONE routing-interface irb.0
set routing-instances BETA instance-type evpn
set routing-instances BETA vlan-id 300
set routing-instances BETA interface ge-0/0/4.0
set routing-instances BETA interface ae1.0
set routing-instances BETA routing-interface irb.1
set routing-instances BETA route-distinguisher 10.255.0.1:300
set routing-instances BETA vrf-target target:300:300
set routing-instances DELTA instance-type vrf
set routing-instances DELTA interface irb.0
set routing-instances DELTA interface irb.1
set routing-instances DELTA route-distinguisher 10.255.0.1:200

```

```
set routing-instances DELTA vrf-target target:200:200
set routing-instances DELTA vrf-table-label
```

PE2

```
set interfaces ge-0/0/0 unit 0 family inet address 198.51.100.3/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 gigether-options 802.3ad ae0
set interfaces ge-0/0/2 vlan-tagging
set interfaces ge-0/0/2 encapsulation flexible-ethernet-services
set interfaces ge-0/0/2 unit 0 encapsulation vlan-bridge
set interfaces ge-0/0/2 unit 0 vlan-id 100
set interfaces ge-0/0/3 gigether-options 802.3ad ae1
set interfaces ge-0/0/4 vlan-tagging
set interfaces ge-0/0/4 encapsulation flexible-ethernet-services
set interfaces ge-0/0/4 esi 00:22:44:66:88:00:22:44:66:88
set interfaces ge-0/0/4 esi single-active
set interfaces ge-0/0/4 unit 0 encapsulation vlan-bridge
set interfaces ge-0/0/4 unit 0 vlan-id 300
set interfaces ae0 vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 esi single-active
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 100
set interfaces ae1 vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 unit 0 encapsulation vlan-bridge
set interfaces ae1 unit 0 vlan-id 300
set interfaces irb unit 0 family inet address 10.0.0.250/24
set interfaces irb unit 0 family inet address 10.0.0.252/24
set interfaces irb unit 0 mac 00:11:22:33:44:55
set interfaces irb unit 1 family inet address 192.0.2.10/24
set interfaces irb unit 1 family inet address 192.0.2.12/24
set interfaces irb unit 1 mac 00:22:44:66:88:00
set interfaces lo0 unit 0 family inet address 10.255.0.2/32 primary
set interfaces lo0 unit 0 family inet address 10.255.0.2/32 preferred
set routing-options router-id 10.255.0.2
set routing-options autonomous-system 100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols mpls interface lo0.0
set protocols mpls interface ge-0/0/0.0
```

```

set protocols bgp group EVPN-PE type internal
set protocols bgp group EVPN-PE local-address 10.255.0.2
set protocols bgp group EVPN-PE family evpn signaling
set protocols bgp group EVPN-PE neighbor 10.255.0.1
set protocols bgp group EVPN-PE neighbor 10.255.0.3
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0
set routing-instances ALPHA instance-type virtual-switch
set routing-instances ALPHA route-distinguisher 10.255.0.2:100
set routing-instances ALPHA vrf-target target:100:100
set routing-instances ALPHA protocols evpn extended-vlan-list 100
set routing-instances ALPHA bridge-domains ONE domain-type bridge
set routing-instances ALPHA bridge-domains ONE vlan-id 100
set routing-instances ALPHA bridge-domains ONE interface ae0.0
set routing-instances ALPHA bridge-domains ONE interface ge-0/0/2.0
set routing-instances ALPHA bridge-domains ONE routing-interface irb.0
set routing-instances BETA instance-type evpn
set routing-instances BETA vlan-id 300
set routing-instances BETA interface ge-0/0/4.0
set routing-instances BETA interface ae1.0
set routing-instances BETA routing-interface irb.1
set routing-instances BETA route-distinguisher 10.255.0.2:300
set routing-instances BETA vrf-target target:300:300
set routing-instances DELTA instance-type vrf
set routing-instances DELTA interface irb.0
set routing-instances DELTA interface irb.1
set routing-instances DELTA route-distinguisher 10.255.0.2:200
set routing-instances DELTA vrf-target target:200:200
set routing-instances DELTA vrf-table-label

```

PE3

```

set interfaces ge-0/0/0 unit 0 family inet address 198.51.100.5/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 gigether-options 802.3ad ae0
set interfaces ge-0/0/2 gigether-options 802.3ad ae1
set interfaces ae0 vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 100

```

```

set interfaces ae1 vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 unit 0 encapsulation vlan-bridge
set interfaces ae1 unit 0 vlan-id 300
set interfaces irb unit 0 family inet address 10.0.0.253/24
set interfaces irb unit 1 family inet address 192.0.2.13/24
set interfaces lo0 unit 0 family inet address 10.255.0.3/32 primary
set interfaces lo0 unit 0 family inet address 10.255.0.3/32 preferred
set routing-options router-id 10.255.0.3
set routing-options autonomous-system 100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols mpls interface lo0.0
set protocols mpls interface ge-0/0/0.0
set protocols bgp group EVPN-PE type internal
set protocols bgp group EVPN-PE local-address 10.255.0.3
set protocols bgp group EVPN-PE family evpn signaling
set protocols bgp group EVPN-PE neighbor 10.255.0.1
set protocols bgp group EVPN-PE neighbor 10.255.0.2
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0
set routing-instances ALPHA instance-type virtual-switch
set routing-instances ALPHA route-distinguisher 10.255.0.3:100
set routing-instances ALPHA vrf-target target:100:100
set routing-instances ALPHA protocols evpn extended-vlan-list 100
set routing-instances ALPHA bridge-domains ONE domain-type bridge
set routing-instances ALPHA bridge-domains ONE vlan-id 100
set routing-instances ALPHA bridge-domains ONE interface ae0.0
set routing-instances ALPHA bridge-domains ONE routing-interface irb.0
set routing-instances BETA instance-type evpn
set routing-instances BETA vlan-id 300
set routing-instances BETA interface ae1.0
set routing-instances BETA routing-interface irb.1
set routing-instances BETA route-distinguisher 10.255.0.3:300
set routing-instances BETA vrf-target target:300:300
set routing-instances DELTA instance-type vrf
set routing-instances DELTA interface irb.0
set routing-instances DELTA interface irb.1
set routing-instances DELTA route-distinguisher 10.255.0.3:200
set routing-instances DELTA vrf-target target:200:200
set routing-instances DELTA vrf-table-label

```


P

```

set interfaces ge-0/0/0 unit 0 family inet address 198.51.100.2/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 unit 0 family inet address 198.51.100.4/24
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces ge-0/0/2 unit 0 family inet address 198.51.100.6/24
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.0.4/32 primary
set interfaces lo0 unit 0 family inet address 10.255.0.4/32 preferred
set routing-options router-id 10.255.0.4
set protocols mpls interface lo0.0
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface ge-0/0/1.0
set protocols mpls interface ge-0/0/2.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface ge-0/0/1.0
set protocols ldp interface ge-0/0/2.0
set protocols ldp interface lo0.0

```

Procedure

Step-by-Step Procedure

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

To configure Router PE1:



NOTE: Repeat this procedure for Router PE2 after modifying the appropriate interface names, addresses, and other parameters.

1. Configure Router PE1 interfaces.

```
[edit interfaces]
```

```

user@PE1# set ge-0/0/0 unit 0 family inet address 198.51.100.1/24
user@PE1# set ge-0/0/0 unit 0 family mpls
user@PE1# set ge-0/0/1 gigether-options 802.3ad ae0
user@PE1# set ge-0/0/2 vlan-tagging
user@PE1# set ge-0/0/2 encapsulation flexible-ethernet-services
user@PE1# set ge-0/0/2 unit 0 encapsulation vlan-bridge
user@PE1# set ge-0/0/2 unit 0 vlan-id 100
user@PE1# set ge-0/0/3 gigether-options 802.3ad ae1
user@PE1# set ge-0/0/4 vlan-tagging
user@PE1# set ge-0/0/4 encapsulation flexible-ethernet-services
user@PE1# set ge-0/0/4 esi 00:22:44:66:88:00:22:44:66:88
user@PE1# set ge-0/0/4 esi single-active
user@PE1# set ge-0/0/4 unit 0 encapsulation vlan-bridge
user@PE1# set ge-0/0/4 unit 0 vlan-id 300
user@PE1# set ae0 vlan-tagging
user@PE1# set ae0 encapsulation flexible-ethernet-services
user@PE1# set ae0 esi 00:11:22:33:44:55:66:77:88:99
user@PE1# set ae0 esi single-active
user@PE1# set ae0 unit 0 encapsulation vlan-bridge
user@PE1# set ae0 unit 0 vlan-id 100
user@PE1# set ae1 vlan-tagging
user@PE1# set ae1 encapsulation flexible-ethernet-services
user@PE1# set ae1 unit 0 encapsulation vlan-bridge
user@PE1# set ae1 unit 0 vlan-id 300
user@PE1# set irb unit 0 family inet address 10.0.0.250/24
user@PE1# set irb unit 0 family inet address 10.0.0.251/24
user@PE1# set irb unit 0 mac 00:11:22:33:44:55
user@PE1# set irb unit 1 family inet address 192.0.2.10/24
user@PE1# set irb unit 1 family inet address 192.0.2.15/24
user@PE1# set irb unit 1 mac 00:22:44:66:88:00
user@PE1# set lo0 unit 0 family inet address 10.255.0.1/32 primary
user@PE1# set lo0 unit 0 family inet address 10.255.0.1/32 preferred

```

2. Configure the loopback address of Router PE1 as the router ID.

```

[edit routing-options]
user@PE1# set router-id 10.255.0.1

```

3. Set the autonomous system number for Router PE1.

```
[edit routing-options]  
user@PE1# set autonomous-system 100
```

4. Enable chained composite next hop for the EVPN.

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

5. Enable MPLS on the loopback interface of Router PE1 and the interface connecting PE1 to Router P.

```
[edit protocols]  
user@PE1# set mpls interface lo0.0  
user@PE1# set mpls interface ge-0/0/0.0
```

6. Configure the BGP group for Router PE1.

```
[edit protocols]  
user@PE1# set bgp group EVPN-PE type internal
```

7. Assign local and neighbor addresses to the EVPN-PE BGP group for Router PE1 to peer with Routers PE2 and PE3.

```
[edit protocols]  
user@PE1# set bgp group EVPN-PE local-address 10.255.0.1  
user@PE1# set bgp group EVPN-PE neighbor 10.255.0.2  
user@PE1# set bgp group EVPN-PE neighbor 10.255.0.3
```

8. Include the EVPN signaling Network Layer Reachability Information (NLRI) to the EVPN-PE group.

```
[edit protocols]
user@PE1# set bgp group EVPN-PE family evpn signaling
```

9. Configure OSPF on the loopback interface of Router PE1 and the interface connecting PE1 to Router P.

```
[edit protocols]
user@PE1# set ospf area 0.0.0.0 interface lo0.0 passive
user@PE1# set ospf area 0.0.0.0 interface ge-0/0/0.0
```

10. Enable LDP on the loopback interface of Router PE1 and the interface connecting PE1 to Router P.

```
[edit protocols]
user@PE1# set ldp interface lo0.0
user@PE1# set ldp interface ge-0/0/0.0
```

11. Configure the virtual switch routing instance – ALPHA.

```
[edit routing-instances]
user@PE1# set ALPHA instance-type virtual-switch
```

12. Configure the extended VLAN list for the ALPHA routing instance.

```
[edit routing-instances]
user@PE1# set ALPHA protocols evpn extended-vlan-list 100
```

13. Set the type for the bridging domain in the ALPHA routing instance.

```
[edit routing-instances]
user@PE1# set ALPHA bridge-domains ONE domain-type bridge
```

14. Set the VLAN for the bridging domain in the ALPHA routing instance.

```
[edit routing-instances]
user@PE1# set ALPHA bridge-domains ONE vlan-id 100
```

15. Configure the interface names for the ALPHA routing instance.

```
[edit routing-instances]
user@PE1# set ALPHA bridge-domains ONE interface ae0.0
user@PE1# set ALPHA bridge-domains ONE interface ge-0/0/2.0
user@PE1# set ALPHA bridge-domains ONE routing-interface irb.0
```

16. Configure the route distinguisher for the ALPHA routing instance.

```
[edit routing-instances]
user@PE1# set ALPHA route-distinguisher 10.255.0.1:100
```

17. Configure the VPN routing and forwarding (VRF) target community for the ALPHA routing instance.

```
[edit routing-instances]
user@PE1# set ALPHA vrf-target target:100:100
```

18. Configure the EVPN routing instance – BETA.

```
[edit routing-instances]
user@PE1# set BETA instance-type evpn
```

19. Set the VLAN identifier for the bridging domain in the BETA routing instance.

```
[edit routing-instances]
user@PE1# set BETA vlan-id 300
```

20. Configure the interface names for the BETA routing instance.

```
[edit routing-instances]
user@PE1# set BETA interface ge-0/0/4.0
```

```
user@PE1# set BETA interface ae1.0
user@PE1# set BETA routing-interface irb.1
```

21. Configure the route distinguisher for the BETA routing instance.

```
[edit routing-instances]
user@PE1# set BETA route-distinguisher 10.255.0.1:300
```

22. Configure the VPN routing and forwarding (VRF) target community for the BETA routing instance.

```
[edit routing-instances]
user@PE1# set BETA vrf-target target:300:300
```

23. Configure the VRF routing instance – DELTA.

```
[edit routing-instances]
user@PE1# set DELTA instance-type vrf
```

24. Configure the interface names for the DELTA routing instance.

```
[edit routing-instances]
user@PE1# set DELTA interface irb.0
user@PE1# set DELTA interface irb.1
```

25. Configure the route distinguisher for the DELTA routing instance.

```
[edit routing-instances]
user@PE1# set DELTA route-distinguisher 10.255.0.1:200
```

26. Configure the VPN routing and forwarding (VRF) target community for the DELTA routing instance.

```
[edit routing-instances]
user@PE1# set DELTA vrf-target target:200:200
user@PE1# set DELTA vrf-table-label
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show routing-options`, `show protocols`, 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-0/0/0 {
  unit 0 {
    family inet {
      address 198.51.100.1/24;
    }
    family mpls;
  }
}
ge-0/0/1 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/0/2 {
  vlan-tagging;
  encapsulation flexible-ethernet-services;
  unit 0 {
    encapsulation vlan-bridge;
    vlan-id 100;
  }
}
ge-0/0/3 {
  gigether-options {
    802.3ad ae1;
  }
}
ge-0/0/4 {
  vlan-tagging;
  encapsulation flexible-ethernet-services;
  esi {
    00:22:44:66:88:00:22:44:66:88;
```

```

        single-active;
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 300;
    }
}
ae0 {
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
        00:11:22:33:44:55:66:77:88:99;
        single-active;
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 100;
    }
}
ae1 {
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 300;
    }
}
irb {
    unit 0 {
        family inet {
            address 10.0.0.250/24;
            address 10.0.0.251/24;
        }
        mac 00:11:22:33:44:55;
    }
    unit 1 {
        family inet {
            address 192.0.2.10/24;
            address 192.0.2.15/24;
        }
        mac 00:22:44:66:88:00;
    }
}
}

```



```

lo0 {
  unit 0 {
    family inet {
      address 10.255.0.1/32 {
        primary;
        preferred;
      }
    }
  }
}

```

user@PE1#

show routing-options

```

router-id 10.255.0.1;
autonomous-system 100;
forwarding-table {
  chained-composite-next-hop {
    ingress {
      evpn;
    }
  }
}

```

user@PE1#

show protocols

```

mpls {
  interface lo0.0;
  interface ge-0/0/0.0;
}
bgp {
  group EVPN-PE {
    type internal;
    local-address 10.255.0.1;
    family evpn {

```

```

        signaling;
    }
    neighbor 10.255.0.2;
    neighbor 10.255.0.3;
}
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0;
    }
}
ldp {
    interface ge-0/0/0.0;
    interface lo0.0;
}

```

user@PE1#

show routing-instances

```

ALPHA {
    instance-type virtual-switch;
    route-distinguisher 10.255.0.1:100;
    vrf-target target:100:100;
    protocols {
        evpn {
            extended-vlan-list 100;
        }
    }
    bridge-domains {
        ONE {
            domain-type bridge;
            vlan-id 100;
            interface ae0.0;
            interface ge-0/0/2.0;
            routing-interface irb.0;
        }
    }
}

```

```

    }
}
BETA {
    instance-type evpn;
    vlan-id 300;
    interface ge-0/0/4.0;
    interface ae1.0;
    routing-interface irb.1;
    route-distinguisher 10.255.0.1:300;
    vrf-target target:300:300;
}
DELTA {
    instance-type vrf;
    interface irb.0;
    interface irb.1;
    route-distinguisher 10.255.0.1:200;
    vrf-target target:200:200;
    vrf-table-label;
}

```

Verification

IN THIS SECTION

- [Verifying the EVPN Instance Status | 331](#)
- [Verifying the Autodiscovery Routes per Ethernet Segment | 337](#)
- [Verifying the Ethernet Segment Route | 344](#)
- [Verifying the DF Status | 346](#)
- [Verifying the BDF Status | 346](#)
- [Verifying Remote IRB MAC | 347](#)
- [Verifying Remote IRB and Host IP | 348](#)
- [Verifying ARP Table | 352](#)
- [Verifying Bridge ARP Table | 353](#)
- [Verifying Bridge MAC Table | 354](#)

Confirm that the configuration is working properly in the following different areas on all the PE routers, where Router PE1 is the designated forwarder (DF), Router PE2 is the non-DF, and Router PE3 is the remote PE:

1. EVPN routing instance configuration
2. EVPN multihoming routes
3. DF election process
4. IRB and virtual switch routing instance configuration
5. Host route entries

Verifying the EVPN Instance Status

Purpose

Verify the EVPN routing instances and their status.

Action

Router PE1

From operational mode, run the **show evpn instance extensive** command.

```
user@PE1> show evpn instance extensive
Instance: ALPHA
Route Distinguisher: 10.255.0.1:100
Per-instance MAC route label: 300144
Per-instance multicast route label: 300160
MAC database status          Local  Remote
Total MAC addresses:         3      4
Default gateway MAC addresses: 1      2
Number of local interfaces: 2 (2 up)
Interface name  ESI                               Mode          SH label
ae0.0           00:11:22:33:44:55:66:77:88:99  single-active
ge-0/0/2.0      00:00:00:00:00:00:00:00:00:00  single-homed
Number of IRB interfaces: 1 (1 up)
Interface name  L3 context
irb.0           DELTA
Number of neighbors: 2
10.255.0.2
Received routes
```

```

    MAC address advertisement:      2
    MAC+IP address advertisement:    3
    Inclusive multicast:             1
    Ethernet auto-discovery:         1

```

10.255.0.3

Received routes

```

    MAC address advertisement:      2
    MAC+IP address advertisement:    2
    Inclusive multicast:             1
    Ethernet auto-discovery:         0

```

Number of ethernet segments: 1

ESI: 00:11:22:33:44:55:66:77:88:99

Designated forwarder: 10.255.0.1

Backup forwarder: 10.255.0.2

Instance: BETA

Route Distinguisher: 10.255.0.1:300

VLAN ID: 300

Per-instance MAC route label: 300176

Per-instance multicast route label: 300192

MAC database status Local Remote

Total MAC addresses: 3 4

Default gateway MAC addresses: 1 2

Number of local interfaces: 2 (2 up)

Interface name	ESI	Mode	SH label
ae1.0	00:00:00:00:00:00:00:00:00:00	single-homed	
ge-0/0/4.0	00:22:44:66:88:00:22:44:66:88	single-active	

Number of IRB interfaces: 1 (1 up)

Interface name L3 context

irb.1 DELTA

Number of neighbors: 2

10.255.0.2

Received routes

```

    MAC address advertisement:      2
    MAC+IP address advertisement:    3
    Inclusive multicast:             1
    Ethernet auto-discovery:         1

```

10.255.0.3

Received routes

```

    MAC address advertisement:      2
    MAC+IP address advertisement:    2
    Inclusive multicast:             1
    Ethernet auto-discovery:         0

```

```

Number of ethernet segments: 1
ESI: 00:22:44:66:88:00:22:44:66:88
  Designated forwarder: 10.255.0.1
  Backup forwarder: 10.255.0.2

Instance: __default_evpn__
Route Distinguisher: 10.255.0.1:0
VLAN ID: 0
Per-instance MAC route label: 300208
Per-instance multicast route label: 300224
MAC database status          Local  Remote
  Total MAC addresses:         0      0
  Default gateway MAC addresses: 0      0
Number of local interfaces: 0 (0 up)
Number of IRB interfaces: 0 (0 up)
Number of neighbors: 1
  10.255.0.2
  Received routes
    Ethernet auto-discovery:         0
    Ethernet Segment:                2
Number of ethernet segments: 0

```

Router PE2

From operational mode, run the **show evpn instance extensive** command.

```

user@PE2> show evpn instance extensive
Instance: ALPHA
Route Distinguisher: 10.255.0.2:100
Per-instance MAC route label: 300208
Per-instance multicast route label: 300224
MAC database status          Local  Remote
  Total MAC addresses:         2      5
  Default gateway MAC addresses: 1      2
Number of local interfaces: 2 (2 up)
  Interface name  ESI                                Mode          SH label
  ae0.0           00:11:22:33:44:55:66:77:88:99  single-active
  ge-0/0/2.0      00:00:00:00:00:00:00:00:00:00  single-homed
Number of IRB interfaces: 1 (1 up)
  Interface name  L3 context
  irb.0           DELTA
Number of neighbors: 2

```

10.255.0.1

Received routes

MAC address advertisement:	3
MAC+IP address advertisement:	4
Inclusive multicast:	1
Ethernet auto-discovery:	1

10.255.0.3

Received routes

MAC address advertisement:	2
MAC+IP address advertisement:	2
Inclusive multicast:	1
Ethernet auto-discovery:	0

Number of ethernet segments: 1

ESI: 00:11:22:33:44:55:66:77:88:99

Designated forwarder: 10.255.0.1

Backup forwarder: 10.255.0.2

Instance: BETA

Route Distinguisher: 10.255.0.2:300

VLAN ID: 300

Per-instance MAC route label: 300240

Per-instance multicast route label: 300256

MAC database status

	Local	Remote
--	-------	--------

Total MAC addresses:	2	5
----------------------	---	---

Default gateway MAC addresses:	1	2
--------------------------------	---	---

Number of local interfaces: 2 (2 up)

Interface name	ESI	Mode	SH label
ae1.0	00:00:00:00:00:00:00:00:00:00	single-homed	
ge-0/0/4.0	00:22:44:66:88:00:22:44:66:88	single-active	

Number of IRB interfaces: 1 (1 up)

Interface name L3 context

irb.1	DELTA
-------	-------

Number of neighbors: 2

10.255.0.1

Received routes

MAC address advertisement:	3
MAC+IP address advertisement:	4
Inclusive multicast:	1
Ethernet auto-discovery:	1

10.255.0.3

Received routes

MAC address advertisement:	2
MAC+IP address advertisement:	2

```

    Inclusive multicast:          1
    Ethernet auto-discovery:      0
Number of ethernet segments: 1
    ESI: 00:22:44:66:88:00:22:44:66:88
    Designated forwarder: 10.255.0.1
    Backup forwarder: 10.255.0.2

Instance: __default_evpn__
Route Distinguisher: 10.255.0.2:0
VLAN ID: 0
Per-instance MAC route label: 300272
Per-instance multicast route label: 300288
MAC database status          Local  Remote
    Total MAC addresses:          0      0
    Default gateway MAC addresses: 0      0
Number of local interfaces: 0 (0 up)
Number of IRB interfaces: 0 (0 up)
Number of neighbors: 1
    10.255.0.1
    Received routes
        Ethernet auto-discovery:      0
        Ethernet Segment:              2
Number of ethernet segments: 0

```

Router PE3

From operational mode, run the **show evpn instance extensive** command.

```

user@PE3> show evpn instance extensive
Instance: ALPHA
Route Distinguisher: 10.255.0.3:100
Per-instance MAC route label: 299776
Per-instance multicast route label: 299792
MAC database status          Local  Remote
    Total MAC addresses:          2      4
    Default gateway MAC addresses: 1      1
Number of local interfaces: 1 (1 up)
    Interface name  ESI                      Mode          SH label
    ae0.0           00:00:00:00:00:00:00:00:00:00 single-homed
Number of IRB interfaces: 1 (1 up)
    Interface name  L3 context
    irb.0           DELTA

```


Number of neighbors: 2

10.255.0.1

Received routes

MAC address advertisement:	3
MAC+IP address advertisement:	4
Inclusive multicast:	1
Ethernet auto-discovery:	1

10.255.0.2

Received routes

MAC address advertisement:	2
MAC+IP address advertisement:	3
Inclusive multicast:	1
Ethernet auto-discovery:	1

Number of ethernet segments: 0

Instance: BETA

Route Distinguisher: 10.255.0.3:300

VLAN ID: 300

Per-instance MAC route label: 299808

Per-instance multicast route label: 299824

MAC database status	Local	Remote
---------------------	-------	--------

Total MAC addresses:	2	4
----------------------	---	---

Default gateway MAC addresses:	1	1
--------------------------------	---	---

Number of local interfaces: 1 (1 up)

Interface name	ESI	Mode	SH label
ae1.0	00:00:00:00:00:00:00:00:00:00	single-homed	

Number of IRB interfaces: 1 (1 up)

Interface name	L3 context
----------------	------------

irb.1	DELTA
-------	-------

Number of neighbors: 2

10.255.0.1

Received routes

MAC address advertisement:	3
MAC+IP address advertisement:	4
Inclusive multicast:	1
Ethernet auto-discovery:	1

10.255.0.2

Received routes

MAC address advertisement:	2
MAC+IP address advertisement:	3
Inclusive multicast:	1
Ethernet auto-discovery:	1

Number of ethernet segments: 0

```

Instance: __default_evpn__
Route Distinguisher: 10.255.0.3:0
VLAN ID: 0
Per-instance MAC route label: 299840
Per-instance multicast route label: 299856
MAC database status          Local  Remote
Total MAC addresses:         0      0
Default gateway MAC addresses: 0      0
Number of local interfaces: 0 (0 up)
Number of IRB interfaces: 0 (0 up)
Number of neighbors: 0
Number of ethernet segments: 0

```

Meaning

The output provides the following information:

- List of EVPN and virtual switch routing instances.
- Mode of operation of each interface
- Neighbors of each routing instance.
- Number of different routes received from each neighbor.
- ESI attached to each routing instance.
- Number of Ethernet segments on each routing instance.
- DF election roles for each ESI in an EVI.
- VLAN ID and MAC labels for each routing instance.
- IRB interface details
- Number of default gateway MAC addresses received for the virtual switch routing instance (ALPHA).

Verifying the Autodiscovery Routes per Ethernet Segment

Purpose

Verify that the autodiscovery routes per Ethernet segment are received.

Action

Router PE1

From operational mode, run the **show route table ALPHA.evpn.0** command.

```
user@PE1> show route table ALPHA.evpn.0
ALPHA.evpn.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.0.2:0::112233445566778899::0/304
    *[BGP/170] 2d 23:51:27, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808
2:10.255.0.1:100::100::00:00:00:00:00:00:01/304
    *[EVPN/170] 2d 23:52:22
    Indirect
2:10.255.0.1:100::100::00:00:00:00:00:00:04/304
    *[EVPN/170] 2d 23:52:24
    Indirect
2:10.255.0.1:100::100::00:11:22:33:44:55/304
    *[EVPN/170] 2d 23:53:32
    Indirect
2:10.255.0.2:100::100::00:00:00:00:00:00:02/304
    *[BGP/170] 2d 23:51:27, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808
2:10.255.0.2:100::100::00:11:22:33:44:55/304
    *[BGP/170] 2d 23:51:27, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808
2:10.255.0.3:100::100::00:00:00:00:00:00:03/304
    *[BGP/170] 2d 23:39:04, localpref 100, from 10.255.0.3
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299824
2:10.255.0.3:100::100::00:05:86:71:b3:f0/304
    *[BGP/170] 2d 23:39:24, localpref 100, from 10.255.0.3
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299824
2:10.255.0.1:100::100::00:00:00:00:00:01::10.0.0.1/304
    *[EVPN/170] 2d 23:52:22
    Indirect
```

```

2:10.255.0.1:100::100::00:00:00:00:00:04::10.0.0.4/304
    *[EVPN/170] 2d 23:52:24
    Indirect
2:10.255.0.1:100::100::00:11:22:33:44:55::10.0.0.250/304
    *[EVPN/170] 2d 23:53:32
    Indirect
2:10.255.0.1:100::100::00:11:22:33:44:55::10.0.0.251/304
    *[EVPN/170] 2d 23:53:32
    Indirect
2:10.255.0.2:100::100::00:00:00:00:00:02::10.0.0.2/304
    *[BGP/170] 2d 23:51:27, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808
2:10.255.0.2:100::100::00:11:22:33:44:55::10.0.0.250/304
    *[BGP/170] 2d 23:51:27, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808
2:10.255.0.2:100::100::00:11:22:33:44:55::10.0.0.252/304
    *[BGP/170] 2d 23:51:27, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808
2:10.255.0.3:100::100::00:00:00:00:00:03::10.0.0.3/304
    *[BGP/170] 2d 23:39:04, localpref 100, from 10.255.0.3
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299824
2:10.255.0.3:100::100::00:05:86:71:b3:f0::10.0.0.253/304
    *[BGP/170] 2d 23:39:24, localpref 100, from 10.255.0.3
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299824
3:10.255.0.1:100::100::10.255.0.1/304
    *[EVPN/170] 2d 23:52:33
    Indirect
3:10.255.0.2:100::100::10.255.0.2/304
    *[BGP/170] 2d 23:51:27, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808
3:10.255.0.3:100::100::10.255.0.3/304
    *[BGP/170] 2d 23:39:24, localpref 100, from 10.255.0.3
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299824

```

Router PE2

From operational mode, run the **show route table ALPHA.evpn.0** command.

```

user@PE2> show show route table ALPHA.evpn.0
ALPHA.evpn.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.0.1:0::112233445566778899::0/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:00:00:00:00:01/304
    *[BGP/170] 10:43:51, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:00:00:00:00:04/304
    *[BGP/170] 10:45:06, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:11:22:33:44:55/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299776
2:10.255.0.2:100::100::00:00:00:00:00:02/304
    *[EVPN/170] 10:43:48
    Indirect
2:10.255.0.2:100::100::00:11:22:33:44:55/304
    *[EVPN/170] 10:46:04
    Indirect
2:10.255.0.3:100::100::00:00:00:00:00:03/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.3
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299792
2:10.255.0.3:100::100::00:05:86:71:79:f0/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.3
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299792
2:10.255.0.1:100::100::00:00:00:00:00:01::10.0.0.1/304
    *[BGP/170] 10:41:52, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:00:00:00:00:04::10.0.0.4/304
    *[BGP/170] 10:45:06, localpref 100, from 10.255.0.1

```

```

        AS path: I, validation-state: unverified
        > to 198.51.100.4 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:11:22:33:44:55::10.0.0.250/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.1
        AS path: I, validation-state: unverified
        > to 198.51.100.4 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:11:22:33:44:55::10.0.0.251/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.1
        AS path: I, validation-state: unverified
        > to 198.51.100.4 via ge-0/0/0.0, Push 299776
2:10.255.0.2:100::100::00:00:00:00:00:02::10.0.0.2/304
    *[EVPN/170] 10:40:25
        Indirect
2:10.255.0.2:100::100::00:11:22:33:44:55::10.0.0.250/304
    *[EVPN/170] 10:46:04
        Indirect
2:10.255.0.2:100::100::00:11:22:33:44:55::10.0.0.252/304
    *[EVPN/170] 10:46:04
        Indirect
2:10.255.0.3:100::100::00:00:00:00:00:03::10.0.0.3/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.3
        AS path: I, validation-state: unverified
        > to 198.51.100.4 via ge-0/0/0.0, Push 299792
2:10.255.0.3:100::100::00:05:86:71:79:f0::10.0.0.253/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.3
        AS path: I, validation-state: unverified
        > to 198.51.100.4 via ge-0/0/0.0, Push 299792
3:10.255.0.1:100::100::10.255.0.1/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.1
        AS path: I, validation-state: unverified
        > to 198.51.100.4 via ge-0/0/0.0, Push 299776
3:10.255.0.2:100::100::10.255.0.2/304
    *[EVPN/170] 10:46:04
        Indirect
3:10.255.0.3:100::100::10.255.0.3/304
    *[BGP/170] 10:46:04, localpref 100, from 10.255.0.3
        AS path: I, validation-state: unverified
        > to 198.51.100.4 via ge-0/0/0.0, Push 299792

```

Router PE3

From operational mode, run the **show route table ALPHA.evpn.0** command.

```

user@PE3> show route table ALPHA.evpn.0
ALPHA.evpn.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.0.1:0::112233445566778899::0/304
    *[BGP/170] 10:47:43, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ge-0/0/0.0, Push 299776
1:10.255.0.2:0::112233445566778899::0/304
    *[BGP/170] 10:47:34, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ge-0/0/0.0, Push 299808
2:10.255.0.1:100::100::00:00:00:00:00:01/304
    *[BGP/170] 10:45:21, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:00:00:00:00:04/304
    *[BGP/170] 10:46:36, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:11:22:33:44:55/304
    *[BGP/170] 10:47:43, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ge-0/0/0.0, Push 299776
2:10.255.0.2:100::100::00:00:00:00:00:02/304
    *[BGP/170] 10:45:18, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ge-0/0/0.0, Push 299808
2:10.255.0.2:100::100::00:11:22:33:44:55/304
    *[BGP/170] 10:47:34, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ge-0/0/0.0, Push 299808
2:10.255.0.3:100::100::00:00:00:00:00:03/304
    *[EVPN/170] 10:59:05
    Indirect
2:10.255.0.3:100::100::00:05:86:71:79:f0/304
    *[EVPN/170] 11:00:23
    Indirect
2:10.255.0.1:100::100::00:00:00:00:00:01::10.0.0.1/304
    *[BGP/170] 10:43:22, localpref 100, from 10.255.0.1

```

```

        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:00:00:00:04::10.0.0.4/304
    *[BGP/170] 10:46:36, localpref 100, from 10.255.0.1
        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:11:22:33:44:55::10.0.0.250/304
    *[BGP/170] 10:47:43, localpref 100, from 10.255.0.1
        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299776
2:10.255.0.1:100::100::00:11:22:33:44:55::10.0.0.251/304
    *[BGP/170] 10:47:43, localpref 100, from 10.255.0.1
        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299776
2:10.255.0.2:100::100::00:00:00:00:02::10.0.0.2/304
    *[BGP/170] 10:41:55, localpref 100, from 10.255.0.2
        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299808
2:10.255.0.2:100::100::00:11:22:33:44:55::10.0.0.250/304
    *[BGP/170] 10:47:34, localpref 100, from 10.255.0.2
        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299808
2:10.255.0.2:100::100::00:11:22:33:44:55::10.0.0.252/304
    *[BGP/170] 10:47:34, localpref 100, from 10.255.0.2
        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299808
2:10.255.0.3:100::100::00:00:00:00:03::10.0.0.3/304
    *[EVPN/170] 10:59:05
        Indirect
2:10.255.0.3:100::100::00:05:86:71:79:f0::10.0.0.253/304
    *[EVPN/170] 11:00:23
        Indirect
3:10.255.0.1:100::100::10.255.0.1/304
    *[BGP/170] 10:47:43, localpref 100, from 10.255.0.1
        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299776
3:10.255.0.2:100::100::10.255.0.2/304
    *[BGP/170] 10:47:34, localpref 100, from 10.255.0.2
        AS path: I, validation-state: unverified
        > to 198.51.100.6 via ge-0/0/0.0, Push 299808
3:10.255.0.3:100::100::10.255.0.3/304

```



```
*[EVPN/170] 10:59:40
    Indirect
```

Meaning

The remote type 1 autodiscovery route is received for the ESI attached to Router PE2, which is the other PE router connected to the multihomed CE device.

Verifying the Ethernet Segment Route

Purpose

Verify that the local and advertised autodiscovery routes per Ethernet segment and the Ethernet segment routes are received.

Action

Router PE1

From operational mode, run the **show route table __default_evpn__.evpn.0** command.

```
user@PE1> show route table __default_evpn__.evpn.0
__default_evpn__.evpn.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.0.1:0::112233445566778899::0/304
    *[EVPN/170] 3d 00:00:31
    Indirect
1:10.255.0.1:0::224466880022446688::0/304
    *[EVPN/170] 3d 00:00:31
    Indirect
4:10.255.0.1:0::112233445566778899:10.255.0.1/304
    *[EVPN/170] 3d 00:00:31
    Indirect
4:10.255.0.1:0::224466880022446688:10.255.0.1/304
    *[EVPN/170] 3d 00:00:31
    Indirect
4:10.255.0.2:0::112233445566778899:10.255.0.2/304
    *[BGP/170] 3d 00:00:20, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808
```

```

4:10.255.0.2:0::224466880022446688:10.255.0.2/304
    *[BGP/170] 3d 00:00:20, localpref 100, from 10.255.0.2
    AS path: I, validation-state: unverified
    > to 198.51.100.2 via ge-0/0/0.0, Push 299808

```

Router PE2

From operational mode, run the **show route table __default_evpn__.evpn.0** command.

```

user@PE2> show route table __default_evpn__.evpn.0
__default_evpn__.evpn.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.0.2:0::112233445566778899::0/304
    *[EVPN/170] 10:49:26
    Indirect
1:10.255.0.2:0::224466880022446688::0/304
    *[EVPN/170] 10:49:26
    Indirect
4:10.255.0.1:0::112233445566778899:10.255.0.1/304
    *[BGP/170] 10:49:26, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299776
4:10.255.0.1:0::224466880022446688:10.255.0.1/304
    *[BGP/170] 10:49:26, localpref 100, from 10.255.0.1
    AS path: I, validation-state: unverified
    > to 198.51.100.4 via ge-0/0/0.0, Push 299776
4:10.255.0.2:0::112233445566778899:10.255.0.2/304
    *[EVPN/170] 10:49:26
    Indirect
4:10.255.0.2:0::224466880022446688:10.255.0.2/304
    *[EVPN/170] 10:49:26
    Indirect

```

Meaning

The output displays the local and remote type 1 (autodiscovery) and type 4 (Ethernet segment) routes:

- 1:10.255.0.1:0::112233445566778899::0/304—Autodiscovery route per Ethernet segment for each local ESI attached to Router PE1 and Router PE2.

- 4:10.255.0.1:0::112233445566778899:10.255.0.1/304—Ethernet segment route for each local ESI attached to Router PE1 and Router PE2.
- 4:10.255.0.2:0::112233445566778899:10.255.0.2/304—Remote Ethernet segment route from Router PE2.

Verifying the DF Status

Purpose

Confirm which PE router is the designated forwarder (DF).

Action

From operational mode, run the **show evpn instance ALPHA esi esi/designated-forwarder** command.

```
user@PE1> show evpn instance ALPHA esi 00:11:22:33:44:55:66:77:88:99
designated-forwarder
Instance: ALPHA
  Number of ethernet segments: 1
    ESI: 00:11:22:33:44:55:66:77:88:99
      Designated forwarder: 10.255.0.1
```

Meaning

Router PE1 is the DF for the ALPHA routing instance.

Verifying the BDF Status

Purpose

Confirm which PE router is the backup designated forwarder (BDF).

Action

From operational mode, run the **show evpn instance ALPHA esi esi/backup-forwarder** command.

```
user@PE1> show evpn instance ALPHA esi 00:11:22:33:44:55:66:77:88:99
backup-forwarder
Instance: ALPHA
```

```

Number of ethernet segments: 1
ESI: 00:11:22:33:44:55:66:77:88:99
Backup forwarder: 10.255.0.2

```

Meaning

Router PE2 is the BDF for the ALPHA routing instance.

Verifying Remote IRB MAC

Purpose

Verify that the remote gateway MAC addresses are synchronized among all the PE routers.

Action

Router PE1

From operational mode, run the **show bridge evpn peer-gateway-mac** command.

```

user@PE1> show bridge evpn peer-gateway-mac
Routing instance : ALPHA
Bridging domain : ONE, VLAN : 100
Installed GW MAC addresses:
00:05:86:71:79:f0

```

Router PE2

From operational mode, run the **show bridge evpn peer-gateway-mac** command.

```

user@PE2> show bridge evpn peer-gateway-mac
Routing instance : ALPHA
Bridging domain : ONE, VLAN : 100
Installed GW MAC addresses:
00:05:86:71:79:f0

```

Router PE3

From operational mode, run the **show bridge evpn peer-gateway-mac** command.

```
user@PE2> show bridge evpn peer-gateway-mac
Routing instance : ALPHA
Bridging domain : ONE, VLAN : 100
Installed GW MAC addresses:
00:11:22:33:44:55
```

Meaning

The remote gateway MAC addresses are synchronized:

- Router PE3 gateway MAC is installed in Routers PE1 and PE2 peer-gateway-mac table.
- Routers PE1 and PE2 gateway MAC addresses are installed in Router PE3 peer-gateway-mac table.

Verifying Remote IRB and Host IP

Purpose

Verify that the remote IRB IP and the host IP are received.

Action

Router PE1

From operational mode, run the **show route table DELTA** command.

```
user@PE1> show route table DELTA
DELTA.inet.0: 16 destinations, 18 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.0/24      *[Direct/0] 11:25:54
                 > via irb.0
                 [Direct/0] 11:25:54
                 > via irb.0
10.0.0.1/32     *[EVPN/7] 11:21:33
                 > via irb.0
10.0.0.2/32     *[EVPN/7] 11:20:06, metric2 1
                 > to 198.51.100.2 via ge-0/0/0.0, Push 300208, Push 299808(top)
10.0.0.3/32     *[EVPN/7] 11:25:54, metric2 1
```

```

> to 198.51.100.2 via ge-0/0/0.0, Push 299776, Push 299792(top)
10.0.0.4/32      *[EVPN/7] 11:24:47
> via irb.0
10.0.0.250/32   *[Local/0] 11:38:29
                Local via irb.0
10.0.0.251/32   *[Local/0] 11:38:29
                Local via irb.0
10.0.0.253/32   *[EVPN/7] 11:25:54, metric2 1
> to 198.51.100.2 via ge-0/0/0.0, Push 299776, Push 299792(top)
192.0.2.0/24    *[Direct/0] 11:25:55
> via irb.1
                [Direct/0] 11:25:55
> via irb.1
192.0.2.1/24    *[EVPN/7] 11:21:20
> via irb.1
192.0.2.2/24    *[EVPN/7] 11:19:54, metric2 1
> to 198.51.100.2 via ge-0/0/0.0, Push 300240, Push 299808(top)
192.0.2.3/24    *[EVPN/7] 11:25:54, metric2 1
> to 198.51.100.2 via ge-0/0/0.0, Push 299808, Push 299792(top)
192.0.2.4/24    *[EVPN/7] 11:24:40
> via irb.1
192.0.2.10/24   *[Local/0] 11:38:29
                Local via irb.1
192.0.2.15/24   *[Local/0] 11:38:29
                Local via irb.1
192.0.2.13/24   *[EVPN/7] 11:25:54, metric2 1
> to 198.51.100.2 via ge-0/0/0.0, Push 299808, Push 299792(top)

```

Router PE2

From operational mode, run the **show route table DELTA** command.

```

user@PE2> show route table DELTA
DELTA.inet.0: 16 destinations, 18 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.0/24      *[Direct/0] 11:30:02
> via irb.0
                [Direct/0] 11:30:02
> via irb.0
10.0.0.1/32      *[EVPN/7] 11:25:50, metric2 1
> to 198.51.100.4 via ge-0/0/0.0, Push 300144, Push 299776(top)

```

```

10.0.0.2/32      *[EVPN/7] 11:24:23
                  > via irb.0
10.0.0.3/32      *[EVPN/7] 11:30:02, metric2 1
                  > to 198.51.100.4 via ge-0/0/0.0, Push 299776, Push 299792(top)
10.0.0.4/32      *[EVPN/7] 11:29:04, metric2 1
                  > to 198.51.100.4 via ge-0/0/0.0, Push 300144, Push 299776(top)
10.0.0.250/32    *[Local/0] 11:42:33
                  Local via irb.0
10.0.0.252/32    *[Local/0] 11:42:33
                  Local via irb.0
10.0.0.253/32    *[EVPN/7] 11:30:02, metric2 1
                  > to 198.51.100.4 via ge-0/0/0.0, Push 299776, Push 299792(top)
192.0.2.0/24     *[Direct/0] 11:30:02
                  > via irb.1
                  [Direct/0] 11:30:02
                  > via irb.1
192.0.2.1/24     *[EVPN/7] 11:25:37, metric2 1
                  > to 198.51.100.4 via ge-0/0/0.0, Push 300176, Push 299776(top)
192.0.2.2/24     *[EVPN/7] 11:24:11
                  > via irb.1
192.0.2.3/24     *[EVPN/7] 11:30:02, metric2 1
                  > to 198.51.100.4 via ge-0/0/0.0, Push 299808, Push 299792(top)
192.0.2.4/24     *[EVPN/7] 11:28:57, metric2 1
                  > to 198.51.100.4 via ge-0/0/0.0, Push 300176, Push 299776(top)
192.0.2.10/24    *[Local/0] 11:42:33
                  Local via irb.1
192.0.2.12/24    *[Local/0] 11:42:33
                  Local via irb.1
192.0.2.13/24    *[EVPN/7] 11:30:02, metric2 1
                  > to 198.51.100.4 via ge-0/0/0.0, Push 299808, Push 299792(top)

```

Router PE3

From operational mode, run the **show route table DELTA** command.

```

user@PE3> show route table DELTA
DELTA.inet.0: 16 destinations, 16 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.0/24      *[Direct/0] 11:42:15
                  > via irb.0
10.0.0.1/32      *[EVPN/7] 11:25:56, metric2 1

```

```

> to 198.51.100.6 via ge-0/0/0.0, Push 300144, Push 299776(top)
10.0.0.2/32      *[EVPN/7] 11:24:29, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300208, Push 299808(top)
10.0.0.3/32      *[EVPN/7] 11:41:39
> via irb.0
10.0.0.4/32      *[EVPN/7] 11:29:10, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300144, Push 299776(top)
10.0.0.250/32    *[EVPN/7] 11:30:08, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300208, Push 299808(top)
10.0.0.252/32    *[EVPN/7] 11:30:08, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300208, Push 299808(top)
10.0.0.253/32    *[Local/0] 11:42:57
Local via irb.0
192.0.2.0/24     *[Direct/0] 11:42:15
> via irb.1
192.0.2.1/24     *[EVPN/7] 11:25:43, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300176, Push 299776(top)
192.0.2.2/24     *[EVPN/7] 11:24:17, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300240, Push 299808(top)
192.0.2.3/24     *[EVPN/7] 11:42:04
> via irb.1
192.0.2.4/24     *[EVPN/7] 11:29:03, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300176, Push 299776(top)
192.0.2.10/24    *[EVPN/7] 11:30:08, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300240, Push 299808(top)
192.0.2.12/24    *[EVPN/7] 11:30:08, metric2 1
> to 198.51.100.6 via ge-0/0/0.0, Push 300240, Push 299808(top)
192.0.2.13/24    *[Local/0] 11:42:57
Local via irb.1

```

Meaning

The output displays the local and remote IRB interfaces. It also displays the local and remote hosts that are installed in the VRF table:

On Router PE1:

- 10.0.0.1/32—Local host in the virtual switch routing instance.
- 10.0.0.2/32 and 10.0.0.3/32—Remote host in the virtual switch routing instance.
- 10.0.0.250/32—Local IRB in the virtual switch routing instance.
- 10.0.0.253/32—Remote IRB in the virtual switch routing instance.

Verifying ARP Table

Purpose

Verify the ARP table entries.

Action

Router PE1

From operational mode, run the **show evpn arp-table** command.

```
user@PE1> show evpn arp-table
```

INET	MAC	Logical	Routing	Bridging
address	address	interface	instance	domain
192.0.2.1	00:00:00:00:00:01	irb.1	BETA	__BETA__
192.0.2.4	00:00:00:00:00:04	irb.1	BETA	__BETA__

Router PE2

From operational mode, run the **show evpn arp-table** command.

```
user@PE2> show evpn arp-table
```

INET	MAC	Logical	Routing	Bridging
address	address	interface	instance	domain
192.0.2.2	00:00:00:00:00:02	irb.1	BETA	__BETA__

Router PE3

From operational mode, run the **show evpn arp-table** command.

```
user@PE3> show evpn arp-table
```

INET	MAC	Logical	Routing	Bridging
address	address	interface	instance	domain
192.0.2.3	00:00:00:00:00:03	irb.1	BETA	__BETA__

Meaning

The EVPN instance and ARP are synchronized with the host MAC and IP address for local hosts.

Verifying Bridge ARP Table

Purpose

Verify the bridge ARP table entries.

Action

Router PE1

From operational mode, run the **show bridge evpn arp-table** command.

```
user@PE3> show bridge evpn arp-table
```

INET	MAC	Logical	Routing	Bridging
address	address	interface	instance	domain
10.0.0.1	00:00:00:00:00:01	irb.0	ALPHA	ONE
10.0.0.4	00:00:00:00:00:04	irb.0	ALPHA	ONE

Router PE2

From operational mode, run the **show bridge evpn arp-table** command.

```
user@PE3> show bridge evpn arp-table
```

INET	MAC	Logical	Routing	Bridging
address	address	interface	instance	domain
10.0.0.2	00:00:00:00:00:02	irb.0	ALPHA	ONE

Router PE3

From operational mode, run the **show bridge evpn arp-table** command.

```
user@PE3> show bridge evpn arp-table
```

INET	MAC	Logical	Routing	Bridging
address	address	interface	instance	domain
10.0.0.3	00:00:00:00:00:03	irb.0	ALPHA	ONE

Meaning

The virtual switch instance and ARP are synchronized with the local host MAC and IP address.

Verifying Bridge MAC Table

Purpose

Verify the bridge MAC table entries.

Action

Router PE1

From operational mode, run the **show bridge mac-table** command.

```
user@PE1> show bridge 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 : ALPHA
Bridging domain : ONE, VLAN : 100
```

MAC address	MAC flags	Logical interface	NH Index	RTR ID
00:00:00:00:00:01	D	ge-0/0/2.0		
00:00:00:00:00:02	DC		1048583	1048583
00:00:00:00:00:03	DC		1048574	1048574
00:00:00:00:00:04	D	ae0.0		

Router PE2

From operational mode, run the **show bridge mac-table** command.

```
user@PE2> show bridge 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 : ALPHA
Bridging domain : ONE, VLAN : 100
```

MAC address	MAC flags	Logical interface	NH Index	RTR ID
00:00:00:00:00:01	DC		1048577	1048577
00:00:00:00:00:02	D	ge-0/0/2.0		
00:00:00:00:00:03	DC		1048578	1048578
00:00:00:00:00:04	DC		1048577	1048577

Router PE3

From operational mode, run the **show bridge mac-table** command.

```

user@PE3> show bridge 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 : ALPHA
Bridging domain : ONE, VLAN : 100

```

MAC address	MAC flags	Logical interface	NH Index	RTR ID
00:00:00:00:00:01	DC		1048575	1048575
00:00:00:00:00:02	DC		1048582	1048582
00:00:00:00:00:03	D	ae0.0		
00:00:00:00:00:04	DC		1048575	1048575

Meaning

The virtual switch instance installed local and remote host MAC addresses in the bridge MAC table.

RELATED DOCUMENTATION

- [EVPN Multihoming Overview | 162](#)
- [Configuring EVPN-MPLS Active-Standby Multihoming | 280](#)

Example: Configuring EVPN-MPLS Active-Standby Multihoming with ESI

SUMMARY

This example covers the steps necessary to configure Ethernet VPN-MPLS (EVPN-MPLS) active-standby multihoming using either Ethernet Segment Identifier (ESI) per physical interface (IFD) with Link Aggregation Control Protocol (LACP), or ESI per

IN THIS SECTION

- [Example Prerequisites | 356](#)
- [Configure ESI per IFD with LACP | 357](#)
- [Configure ESI per IFL with CFM | 359](#)

logical interface (IFL) with Connectivity Fault Management (CFM).

- [Verification | 362](#)
- [Release Information | 362](#)

Understanding LACP and CFM in EVPN-MPLS Active-Standby Multihomed Environments

EVPN-MPLS active-standby multihoming with ESI leverages advanced networking techniques to provide robust redundancy and high availability. By configuring active-standby roles, you ensure that traffic is consistently forwarded through a single active link while multiple standby links remain ready to take over in case of a failure. This setup is critical for maintaining uninterrupted service and ensuring network resilience. The feature supports both LACP-based and CFM-based solutions, each tailored to different interface types—aggregated ethernet (AE) and non-AE interfaces, respectively. We support active-standby multihoming in EVPN fabrics only with MPLS.

For the LACP-based solution, you configure the system to use LACP out-of-sync mechanisms. This approach communicates the non-designated forwarder (DF) state to the customer edge (CE) device, ensuring it does not use non-DF links for traffic forwarding. The configuration involves setting the DF election granularity and enabling LACP out-of-sync for non-DF states via specific CLI commands. This method provides effective load balancing and redundancy management, making it ideal for environments using AE interfaces. The LACP-based solution is used for ESI per IFD in EVPN deployments. ESI per IFL is not supported with LACP in active-standby EVPN-MPLS deployments, since LACP will bring down the affected AE and all of its member links.

The CFM-based solution manages interface status transitions on non-AE interfaces. By configuring CFM, you enable the CE to detect DF/non-DF transitions through interface-status-tlv messages, which adjust the interface status accordingly. This method ensures robust fault management and status monitoring, crucial for scenarios where non-AE interfaces are used. Specific CLI commands are used to set up the CFM maintenance domain (MD), maintenance association (MA), and maintenance end point (MEP) configurations, providing a detailed framework for managing high availability and redundancy. The CFM-based solution is used for ESI per IFL in EVPN deployments. CFM is not supported if the CE uses an AE for the multihomed interface.

Example Prerequisites

You can refer to the following table for platform and release requirements specific to this example. You can use the same platform for all devices, or a mix of platforms.

Hardware requirements	<ul style="list-style-type: none"> • ACX Series routers • MX Series routers • PTX10001-36MR, PTX10004, PTX10008, and PTX10016 routers • All listed platforms support ESI per IFD with LACP. Only the listed PTX platforms support ESI per IFL with CFM.
Software requirements	Junos OS Evolved Release 24.2R1 and later is required for the ACX and PTX platforms.

Configure ESI per IFD with LACP

1. Configure LACP for ESI per IFD active-standby on a multihomed CE.

Configure the AE member interface towards PE1.

```
set interfaces <interface-name> ether-options 802.3ad <aggregate-ethernet-interface-name>
```

Configure the AE member interface towards PE2.

```
set interfaces <interface-name> ether-options 802.3ad <aggregate-ethernet-interface-name>
```

Configure the AE interface.

```
set interfaces <aggregate-ethernet-interface-name> flexible-vlan-tagging
set interfaces <aggregate-ethernet-interface-name> encapsulation flexible-ethernet-services
set interfaces <aggregate-ethernet-interface-name> unit <unit-number> encapsulation vlan-bridge
set interfaces <aggregate-ethernet-interface-name> unit <unit-number> vlan-id <vlan-id>
set interfaces <aggregate-ethernet-interfae-name> aggregated-ether-options lacp active
```

2. Configure LACP for ESI per IFD active-standby on the PE devices.

Configure the AE member interface on PE1 towards the CE.

```
set interfaces et-0/0/1 ether-options 802.3ad <aggregate-ethernet-interface-name>
```

Configure the AE interface on PE1.

```
set interfaces <aggregate-ethernet-interface-name> flexible-vlan-tagging
set interfaces <aggregate-ethernet-interface-name> encapsulation flexible-ethernet-services
set interfaces <aggregate-ethernet-interface-name> esi <esi-value>
set interfaces <aggregate-ethernet-interface-name> esi single-active
set interfaces <aggregate-ethernet-interface-name> esi df-election-granularity per-esi lacp-
oos-on-ndf
set interfaces <aggregate-ethernet-interface-name> esi df-election-type preference value
<value>
set interfaces <aggregate-ethernet-interface-name> unit <unit> encapsulation vlan-bridge
set interfaces <aggregate-ethernet-interface-name> unit <unit> vlan-id <vlan-id>
```

3. Configure the AE interface on PE2 identically to PE1. The only difference is the df-election-type preference value must be a different value from PE1.

```
set interfaces <aggregate-ethernet-interface-name> flexible-vlan-tagging
set interfaces <aggregate-ethernet-interface-name> encapsulation flexible-ethernet-services
set interfaces <aggregate-ethernet-interface-name> esi <esi-value>
set interfaces <aggregate-ethernet-interface-name> esi single-active
set interfaces <aggregate-ethernet-interface-name> esi df-election-granularity per-esi lacp-
oos-on-ndf
set interfaces <aggregate-ethernet-interface-name> esi df-election-type preference value
<value>
set interfaces <aggregate-ethernet-interface-name> unit <unit> encapsulation vlan-bridge
set interfaces <aggregate-ethernet-interface-name> unit <unit> vlan-id <vlan-id>
```

4. Configure a MAC-VRF instance on each provider edge (PE) device.

```
set routing-instances <vrf-name> instance-type mac-vrf
```

5. Configure the EVPN protocol.

```
set routing-instances <vrf-name> protocols evpn
```

6. Configure a service type. We support vlan-based, vlan-aware, and vlan-bundle service types. We use vlan-based in this example.

```
set routing-instances <vrf-name> service-type vlan-based
```

7. Add the AE interface from step 1.

```
set routing-instances <vrf-name> interface <interface-name>
```

8. Configure a unique route distinguisher and vrf target.

```
set routing-instances <vrf-name> route-distinguisher <route-distinguisher-value>
set routing-instances <vrf-name> vrf-target <vrf-target>
```

9. Configure all necessary VLAN's. Include the AE interface and a Layer 3 interface.

```
set routing-instances <vrf-name> vlans <bridge-domain-name> vlan-id <vlan-id>
set routing-instances <vrf-name> vlans <bridge-domain-name> interface <interface-name>
set routing-instances <vrf-name> vlans <bridge-domain-name> l3-interface <l3-interface-name>
```

Configure ESI per IFL with CFM

1. Configure CFM for ESI per IFL active-standby on a multihomed CE.

Configure an action profile.

```
set protocols oam ethernet connectivity-fault-management action-profile <ap-name> event
interface-status-tlv down
set protocols oam ethernet connectivity-fault-management action-profile <ap-name> action
interface-down
```

Configure CFM parameters towards PE1.

```
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
level <value>
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> continuity-check interval 1s
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> mep <mep-id> interface <interface-name>
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> mep <mep-id> direction down
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> mep <mep-id> auto-discovery
```



```
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> mep <mep-id> remote-mep <mep-id> action-profile <ap-name>
```

Configure CFM parameters towards PE2.

```
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
level <value>
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> continuity-check interval 1s
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> mep <mep-id> interface <interface-name>
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> mep <mep-id> direction down
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> mep <mep-id> auto-discovery
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> mep <mep-id> remote-mep <mep-id> action-profile <ap-name>
```

2. Configure CFM on PE1.

```
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
level <value>
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> continuity-check interval 1s
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> continuity-check interface-status-tlv
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> mep <mep-id> interface <interface-name>
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> mep <mep-id> direction down
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE1-md-name>
maintenance-association <ma-name> mep <mep-id> auto-discovery
```

3. Configure CFM on PE2.

```
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
level <value>
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> continuity-check interval 1s
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> continuity-check interface-status-tlv
```

```

set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> mep <mep-id> interface <interface-name>
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> mep <mep-id> direction down
set protocols oam ethernet connectivity-fault-management maintenance-domain <PE2-md-name>
maintenance-association <ma-name> mep <mep-id> auto-discovery

```

4. Configure a routing instance with the EVPN protocol on both PE devices. We use a MAC-VRF instance in this example.

```

set routing-instances <vrf-name> instance-type mac-vrf
set routing-instances <vrf-name> protocols evpn

```

5. Configure a service type. We support vlan-based, vlan-aware, and vlan-bundle service types. We use vlan-based in this example.

```

set routing-instances <vrf-name> service-type vlan-based

```

6. Add the CE-facing interface. Ensure you specify the correct CE-facing interface on each PE. These are the same interfaces specified in steps 2 and 3.

Add the CE-facing interface on PE1:

```

set routing-instances <vrf-name> interface <interface-name>

```

Add the CE-facing interface on PE2:

```

set routing-instances <vrf-name> interface <interface-name>

```

7. Configure a unique route distinguisher and vrf target.

```

set routing-instances <vrf-name> route-distinguisher <route-distinguisher-value>
set routing-instances <vrf-name> vrf-target <vrf-target>

```

8. Add all necessary VLAN's and interfaces.

```
set routing-instances <vrf-name> vlans <bridge-domain-name> vlan-id <vlan-id>
set routing-instances <vrf-name> vlans <bridge-domain-name> interface <interface-name>
set routing-instances <vrf-name> vlans <bridge-domain-name> l3-interface <l3-interface-name>
```

Verification

Use the following commands to check various elements in either the LACP or CFM based deployments.

Command	Verification Task
show evpn instance extensive	On a PE device, check the status of the DF and non-DF nodes.
show lacp interfaces	On a CE device, check the status of the DF and non-DF interfaces.
show interfaces <interface-name>	On a CE device, check the status of the DF and non-DF nodes in CFM.
show oam ethernet connectivity- fault-management interfaces detail	On a CE device, check the status of CFM.

Release Information

(PTX10001-36MR, PTX10004, PTX10008, and PTX10016)—Support for ESI per IFD with LACP and ESI per IFL with CFM added in Junos OS Evolved Release 24.2R1.

(ACX7332 and ACX7024X)—Support for ESI per IFD with LACP added in Junos OS Evolved Release 24.2R1.

(ACX7100-32C, ACX7100-48L, ACX7348, ACX7509, and ACX7024)—Support for ESI per IFD with LACP added in Junos OS Evolved Release 23.4R1.

Example: Configuring Basic EVPN Active-Active Multihoming

IN THIS SECTION

● Requirements | 363

- Overview | 363
- Configuration | 364

This example shows how to configure an active-active multihomed customer edge (CE) devices and provider edge (PE) devices in an Ethernet VPN (EVPN).

Requirements

This example uses the following hardware and software components:

- Four MX Series 5G Universal Routing Platforms with MPC interfaces only, where:
 - Two devices are configured as provider edge (PE) routers connected to a common multihomed customer site.
 - One device is configured as a remote PE router connected to a single-homed customer site.
 - One device is configured as a provider router.
- Two customer edge (CE) devices, where:
 - Two CE devices that are multihomed.
 - One CE devices that is single-homed .

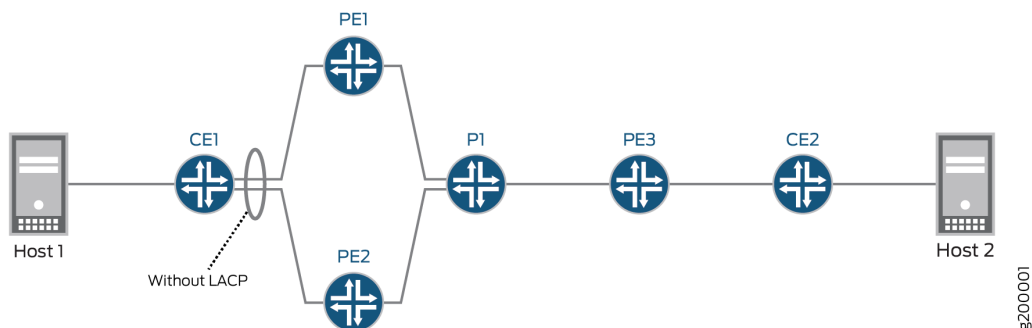
Before you begin:

1. Configure the router interfaces.
2. Configure OSPF or any other IGP protocol.
3. Configure a BGP internal group.
4. Configure MPLS.

Overview

[Figure 28 on page 364](#) illustrates a simple EVPN topology. Routers PE1 and PE2 are provider edge (PE) routers connected to multihomed customer edge (CE) router CE1. Router P is the provider router connected to Routers PE1, PE2, and PE3. Router PE3 is connected to customer edge router CE2.

Figure 28: Simple EVPN Multihomed Topology



Configuration

IN THIS SECTION

- [CLI Quick Configuration | 364](#)

CLI Quick Configuration

The configurations parameters for the routers are as follows:

CE1

```
chassis {
  aggregated-devices {
    ethernet {
      device-count 1;
    }
  }
}
interfaces {
  ge-0/0/1 {
    description To-PE1;
    gigether-options {
      802.3ad ae0;
    }
  }
  ge-0/0/2 {
```

```

        description To-PE2;
        gigether-options {
            802.3ad ae0;
        }
    }
    ge-0/0/7 {
        unit 0 {
            family bridge {
                interface-mode access;
                vlan-id 10;
            }
        }
    }
    ge-0/0/8 {
        unit 0 {
            family bridge {
                interface-mode access;
                vlan-id 20;
            }
        }
    }
    ae0 {
        description To-PE1_and_PE2;
        unit 0 {
            family bridge {
                interface-mode trunk;
                vlan-id-list [ 10 20 ];
            }
        }
    }
}
bridge-domains {
    BD {
        vlan-id-list [ 10 20 ];
    }
}

```

PE1

```

interfaces {
    ge-0/0/1 {
        description To-CE1;
    }
}

```

```

flexible-vlan-tagging;
encapsulation flexible-ethernet-services;
esi {
    00:11:11:11:11:11:11:11:11:11;
    all-active;
}
unit 10 {
    family bridge {
        interface-mode trunk;
        vlan-id-list [ 10 20 ];
    }
}
}
ge-0/0/3 {
    unit 0 {
        family inet {
            address 10.3.3.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.1.1/32;
        }
    }
}
}
routing-options {
    router-id 192.168.1.1;
    autonomous-system 65432;
    forwarding-table {
        export evpn-pplb;
    }
}
}
protocols {
    rsvp {
        interface ge-0/0/3.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE1-to-PE2 {

```

```

        to 192.168.2.2;
    }
    label-switched-path PE1-to-PE3 {
        to 192.168.3.3;
    }
    interface ge-0/0/3.0;
}
bgp {
    group EVPN-PE {
        type internal;
        local-address 192.168.1.1;
        family inet-vpn {
            unicast;
        }
        family evpn {
            signaling;
        }
        neighbor 192.168.3.3;
        neighbor 192.168.2.2;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all {
            interface-type p2p;
        }
        interface fxp0.0 {
            disable;
        }
    }
}
}
routing-instances {
    evpn-ta {
        instance-type virtual-switch;
        interface ge-0/0/1.10;
        route-distinguisher 65432:10;
        vrf-target target:65432:10;
        protocols {
            evpn {
                extended-vlan-list [ 10 20 ];
            }
        }
    }
}

```



```

    }
    bridge-domains {
        bd10 {
            domain-type bridge;
            vlan-id 10;
        }
        bd20 {
            domain-type bridge;
            vlan-id 20;
        }
    }
}
}
policy-statement evpn-pplb {
    from protocol evpn;
    then {
        load-balance per-packet;
    }
}
}

```

PE2

```

interfaces {
    ge-0/0/2 {
        description To-CE1;
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        esi {
            00:11:11:11:11:11:11:11:11:11;
            all-active;
        }
        unit 10 {
            family bridge {
                interface-mode trunk;
                vlan-id-list [ 10 20 ];
            }
        }
    }
    ge-0/0/4 {
        unit 0 {
            family inet {
                address 10.4.4.1/30;
            }
        }
    }
}

```

```

        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.2.2/32;
        }
    }
}
}
routing-options {
    router-id 192.168.2.2;
    autonomous-system 65432;
    forwarding-table {
        export evpn-pplb;
    }
}
protocols {
    rsvp {
        interface ge-0/0/4.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE2-to-PE1 {
            to 192.168.1.1;
        }
        label-switched-path PE2-to-PE3 {
            to 192.168.3.3;
        }
        interface ge-0/0/4.0;
    }
    bgp {
        group EVPN-PE {
            type internal;
            local-address 192.168.2.2;
            family inet-vpn {
                unicast;
            }
            family evpn {
                signaling;
            }
        }
    }
}

```

```

        neighbor 192.168.1.1;
        neighbor 192.168.3.3;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all {
            interface-type p2p;
        }
        interface fxp0.0 {
            disable;
        }
    }
}
}
routing-instances {
    evpn-ta {
        instance-type virtual-switch;
        interface ge-0/0/2.10;
        route-distinguisher 65432:10;
        vrf-target target:65432:10;
        protocols {
            evpn {
                extended-vlan-list [ 10 20 ];
            }
        }
        bridge-domains {
            bd10 {
                domain-type bridge;
                vlan-id 10;
            }
            bd20 {
                domain-type bridge;
                vlan-id 20;
            }
        }
    }
}
}
policy-statement evpn-pplb {
    from protocol evpn;
    then {
        load-balance per-packet;
    }
}

```

```

    }
}

```

Router PE3

```

interfaces {
    ge-0/0/5 {
        unit 0 {
            family inet {
                address 10.5.5.1/30;
            }
            family mpls;
        }
    }
    ge-0/0/6 {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 10 {
            family bridge {
                interface-mode trunk;
                vlan-id-list 10;
            }
        }
        unit 20 {
            family bridge {
                interface-mode trunk;
                vlan-id-list 20;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.168.3.3/32;
            }
        }
    }
    routing-options {
        router-id 192.168.3.3;
        autonomous-system 65432;
        forwarding-table {
            export evpn-pplb;
        }
    }
}

```

```

    }
}
protocols {
    rsvp {
        interface ge-0/0/5.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE3-to-PE1 {
            to 192.168.1.1;
        }
        label-switched-path PE3-to-PE2 {
            to 192.168.2.2;
        }
        interface ge-0/0/5.0;
    }
    bgp {
        group EVPN-PE {
            type internal;
            local-address 192.168.3.3;
            family inet-vpn {
                unicast;
            }
            family evpn {
                signaling;
            }
            neighbor 192.168.1.1;
            neighbor 192.168.2.2;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all {
                interface-type p2p;
            }
            interface fxp0.0 {
                disable;
            }
        }
    }
}
routing-instances {

```

```

evpn-ta {
    instance-type virtual-switch;
    interface ge-0/0/6.10;
    interface ge-0/0/6.20;
    route-distinguisher 65432:10;
    vrf-target target:65432:10;
    protocols {
        evpn {
            extended-vlan-list [ 10 20 ];
        }
    }
    bridge-domains {
        bd10 {
            domain-type bridge;
            vlan-id 10;
            bridge-options {
                interface ge-0/0/6.10;
            }
        }
        bd20 {
            domain-type bridge;
            vlan-id 20;
            bridge-options {
                interface ge-0/0/6.20;
            }
        }
    }
}

policy-statement evpn-pplb {
    from protocol evpn;
    then {
        load-balance per-packet;
    }
}

```

Router P

```

interfaces {
    ge-0/0/3 {
        unit 0 {

```

```

        family inet {
            address 10.3.3.2/30;
        }
        family mpls;
    }
}
ge-0/0/4 {
    unit 0 {
        family inet {
            address 10.4.4.2/30;
        }
        family mpls;
    }
}
ge-0/0/5 {
    unit 0 {
        family inet {
            address 10.5.5.2/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.4.4/32;
        }
    }
}
}
routing-options {
    router-id 192.168.4.4;
    autonomous-system 65432;
}
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
mpls {
    interface all;

```

```

        interface fxp0.0 {
            disable;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all {
                interface-type p2p;
            }
            interface fxp0.0 {
                disable;
            }
        }
    }
}

```

Example: Configuring EVPN Active-Active Multihoming

IN THIS SECTION

- [Requirements | 375](#)
- [Overview and Topology | 376](#)
- [Configuration | 379](#)
- [Verification | 418](#)

This example shows how to configure Ethernet VPN (EVPN) for multihomed customer edge devices in the active-active redundancy mode, so the Layer 2 unicast traffic can be load-balanced across all the multihomed links on and toward the CE device.

Requirements

This example uses the following hardware and software components:

- Five MX Series 5G Universal Routing Platforms with MPC interfaces only, where:

- Three devices are configured as provider edge (PE) routers connected to a common multihomed customer site.
- One device is configured as a remote PE router connected to a single-homed customer site.
- Six customer edge (CE) devices, with one multihomed CE device and the rest of the CE devices being single-homed to each of the PE routers.
- Junos OS Release 16.1 or later running on all the PE routers.



NOTE: See [Feature Explorer](#) for platform and release support for any Junos OS feature, including EVPN multihoming active-active mode.

Before you begin:

1. Configure the router interfaces.
2. Configure IS-IS or any other IGP protocol.
3. Configure BGP.
4. Configure LDP.
5. Configure MPLS.
6. Configure RSVP MPLS LSP or GRE tunnels.

Overview and Topology

This EVPN solution on MX Series routers with MPC interfaces is extended to provide multihoming functionality in the active-active redundancy mode of operation. This feature enables load balancing of Layer 2 unicast traffic across all the multihomed links on and toward a customer edge device.

The EVPN active-active multihoming feature provides link-level and node-level redundancy along with effective utilization of resources.

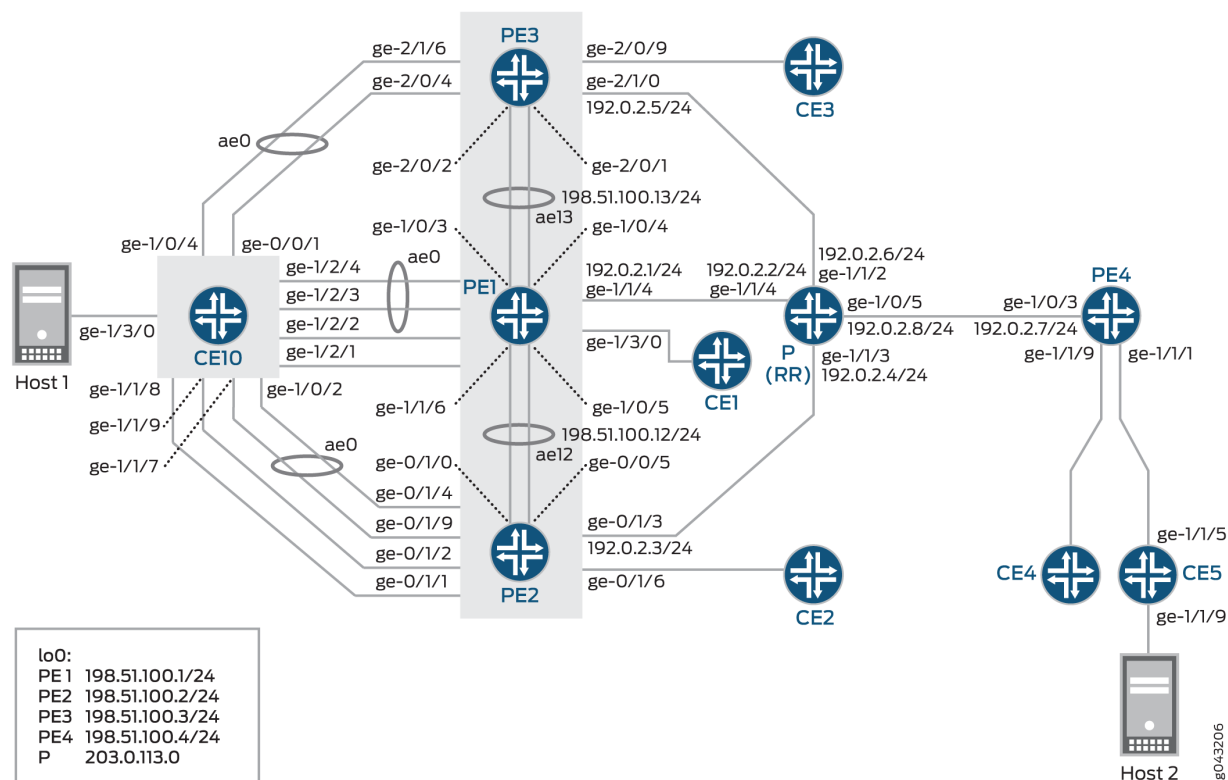
To enable EVPN active-active multihoming, include the `all-active` statement at the `[edit interfaces esi]` hierarchy level.

In [Figure 29 on page 377](#), the core consists of four provider edge (PE) routers and a provider router (P) that is configured as a route reflector (RR). Router CE10 is multihomed to Routers PE1, PE2, and PE3. Each PE router is also connected to a single-homed customer site.

There are three routing instances running in the topology – VS-1, VS-2, and mhevpn, along with the default routing instance. The routing instances share nine VLANs with three ESIs each. The VS-1 and VS-2 routing instances are configured as a virtual switch type of routing instance, and the mhevpn routing instance is an EVPN routing instance.

Three aggregated bundles – ae0, ae1, and ae2 – are used to connect the multihomed CE device, CE10, to Routers PE1, PE2, and PE3. These aggregated bundles are configured for three ESIs each. The aggregated bundles, ae12 and ae13, are used to interconnect Routers PE1 and PE2, and PE1 and PE3, respectively.

Figure 29: EVPN Active-Active Multihoming Topology



For Active-Active redundancy, additional configurations are required in the “interconnect” stanza to enable DCI interconnection. For a default switch (switch-options) configuration, be sure to set the DCI under global protocols evpn: [edit global protocols evpn].

Protocols EVPN Example:

```
evpn-vxlan-dc1
  vtep-source-interface lo0.0;
  instance-type mac-vrf;
  route-distinguisher 101:1;
  vrf-target target:1:1;
  protocols {
    evpn {
```

```

        encapsulation vxlan;
        extended-vni-list all;
    interconnect {
        vrf-target target:2:2;
        vrf-import
        route-distinguisher 101:2;
        interconnected-vni-list all;
        esi {
            00:00:01:02:03:04:05:06:07:08;
            all-active;
        }
    }
}

vlangs {
    bd1 {
        vlan-id 51;
        l3-interface irb.0;
        vxlan {
            vni 51;
            translation-vni
        }
    }
}

global
protocols {
    evpn {
        interconnect-multihoming-peer-gateways
    }
}

```



NOTE: Configure this statement `interconnect-multihoming-peer-gateways` to contain a list of all DCI peers on the same DC.

The list can contain up to 64 peer gateway entries. Be sure to configure under the `global protocol evpn` stanza [edit `global protocols evpn`] and not under any `mac-vrf` setting [edit `protocols evpn-vxlan-dc1 instance-type mac-vrf`].

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 379](#)
- [Procedure | 397](#)
- [Configuration on EX9200 Switches | 408](#)
- [Results | 409](#)

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.

CE10

```
set chassis aggregated-devices ethernet device-count 20
set chassis network-services enhanced-ip
set interfaces ge-0/0/1 gigether-options 802.3ad ae0
set interfaces ge-1/0/2 gigether-options 802.3ad ae0
set interfaces ge-1/0/4 gigether-options 802.3ad ae0
set interfaces ge-1/1/7 gigether-options 802.3ad ae0
set interfaces ge-1/1/8 gigether-options 802.3ad ae2
set interfaces ge-1/1/9 gigether-options 802.3ad ae1
set interfaces ge-1/2/1 gigether-options 802.3ad ae2
set interfaces ge-1/2/2 gigether-options 802.3ad ae1
set interfaces ge-1/2/3 gigether-options 802.3ad ae0
set interfaces ge-1/2/4 gigether-options 802.3ad ae0
set interfaces ge-1/3/0 flexible-vlan-tagging
set interfaces ge-1/3/0 encapsulation flexible-ethernet-services
set interfaces ge-1/3/0 unit 0 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 0 vlan-id 10
set interfaces ge-1/3/0 unit 1 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 1 vlan-id 20
set interfaces ge-1/3/0 unit 2 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 2 vlan-id 30
set interfaces ge-1/3/0 unit 110 encapsulation vlan-bridge
```

```
set interfaces ge-1/3/0 unit 110 vlan-id 110
set interfaces ge-1/3/0 unit 120 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 120 vlan-id 120
set interfaces ge-1/3/0 unit 130 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 130 vlan-id 130
set interfaces ge-1/3/0 unit 210 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 210 vlan-id 210
set interfaces ge-1/3/0 unit 220 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 220 vlan-id 220
set interfaces ge-1/3/0 unit 230 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 230 vlan-id 230
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set interfaces ae0 unit 110 encapsulation vlan-bridge
set interfaces ae0 unit 110 vlan-id 110
set interfaces ae0 unit 210 encapsulation vlan-bridge
set interfaces ae0 unit 210 vlan-id 210
set interfaces ae1 flexible-vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 unit 1 encapsulation vlan-bridge
set interfaces ae1 unit 1 vlan-id 20
set interfaces ae1 unit 120 encapsulation vlan-bridge
set interfaces ae1 unit 120 vlan-id 120
set interfaces ae1 unit 220 encapsulation vlan-bridge
set interfaces ae1 unit 220 vlan-id 220
set interfaces ae2 flexible-vlan-tagging
set interfaces ae2 encapsulation flexible-ethernet-services
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 unit 2 encapsulation vlan-bridge
set interfaces ae2 unit 2 vlan-id 30
set interfaces ae2 unit 130 encapsulation vlan-bridge
set interfaces ae2 unit 130 vlan-id 130
set interfaces ae2 unit 230 encapsulation vlan-bridge
set interfaces ae2 unit 230 vlan-id 230
set routing-options forwarding-table export load-balancing-policy
set protocols l2-learning global-mac-table-aging-time 18000
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set bridge-domains bd10 domain-type bridge
set bridge-domains bd10 vlan-id 10
```

```

set bridge-domains bd10 interface ae0.0
set bridge-domains bd10 interface ge-1/3/0.0
set bridge-domains bd110 domain-type bridge
set bridge-domains bd110 vlan-id 110
set bridge-domains bd110 interface ae0.110
set bridge-domains bd110 interface ge-1/3/0.110
set bridge-domains bd120 domain-type bridge
set bridge-domains bd120 vlan-id 120
set bridge-domains bd120 interface ge-1/3/0.120
set bridge-domains bd120 interface ae1.120
set bridge-domains bd130 domain-type bridge
set bridge-domains bd130 vlan-id 130
set bridge-domains bd130 interface ge-1/3/0.130
set bridge-domains bd130 interface ae2.130
set bridge-domains bd20 domain-type bridge
set bridge-domains bd20 vlan-id 20
set bridge-domains bd20 interface ge-1/3/0.1
set bridge-domains bd20 interface ae1.1
set bridge-domains bd210 domain-type bridge
set bridge-domains bd210 vlan-id 210
set bridge-domains bd210 interface ae0.210
set bridge-domains bd210 interface ge-1/3/0.210
set bridge-domains bd220 domain-type bridge
set bridge-domains bd220 vlan-id 220
set bridge-domains bd220 interface ge-1/3/0.220
set bridge-domains bd220 interface ae1.220
set bridge-domains bd230 domain-type bridge
set bridge-domains bd230 vlan-id 230
set bridge-domains bd230 interface ge-1/3/0.230
set bridge-domains bd230 interface ae2.230
set bridge-domains bd30 domain-type bridge
set bridge-domains bd30 vlan-id 30
set bridge-domains bd30 interface ge-1/3/0.2
set bridge-domains bd30 interface ae2.2

```

CE5

```

set chassis aggregated-devices ethernet device-count 20
set chassis network-services enhanced-ip
set interfaces ge-1/1/5 gigether-options 802.3ad ae0
set interfaces ge-1/1/9 flexible-vlan-tagging
set interfaces ge-1/1/9 encapsulation flexible-ethernet-services

```

```

set interfaces ge-1/1/9 unit 0 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 0 vlan-id 10
set interfaces ge-1/1/9 unit 1 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 1 vlan-id 20
set interfaces ge-1/1/9 unit 2 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 2 vlan-id 30
set interfaces ge-1/1/9 unit 110 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 110 vlan-id 110
set interfaces ge-1/1/9 unit 120 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 120 vlan-id 120
set interfaces ge-1/1/9 unit 130 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 130 vlan-id 130
set interfaces ge-1/1/9 unit 210 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 210 vlan-id 210
set interfaces ge-1/1/9 unit 220 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 220 vlan-id 220
set interfaces ge-1/1/9 unit 230 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 230 vlan-id 230
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set interfaces ae0 unit 1 encapsulation vlan-bridge
set interfaces ae0 unit 1 vlan-id 20
set interfaces ae0 unit 2 encapsulation vlan-bridge
set interfaces ae0 unit 2 vlan-id 30
set interfaces ae0 unit 110 encapsulation vlan-bridge
set interfaces ae0 unit 110 vlan-id 110
set interfaces ae0 unit 120 encapsulation vlan-bridge
set interfaces ae0 unit 120 vlan-id 120
set interfaces ae0 unit 130 encapsulation vlan-bridge
set interfaces ae0 unit 130 vlan-id 130
set interfaces ae0 unit 210 encapsulation vlan-bridge
set interfaces ae0 unit 210 vlan-id 210
set interfaces ae0 unit 220 encapsulation vlan-bridge
set interfaces ae0 unit 220 vlan-id 220
set interfaces ae0 unit 230 encapsulation vlan-bridge
set interfaces ae0 unit 230 vlan-id 230
set interfaces lo0 unit 6 family inet address 203.0.113.0/24
set interfaces lo0 unit 6 family iso address 47.0005.80ff.f800.0000.0108.0000.6006.0070.0600
set routing-options forwarding-table export load-balancing-policy
set protocols l2-learning global-mac-table-aging-time 18000
set policy-options policy-statement load-balancing-policy then load-balance per-packet

```

```

set bridge-domains bd10 domain-type bridge
set bridge-domains bd10 vlan-id 10
set bridge-domains bd10 interface ae0.0
set bridge-domains bd10 interface ge-1/1/9.0
set bridge-domains bd110 domain-type bridge
set bridge-domains bd110 vlan-id 110
set bridge-domains bd110 interface ae0.110
set bridge-domains bd110 interface ge-1/1/9.110
set bridge-domains bd120 domain-type bridge
set bridge-domains bd120 vlan-id 120
set bridge-domains bd120 interface ge-1/1/9.120
set bridge-domains bd120 interface ae0.120
set bridge-domains bd130 domain-type bridge
set bridge-domains bd130 vlan-id 130
set bridge-domains bd130 interface ge-1/1/9.130
set bridge-domains bd130 interface ae0.130
set bridge-domains bd20 domain-type bridge
set bridge-domains bd20 vlan-id 20
set bridge-domains bd20 interface ae0.1
set bridge-domains bd20 interface ge-1/1/9.1
set bridge-domains bd210 domain-type bridge
set bridge-domains bd210 vlan-id 210
set bridge-domains bd210 interface ae0.210
set bridge-domains bd210 interface ge-1/1/9.210
set bridge-domains bd220 domain-type bridge
set bridge-domains bd220 vlan-id 220
set bridge-domains bd220 interface ge-1/1/9.220
set bridge-domains bd220 interface ae0.220
set bridge-domains bd230 domain-type bridge
set bridge-domains bd230 vlan-id 230
set bridge-domains bd230 interface ge-1/1/9.230
set bridge-domains bd230 interface ae0.230
set bridge-domains bd30 domain-type bridge
set bridge-domains bd30 vlan-id 30
set bridge-domains bd30 interface ge-1/1/9.2
set bridge-domains bd30 interface ae0.2

```

PE1

```

set chassis aggregated-devices ethernet device-count 20
set chassis network-services enhanced-ip
set interfaces ge-1/0/3 gigether-options 802.3ad ae13

```



```

set interfaces ge-1/0/4 gigether-options 802.3ad ae13
set interfaces ge-1/0/5 gigether-options 802.3ad ae12
set interfaces ge-1/1/4 unit 0 family inet address 192.0.2.1/24
set interfaces ge-1/1/4 unit 0 family iso
set interfaces ge-1/1/4 unit 0 family mpls
set interfaces ge-1/1/6 gigether-options 802.3ad ae12
set interfaces ge-1/2/1 flexible-vlan-tagging
set interfaces ge-1/2/1 encapsulation flexible-ethernet-services
set interfaces ge-1/2/1 esi 00:33:33:33:33:33:33:33:33:33
set interfaces ge-1/2/1 esi all-active
set interfaces ge-1/2/1 unit 0 encapsulation vlan-bridge
set interfaces ge-1/2/1 unit 0 vlan-id 30
set interfaces ge-1/2/1 unit 130 family bridge interface-mode trunk
set interfaces ge-1/2/1 unit 130 family bridge vlan-id-list 130
set interfaces ge-1/2/1 unit 230 family bridge interface-mode trunk
set interfaces ge-1/2/1 unit 230 family bridge vlan-id-list 230
set interfaces ge-1/2/2 flexible-vlan-tagging
set interfaces ge-1/2/2 encapsulation flexible-ethernet-services
set interfaces ge-1/2/2 esi 00:22:22:22:22:22:22:22:22:22
set interfaces ge-1/2/2 esi all-active
set interfaces ge-1/2/2 unit 0 encapsulation vlan-bridge
set interfaces ge-1/2/2 unit 0 vlan-id 20
set interfaces ge-1/2/2 unit 120 family bridge interface-mode trunk
set interfaces ge-1/2/2 unit 120 family bridge vlan-id-list 120
set interfaces ge-1/2/2 unit 220 family bridge interface-mode trunk
set interfaces ge-1/2/2 unit 220 family bridge vlan-id-list 220
set interfaces ge-1/2/3 gigether-options 802.3ad ae0
set interfaces ge-1/2/4 gigether-options 802.3ad ae0
set interfaces ge-1/3/0 flexible-vlan-tagging
set interfaces ge-1/3/0 encapsulation flexible-ethernet-services
set interfaces ge-1/3/0 unit 0 encapsulation vlan-bridge
set interfaces ge-1/3/0 unit 0 vlan-id 10
set interfaces ge-1/3/0 unit 100 family bridge interface-mode trunk
set interfaces ge-1/3/0 unit 100 family bridge vlan-id-list 110
set interfaces ge-1/3/0 unit 100 family bridge vlan-id-list 120
set interfaces ge-1/3/0 unit 100 family bridge vlan-id-list 130
set interfaces ge-1/3/0 unit 200 family bridge interface-mode trunk
set interfaces ge-1/3/0 unit 200 family bridge vlan-id-list 210
set interfaces ge-1/3/0 unit 200 family bridge vlan-id-list 220
set interfaces ge-1/3/0 unit 200 family bridge vlan-id-list 230
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11

```

```

set interfaces ae0 esi all-active
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set interfaces ae0 unit 110 family bridge interface-mode trunk
set interfaces ae0 unit 110 family bridge vlan-id-list 110
set interfaces ae0 unit 210 family bridge interface-mode trunk
set interfaces ae0 unit 210 family bridge vlan-id-list 210
set interfaces ae12 flexible-vlan-tagging
set interfaces ae12 encapsulation flexible-ethernet-services
set interfaces ae12 aggregated-ether-options minimum-links 1
set interfaces ae12 unit 0 vlan-id 1200
set interfaces ae12 unit 0 family inet address 198.51.100.12/24
set interfaces ae12 unit 0 family iso
set interfaces ae12 unit 0 family mpls
set interfaces ae13 flexible-vlan-tagging
set interfaces ae13 encapsulation flexible-ethernet-services
set interfaces ae13 aggregated-ether-options minimum-links 1
set interfaces ae13 unit 0 vlan-tags outer 1300
set interfaces ae13 unit 0 vlan-tags inner 13
set interfaces ae13 unit 0 family inet address 198.51.100.13/24
set interfaces ae13 unit 0 family iso
set interfaces ae13 unit 0 family mpls
set interfaces irb unit 0 family inet address 192.0.2.9/24
set interfaces irb unit 0 mac 00:99:99:99:01:99
set interfaces irb unit 1 family inet address 192.0.2.10/24
set interfaces irb unit 1 mac 00:99:99:99:02:99
set interfaces irb unit 2 family inet address 192.0.2.11/24
set interfaces irb unit 2 mac 00:99:99:99:03:99
set interfaces irb unit 10 family inet address 192.0.2.12/24
set interfaces irb unit 10 mac 00:99:99:99:01:90
set interfaces lo0 unit 0 family inet address 198.51.100.1/24 primary
set interfaces lo0 unit 0 family iso
set routing-options router-id 198.51.100.1
set routing-options autonomous-system 65221
set routing-options forwarding-table export load-balancing-policy
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path pe1tope2 from 198.51.100.1
set protocols mpls label-switched-path pe1tope2 to 198.51.100.2
set protocols mpls label-switched-path pe1tope2 primary direct_to_pe2
set protocols mpls label-switched-path pe1tope3 from 198.51.100.1
set protocols mpls label-switched-path pe1tope3 to 198.51.100.3
set protocols mpls label-switched-path pe1tope3 primary direct_to_pe3

```

```

set protocols mpls label-switched-path pe1tope4 from 198.51.100.1
set protocols mpls label-switched-path pe1tope4 to 198.51.100.4
set protocols mpls label-switched-path pe1tope4 primary direct_to_pe4
set protocols mpls path pe4_to_pe3 198.51.100.4 strict
set protocols mpls path pe4_to_pe3 198.51.100.3 strict
set protocols mpls path direct_to_pe2 198.51.100.5 strict
set protocols mpls path direct_to_pe3 198.51.100.6 strict
set protocols mpls path direct_to_pe4 198.51.100.9 strict
set protocols mpls path pe2_to_pe3 198.51.100.2 strict
set protocols mpls path pe2_to_pe3 198.51.100.3 strict
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group RR type internal
set protocols bgp group RR local-address 198.51.100.1
set protocols bgp group RR family evpn signaling
set protocols bgp group RR neighbor 203.0.113.0
set protocols isis level 1 disable
set protocols isis interface all level 2 metric 10
set protocols isis interface fxp0.0 disable
set protocols isis interface lo0.0 level 2 metric 0
set protocols ldp deaggregate
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols evpn
set protocols l2-learning global-mac-table-aging-time 18000
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set routing-instances VS-1 instance-type virtual-switch
set routing-instances VS-1 interface ge-1/2/1.130
set routing-instances VS-1 interface ge-1/2/2.120
set routing-instances VS-1 interface ge-1/3/0.100
set routing-instances VS-1 interface ae0.110
set routing-instances VS-1 route-distinguisher 198.51.100.1:101
set routing-instances VS-1 vrf-target target:100:101
set routing-instances VS-1 protocols evpn extended-vlan-list 110
set routing-instances VS-1 protocols evpn extended-vlan-list 120
set routing-instances VS-1 protocols evpn extended-vlan-list 130
set routing-instances VS-1 protocols evpn default-gateway do-not-advertise
set routing-instances VS-1 bridge-domains bd-110 vlan-id 110
set routing-instances VS-1 bridge-domains bd-110 routing-interface irb.0
set routing-instances VS-1 bridge-domains bd-120 vlan-id 120
set routing-instances VS-1 bridge-domains bd-120 routing-interface irb.1
set routing-instances VS-1 bridge-domains bd-130 vlan-id 130
set routing-instances VS-1 bridge-domains bd-130 routing-interface irb.2

```

```

set routing-instances VS-2 instance-type virtual-switch
set routing-instances VS-2 interface ge-1/2/1.230
set routing-instances VS-2 interface ge-1/2/2.220
set routing-instances VS-2 interface ge-1/3/0.200
set routing-instances VS-2 interface ae0.210
set routing-instances VS-2 route-distinguisher 198.51.100.1:201
set routing-instances VS-2 vrf-target target:100:201
set routing-instances VS-2 protocols evpn extended-vlan-list 210
set routing-instances VS-2 protocols evpn extended-vlan-list 220
set routing-instances VS-2 protocols evpn extended-vlan-list 230
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 210
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 220
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 230
set routing-instances mhevpn instance-type evpn
set routing-instances mhevpn vlan-id 10
set routing-instances mhevpn interface ge-1/2/1.0
set routing-instances mhevpn interface ge-1/2/2.0
set routing-instances mhevpn interface ge-1/3/0.0
set routing-instances mhevpn interface ae0.0
set routing-instances mhevpn routing-interface irb.10
set routing-instances mhevpn route-distinguisher 198.51.100.1:1
set routing-instances mhevpn vrf-target target:100:1
set routing-instances mhevpn protocols evpn default-gateway do-not-advertise
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf interface irb.1
set routing-instances vrf interface irb.2
set routing-instances vrf interface irb.10
set routing-instances vrf route-distinguisher 198.51.100.1:11
set routing-instances vrf vrf-target target:100:11
set routing-instances vrf vrf-table-label

```

PE2

```

set chassis aggregated-devices ethernet device-count 20
set chassis network-services enhanced-ip
set interfaces ge-0/0/5 gigether-options 802.3ad ae12
set interfaces ge-0/1/0 gigether-options 802.3ad ae12
set interfaces ge-0/1/1 flexible-vlan-tagging
set interfaces ge-0/1/1 encapsulation flexible-ethernet-services
set interfaces ge-0/1/1 esi 00:33:33:33:33:33:33:33:33:33
set interfaces ge-0/1/1 esi all-active

```

```

set interfaces ge-0/1/1 unit 0 encapsulation vlan-bridge
set interfaces ge-0/1/1 unit 0 vlan-id 30
set interfaces ge-0/1/1 unit 130 family bridge interface-mode trunk
set interfaces ge-0/1/1 unit 130 family bridge vlan-id-list 130
set interfaces ge-0/1/1 unit 230 family bridge interface-mode trunk
set interfaces ge-0/1/1 unit 230 family bridge vlan-id-list 230
set interfaces ge-0/1/2 flexible-vlan-tagging
set interfaces ge-0/1/2 encapsulation flexible-ethernet-services
set interfaces ge-0/1/2 esi 00:22:22:22:22:22:22:22:22:22
set interfaces ge-0/1/2 esi all-active
set interfaces ge-0/1/2 unit 0 encapsulation vlan-bridge
set interfaces ge-0/1/2 unit 0 vlan-id 20
set interfaces ge-0/1/2 unit 120 family bridge interface-mode trunk
set interfaces ge-0/1/2 unit 120 family bridge vlan-id-list 120
set interfaces ge-0/1/2 unit 220 family bridge interface-mode trunk
set interfaces ge-0/1/2 unit 220 family bridge vlan-id-list 220
set interfaces ge-0/1/3 unit 0 family inet address 192.0.2.3/24
set interfaces ge-0/1/3 unit 0 family iso
set interfaces ge-0/1/3 unit 0 family mpls
set interfaces ge-0/1/4 gigether-options 802.3ad ae0
set interfaces ge-0/1/6 flexible-vlan-tagging
set interfaces ge-0/1/6 encapsulation flexible-ethernet-services
set interfaces ge-0/1/6 unit 0 encapsulation vlan-bridge
set interfaces ge-0/1/6 unit 0 vlan-id 10
set interfaces ge-0/1/6 unit 100 family bridge interface-mode trunk
set interfaces ge-0/1/6 unit 100 family bridge vlan-id-list 110
set interfaces ge-0/1/6 unit 100 family bridge vlan-id-list 120
set interfaces ge-0/1/6 unit 100 family bridge vlan-id-list 130
set interfaces ge-0/1/6 unit 200 family bridge interface-mode trunk
set interfaces ge-0/1/6 unit 200 family bridge vlan-id-list 210
set interfaces ge-0/1/6 unit 200 family bridge vlan-id-list 220
set interfaces ge-0/1/6 unit 200 family bridge vlan-id-list 230
set interfaces ge-0/1/9 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set interfaces ae0 unit 110 family bridge interface-mode trunk
set interfaces ae0 unit 110 family bridge vlan-id-list 110
set interfaces ae0 unit 210 family bridge interface-mode trunk
set interfaces ae0 unit 210 family bridge vlan-id-list 210

```

```

set interfaces ae12 flexible-vlan-tagging
set interfaces ae12 encapsulation flexible-ethernet-services
set interfaces ae12 aggregated-ether-options minimum-links 1
set interfaces ae12 unit 0 vlan-id 1200
set interfaces ae12 unit 0 family inet address 198.51.100.5/24
set interfaces ae12 unit 0 family iso
set interfaces ae12 unit 0 family mpls
set interfaces irb unit 0 family inet address 192.0.2.9/24
set interfaces irb unit 0 mac 00:99:99:99:01:99
set interfaces irb unit 1 family inet address 192.0.2.10/24
set interfaces irb unit 1 mac 00:99:99:99:02:99
set interfaces irb unit 2 family inet address 192.0.2.11/24
set interfaces irb unit 2 mac 00:99:99:99:03:99
set interfaces irb unit 10 family inet address 192.0.2.12/24
set interfaces irb unit 10 mac 00:99:99:99:01:90
set interfaces lo0 unit 0 family inet address 198.51.100.2/32 primary
set interfaces lo0 unit 0 family iso
set routing-options router-id 198.51.100.2
set routing-options autonomous-system 65221
set routing-options forwarding-table export load-balancing-policy
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path pe2tope1 from 198.51.100.2
set protocols mpls label-switched-path pe2tope1 to 198.51.100.1
set protocols mpls label-switched-path pe2tope1 primary direct_to_pe1
set protocols mpls label-switched-path pe2tope3 from 198.51.100.2
set protocols mpls label-switched-path pe2tope3 to 198.51.100.3
set protocols mpls label-switched-path pe2tope3 primary direct_to_pe3
set protocols mpls label-switched-path pe2tope4 from 198.51.100.2
set protocols mpls label-switched-path pe2tope4 to 198.51.100.4
set protocols mpls label-switched-path pe2tope4 primary direct_to_pe4
set protocols mpls path direct_to_pe1 198.51.100.12 strict
set protocols mpls path direct_to_pe3 198.51.100.7 strict
set protocols mpls path direct_to_pe4 198.51.100.8 strict
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group RR type internal
set protocols bgp group RR local-address 198.51.100.2
set protocols bgp group RR family evpn signaling
set protocols bgp group RR neighbor 203.0.113.0
set protocols isis level 1 disable
set protocols isis interface all level 2 metric 10
set protocols isis interface fxp0.0 disable

```

```

set protocols isis interface lo0.0 level 2 metric 0
set protocols ldp deaggregate
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols l2-learning global-mac-table-aging-time 18000
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set routing-instances VS-1 instance-type virtual-switch
set routing-instances VS-1 interface ge-0/1/1.130
set routing-instances VS-1 interface ge-0/1/2.120
set routing-instances VS-1 interface ge-0/1/6.100
set routing-instances VS-1 interface ae0.110
set routing-instances VS-1 route-distinguisher 198.51.100.2:101
set routing-instances VS-1 vrf-target target:100:101
set routing-instances VS-1 protocols evpn extended-vlan-list 110
set routing-instances VS-1 protocols evpn extended-vlan-list 120
set routing-instances VS-1 protocols evpn extended-vlan-list 130
set routing-instances VS-1 protocols evpn default-gateway do-not-advertise
set routing-instances VS-1 bridge-domains bd-110 vlan-id 110
set routing-instances VS-1 bridge-domains bd-110 routing-interface irb.0
set routing-instances VS-1 bridge-domains bd-120 vlan-id 120
set routing-instances VS-1 bridge-domains bd-120 routing-interface irb.1
set routing-instances VS-1 bridge-domains bd-130 vlan-id 130
set routing-instances VS-1 bridge-domains bd-130 routing-interface irb.2
set routing-instances VS-2 instance-type virtual-switch
set routing-instances VS-2 interface ge-0/1/1.230
set routing-instances VS-2 interface ge-0/1/2.220
set routing-instances VS-2 interface ge-0/1/6.200
set routing-instances VS-2 interface ae0.210
set routing-instances VS-2 route-distinguisher 198.51.100.2:201
set routing-instances VS-2 vrf-target target:100:201
set routing-instances VS-2 protocols evpn extended-vlan-list 210
set routing-instances VS-2 protocols evpn extended-vlan-list 220
set routing-instances VS-2 protocols evpn extended-vlan-list 230
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 210
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 220
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 230
set routing-instances mhevpn instance-type evpn
set routing-instances mhevpn vlan-id 10
set routing-instances mhevpn interface ge-0/1/1.0
set routing-instances mhevpn interface ge-0/1/2.0
set routing-instances mhevpn interface ge-0/1/6.0
set routing-instances mhevpn interface ae0.0
set routing-instances mhevpn routing-interface irb.10

```

```

set routing-instances mhevpn route-distinguisher 198.51.100.2:1
set routing-instances mhevpn vrf-target target:100:1
set routing-instances mhevpn protocols evpn default-gateway do-not-advertise
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf interface irb.1
set routing-instances vrf interface irb.2
set routing-instances vrf interface irb.10
set routing-instances vrf route-distinguisher 198.51.100.2:11
set routing-instances vrf vrf-target target:100:11
set routing-instances vrf vrf-table-label

```

PE3

```

set chassis aggregated-devices ethernet device-count 20
set chassis network-services enhanced-ip
set interfaces ge-2/0/1 gigether-options 802.3ad ae13
set interfaces ge-2/0/2 gigether-options 802.3ad ae13
set interfaces ge-2/0/4 gigether-options 802.3ad ae0
set interfaces ge-2/0/9 flexible-vlan-tagging
set interfaces ge-2/0/9 encapsulation flexible-ethernet-services
set interfaces ge-2/0/9 unit 0 encapsulation vlan-bridge
set interfaces ge-2/0/9 unit 0 vlan-id 10
set interfaces ge-2/0/9 unit 100 family bridge interface-mode trunk
set interfaces ge-2/0/9 unit 100 family bridge vlan-id-list 110
set interfaces ge-2/0/9 unit 100 family bridge vlan-id-list 120
set interfaces ge-2/0/9 unit 100 family bridge vlan-id-list 130
set interfaces ge-2/0/9 unit 200 family bridge interface-mode trunk
set interfaces ge-2/0/9 unit 200 family bridge vlan-id-list 210
set interfaces ge-2/0/9 unit 200 family bridge vlan-id-list 220
set interfaces ge-2/0/9 unit 200 family bridge vlan-id-list 230
set interfaces ge-2/1/0 unit 0 family inet address 192.0.2.5/24
set interfaces ge-2/1/0 unit 0 family iso
set interfaces ge-2/1/0 unit 0 family mpls
set interfaces ge-2/1/6 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set interfaces ae0 unit 110 family bridge interface-mode trunk

```



```

set interfaces ae0 unit 110 family bridge vlan-id-list 110
set interfaces ae0 unit 210 family bridge interface-mode trunk
set interfaces ae0 unit 210 family bridge vlan-id-list 210
set interfaces ae13 flexible-vlan-tagging
set interfaces ae13 encapsulation flexible-ethernet-services
set interfaces ae13 aggregated-ether-options minimum-links 1
set interfaces ae13 unit 0 vlan-tags outer 1300
set interfaces ae13 unit 0 vlan-tags inner 13
set interfaces ae13 unit 0 family inet address 198.51.100.6/24
set interfaces ae13 unit 0 family iso
set interfaces ae13 unit 0 family mpls
set interfaces irb unit 0 family inet address 192.0.2.9/24
set interfaces irb unit 0 mac 00:99:99:99:01:99
set interfaces irb unit 1 family inet address 192.0.2.10/24
set interfaces irb unit 1 mac 00:99:99:99:02:99
set interfaces irb unit 2 family inet address 192.0.2.11/24
set interfaces irb unit 2 mac 00:99:99:99:03:99
set interfaces irb unit 10 family inet address 192.0.2.12/24
set interfaces irb unit 10 mac 00:99:99:99:01:90
set interfaces lo0 unit 0 family inet address 198.51.100.3/32 primary
set interfaces lo0 unit 0 family iso
set routing-options router-id 198.51.100.3
set routing-options autonomous-system 65221
set routing-options forwarding-table export load-balancing-policy
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path pe3tope1 from 198.51.100.3
set protocols mpls label-switched-path pe3tope1 to 198.51.100.1
set protocols mpls label-switched-path pe3tope1 primary direct_to_pe1
set protocols mpls label-switched-path pe3tope2 from 198.51.100.3
set protocols mpls label-switched-path pe3tope2 to 198.51.100.2
set protocols mpls label-switched-path pe3tope2 primary direct_to_pe2
set protocols mpls label-switched-path pe3tope4 from 198.51.100.3
set protocols mpls label-switched-path pe3tope4 to 198.51.100.4
set protocols mpls label-switched-path pe3tope4 primary direct_to_pe4
set protocols mpls path direct_to_pe1 198.51.100.13 strict
set protocols mpls path direct_to_pe2 198.51.100.10 strict
set protocols mpls path direct_to_pe4 198.51.100.11 strict
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group RR type internal
set protocols bgp group RR local-address 198.51.100.3
set protocols bgp group RR family evpn signaling

```

```

set protocols bgp group RR neighbor 203.0.113.0
set protocols isis level 1 disable
set protocols isis interface all level 2 metric 10
set protocols isis interface fxp0.0 disable
set protocols isis interface lo0.0 level 2 metric 0
set protocols ldp deaggregate
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols l2-learning global-mac-table-aging-time 18000
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set routing-instances VS-1 instance-type virtual-switch
set routing-instances VS-1 interface ge-2/0/9.100
set routing-instances VS-1 interface ae0.110
set routing-instances VS-1 route-distinguisher 198.51.100.3:101
set routing-instances VS-1 vrf-target target:100:101
set routing-instances VS-1 protocols evpn extended-vlan-list 110
set routing-instances VS-1 protocols evpn extended-vlan-list 120
set routing-instances VS-1 protocols evpn extended-vlan-list 130
set routing-instances VS-1 protocols evpn default-gateway do-not-advertise
set routing-instances VS-1 bridge-domains bd-110 vlan-id 110
set routing-instances VS-1 bridge-domains bd-110 routing-interface irb.0
set routing-instances VS-1 bridge-domains bd-120 vlan-id 120
set routing-instances VS-1 bridge-domains bd-120 routing-interface irb.1
set routing-instances VS-1 bridge-domains bd-130 vlan-id 130
set routing-instances VS-1 bridge-domains bd-130 routing-interface irb.2
set routing-instances VS-2 instance-type virtual-switch
set routing-instances VS-2 interface ge-2/0/9.200
set routing-instances VS-2 interface ae0.210
set routing-instances VS-2 route-distinguisher 198.51.100.3:201
set routing-instances VS-2 vrf-target target:100:201
set routing-instances VS-2 protocols evpn extended-vlan-list 210
set routing-instances VS-2 protocols evpn extended-vlan-list 220
set routing-instances VS-2 protocols evpn extended-vlan-list 230
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 210
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 220
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 230
set routing-instances mhevpn instance-type evpn
set routing-instances mhevpn vlan-id 10
set routing-instances mhevpn interface ge-2/0/9.0
set routing-instances mhevpn interface ae0.0
set routing-instances mhevpn routing-interface irb.10
set routing-instances mhevpn route-distinguisher 198.51.100.3:1
set routing-instances mhevpn vrf-target target:100:1

```

```

set routing-instances mhevpn protocols evpn default-gateway do-not-advertise
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf interface irb.1
set routing-instances vrf interface irb.2
set routing-instances vrf interface irb.10
set routing-instances vrf route-distinguisher 198.51.100.3:11
set routing-instances vrf vrf-target target:100:11
set routing-instances vrf vrf-table-label

```

PE4

```

set chassis aggregated-devices ethernet device-count 20
set chassis network-services enhanced-ip
set interfaces ge-1/0/3 unit 0 family inet address 192.0.2.7/24
set interfaces ge-1/0/3 unit 0 family iso
set interfaces ge-1/0/3 unit 0 family mpls
set interfaces ge-1/1/1 flexible-vlan-tagging
set interfaces ge-1/1/1 encapsulation flexible-ethernet-services
set interfaces ge-1/1/1 esi 00:44:44:44:44:44:44:44:44:44
set interfaces ge-1/1/1 esi all-active
set interfaces ge-1/1/1 unit 0 encapsulation vlan-bridge
set interfaces ge-1/1/1 unit 0 vlan-id 10
set interfaces ge-1/1/1 unit 100 family bridge interface-mode trunk
set interfaces ge-1/1/1 unit 100 family bridge vlan-id-list 110
set interfaces ge-1/1/1 unit 100 family bridge vlan-id-list 120
set interfaces ge-1/1/1 unit 100 family bridge vlan-id-list 130
set interfaces ge-1/1/1 unit 200 family bridge interface-mode trunk
set interfaces ge-1/1/1 unit 200 family bridge vlan-id-list 210
set interfaces ge-1/1/1 unit 200 family bridge vlan-id-list 220
set interfaces ge-1/1/1 unit 200 family bridge vlan-id-list 230
set interfaces ge-1/1/9 flexible-vlan-tagging
set interfaces ge-1/1/9 encapsulation flexible-ethernet-services
set interfaces ge-1/1/9 unit 0 encapsulation vlan-bridge
set interfaces ge-1/1/9 unit 0 vlan-id 10
set interfaces ge-1/1/9 unit 100 family bridge interface-mode trunk
set interfaces ge-1/1/9 unit 100 family bridge vlan-id-list 110
set interfaces ge-1/1/9 unit 100 family bridge vlan-id-list 120
set interfaces ge-1/1/9 unit 100 family bridge vlan-id-list 130
set interfaces ge-1/1/9 unit 200 family bridge interface-mode trunk
set interfaces ge-1/1/9 unit 200 family bridge vlan-id-list 210
set interfaces ge-1/1/9 unit 200 family bridge vlan-id-list 220

```

```

set interfaces ge-1/1/9 unit 200 family bridge vlan-id-list 230
set interfaces irb unit 0 family inet address 192.0.2.9/24
set interfaces irb unit 0 mac 00:99:99:99:01:99
set interfaces irb unit 1 family inet address 192.0.2.10/24
set interfaces irb unit 1 mac 00:99:99:99:02:99
set interfaces irb unit 2 family inet address 192.0.2.11/24
set interfaces irb unit 2 mac 00:99:99:99:03:99
set interfaces irb unit 10 family inet address 192.0.2.12/24
set interfaces irb unit 10 mac 00:99:99:99:01:90
set interfaces lo0 unit 0 family inet address 198.51.100.4/32 primary
set routing-options router-id 198.51.100.4
set routing-options autonomous-system 65221
set routing-options forwarding-table export load-balancing-policy
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path pe4tope1 from 198.51.100.4
set protocols mpls label-switched-path pe4tope1 to 198.51.100.1
set protocols mpls label-switched-path pe4tope1 primary direct_to_pe1
set protocols mpls label-switched-path pe4tope2 from 198.51.100.4
set protocols mpls label-switched-path pe4tope2 to 198.51.100.2
set protocols mpls label-switched-path pe4tope2 primary direct_to_pe2
set protocols mpls label-switched-path pe4tope3 from 198.51.100.4
set protocols mpls label-switched-path pe4tope3 to 198.51.100.3
set protocols mpls label-switched-path pe4tope3 primary direct_to_pe3
set protocols mpls path pe2_to_pe3 198.51.100.2 strict
set protocols mpls path pe2_to_pe3 198.51.100.3 strict
set protocols mpls path direct_to_pe1 198.51.100.14 strict
set protocols mpls path direct_to_pe2 198.51.100.15 strict
set protocols mpls path direct_to_pe3 198.51.100.16 strict
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group RR type internal
set protocols bgp group RR local-address 198.51.100.4
set protocols bgp group RR family evpn signaling
set protocols bgp group RR neighbor 203.0.113.0
set protocols isis level 1 disable
set protocols isis interface all level 2 metric 10
set protocols isis interface fxp0.0 disable
set protocols isis interface lo0.0 level 2 metric 0
set protocols ldp deaggregate
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols l2-learning global-mac-table-aging-time 18000

```

```

set policy-options policy-statement load-balancing-policy then load-balance per-packet
set routing-instances VS-1 instance-type virtual-switch
set routing-instances VS-1 interface ge-1/1/1.100
set routing-instances VS-1 interface ge-1/1/9.100
set routing-instances VS-1 route-distinguisher 198.51.100.4:101
set routing-instances VS-1 vrf-target target:100:101
set routing-instances VS-1 protocols evpn extended-vlan-list 110
set routing-instances VS-1 protocols evpn extended-vlan-list 120
set routing-instances VS-1 protocols evpn extended-vlan-list 130
set routing-instances VS-1 protocols evpn default-gateway do-not-advertise
set routing-instances VS-1 bridge-domains bd-110 vlan-id 110
set routing-instances VS-1 bridge-domains bd-110 routing-interface irb.0
set routing-instances VS-1 bridge-domains bd-120 vlan-id 120
set routing-instances VS-1 bridge-domains bd-120 routing-interface irb.1
set routing-instances VS-1 bridge-domains bd-130 vlan-id 130
set routing-instances VS-1 bridge-domains bd-130 routing-interface irb.2
set routing-instances VS-2 instance-type virtual-switch
set routing-instances VS-2 interface ge-1/1/1.200
set routing-instances VS-2 interface ge-1/1/9.200
set routing-instances VS-2 route-distinguisher 198.51.100.4:201
set routing-instances VS-2 vrf-target target:100:201
set routing-instances VS-2 protocols evpn extended-vlan-list 210
set routing-instances VS-2 protocols evpn extended-vlan-list 220
set routing-instances VS-2 protocols evpn extended-vlan-list 230
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 210
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 220
set routing-instances VS-2 bridge-domains bd-a vlan-id-list 230
set routing-instances mhevpn instance-type evpn
set routing-instances mhevpn vlan-id 10
set routing-instances mhevpn interface ge-1/1/1.0
set routing-instances mhevpn interface ge-1/1/9.0
set routing-instances mhevpn routing-interface irb.10
set routing-instances mhevpn route-distinguisher 198.51.100.4:1
set routing-instances mhevpn vrf-target target:100:1
set routing-instances mhevpn protocols evpn default-gateway do-not-advertise
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf interface irb.1
set routing-instances vrf interface irb.2
set routing-instances vrf interface irb.10
set routing-instances vrf route-distinguisher 198.51.100.4:11

```

```
set routing-instances vrf vrf-target target:100:11
set routing-instances vrf vrf-table-label
```

P (RR)

```
set interfaces ge-1/0/5 unit 0 family inet address 192.0.2.8/24
set interfaces ge-1/1/2 unit 0 family inet address 192.0.2.6/24
set interfaces ge-1/1/3 unit 0 family inet address 192.0.2.4/24
set interfaces ge-1/1/4 unit 0 family inet address 192.0.2.2/24
set interfaces lo0 unit 0 family inet address 203.0.113.0
set protocols bgp group RR type internal
set protocols bgp group RR local-address 203.0.113.0
set protocols bgp group RR family evpn signaling
set protocols bgp group RR cluster 1.2.3.4
set protocols bgp group RR neighbor 198.51.100.1
set protocols bgp group RR neighbor 198.51.100.2
set protocols bgp group RR neighbor 198.51.100.3
set protocols bgp group RR neighbor 198.51.100.4
set protocols isis interface all level 1 disable
set protocols ldp interface all
set routing-options router-id 203.0.113.0
set routing-options autonomous-system 65221
```

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 Router PE1:



NOTE: Repeat this procedure for all other multihomed PE routers after modifying the appropriate interface names, addresses, and other parameters.

1. Configure the MX Series router to operate in the enhanced-ip mode because the EVPN active-active functionality is supported on routers with MPCs and MIC interfaces only.

A system reboot is required on committing this configuration.

```
[edit chassis]
user@PE1# set network-services enhanced-ip
```

2. Specify the number of aggregated Ethernet interfaces to be created.

```
[edit chassis]
user@PE1# set aggregated-devices ethernet device-count 20
```

3. Configure Router PE1 interfaces within the ae0 aggregated bundle toward the multihomed customer site, Router CE10.

- a. Assign interfaces ge-1/2/3 and ge-1/2/4 within the ae0 aggregated bundle.

```
[edit interfaces]
user@PE1# set ge-1/2/3 gigether-options 802.3ad ae0
user@PE1# set ge-1/2/4 gigether-options 802.3ad ae0
```

- b. Configure the ae0 aggregated bundle parameters for VLAN tagging and encapsulation.

```
[edit interfaces]
user@PE1# set ae0 flexible-vlan-tagging
user@PE1# set ae0 encapsulation flexible-ethernet-services
```

- c. Assign an ESI value for the first Ethernet segment and enable EVPN active-active multihoming for the ae0 aggregated bundle.

```
[edit interfaces]
user@PE1# set ae0 esi 00:11:11:11:11:11:11:11:11
user@PE1# set ae0 esi all-active
```

- d. Configure a trunk interface on the bridge network for the ae0 aggregated bundle.

```
[edit interfaces]
user@PE1# set ae0 unit 0 encapsulation vlan-bridge
user@PE1# set ae0 unit 0 vlan-id 10
```

```

user@PE1# set ae0 unit 110 family bridge interface-mode trunk
user@PE1# set ae0 unit 110 family bridge vlan-id-list 110
user@PE1# set ae0 unit 210 family bridge interface-mode trunk
user@PE1# set ae0 unit 210 family bridge vlan-id-list 210

```

4. Configure the other Router PE1 interfaces toward the multihomed customer site, Router CE10.
 - a. Configure the VLAN tagging and encapsulation parameters for the ge-1/2/2 PE1 interface.

```

[edit interfaces]
user@PE1# set ge-1/2/2 flexible-vlan-tagging
user@PE1# set ge-1/2/2 encapsulation flexible-ethernet-services

```

- b. Assign an ESI value for the second Ethernet segment and enable EVPN active-active multihoming for the ge-1/2/2 PE1 interface.

```

[edit interfaces]
user@PE1# set ge-1/2/2 esi 00:22:22:22:22:22:22:22
user@PE1# set ge-1/2/2 esi all-active

```

- c. Configure a trunk interface on the bridge network for the ge-1/2/2 PE1 interface.

```

[edit interfaces]
user@PE1# set ge-1/2/2 unit 0 encapsulation vlan-bridge
user@PE1# set ge-1/2/2 unit 0 vlan-id 20
user@PE1# set ge-1/2/2 unit 120 family bridge interface-mode trunk
user@PE1# set ge-1/2/2 unit 120 family bridge vlan-id-list 120
user@PE1# set ge-1/2/2 unit 220 family bridge interface-mode trunk
user@PE1# set ge-1/2/2 unit 220 family bridge vlan-id-list 220

```

- d. Configure the VLAN tagging and encapsulation parameters for the ge-1/2/1 PE1 interface.

```

[edit interfaces]
user@PE1# set ge-1/2/1 flexible-vlan-tagging
user@PE1# set ge-1/2/1 encapsulation flexible-ethernet-services

```


- e. Assign an ESI value for the third Ethernet segment and enable EVPN active-active multihoming for the ge-1/2/1 PE1 interface.

```
[edit interfaces]
user@PE1# set ge-1/2/1 esi 00:33:33:33:33:33:33:33
user@PE1# set ge-1/2/1 esi all-active
```

- f. Configure a trunk interface on the bridge network for the ge-1/2/1 PE1 interface.

```
[edit interfaces]
user@PE1# set ge-1/2/1 unit 0 encapsulation vlan-bridge
user@PE1# set ge-1/2/1 unit 0 vlan-id 30
user@PE1# set ge-1/2/1 unit 130 family bridge interface-mode trunk
user@PE1# set ge-1/2/1 unit 130 family bridge vlan-id-list 130
user@PE1# set ge-1/2/1 unit 230 family bridge interface-mode trunk
user@PE1# set ge-1/2/1 unit 230 family bridge vlan-id-list 230
```

5. Configure Router PE1 interfaces toward Router PE2.

- a. Assign the interfaces ge-1/0/5 and ge-1/1/6 within the ae12 aggregated bundle.

```
[edit interfaces]
user@PE1# set ge-1/0/5 gigether-options 802.3ad ae12
user@PE1# set ge-1/1/6 gigether-options 802.3ad ae12
```

- b. Specify the minimum number of links for the ae12 aggregated bundle to be labeled “up”.

```
[edit interfaces]
user@PE1# set ae12 aggregated-ether-options minimum-links 1
```

- c. Assign an IP address for the ae12 aggregated bundle and enable MPLS and IS-IS protocol families on the bundle.

```
[edit interfaces]
user@PE1# set ae12 unit 0 family inet address 198.51.100.12/24
user@PE1# set ae12 unit 0 family iso
user@PE1# set ae12 unit 0 family mpls
```

- d. Configure the VLAN tagging and encapsulation parameters for the ae12 aggregated bundle and assign VLAN ID 1200 for the bundle.

```
[edit interfaces]
user@PE1# set ae12 flexible-vlan-tagging
user@PE1# set ae12 encapsulation flexible-ethernet-services
user@PE1# set ae12 unit 0 vlan-id 1200
```

6. Configure Router PE1 interfaces toward Router PE3.

- a. Assign the interfaces ge-1/0/3 and ge-1/0/4 within the ae13 aggregated bundle.

```
[edit interfaces]
user@PE1# set ge-1/0/3 gigether-options 802.3ad ae13
user@PE1# set ge-1/0/4 gigether-options 802.3ad ae13
```

- b. Specify the minimum number of links for the ae13 aggregated bundle to be labeled “up”.

```
[edit interfaces]
user@PE1# set ae13 aggregated-ether-options minimum-links 1
```

- c. Assign an IP address for the ae13 aggregated bundle and enable MPLS and IS-IS protocol families on the bundle.

```
[edit interfaces]
user@PE1# set ae13 unit 0 family inet address 198.51.100.13/24
user@PE1# set ae13 unit 0 family iso
user@PE1# set ae13 unit 0 family mpls
```

- d. Configure the VLAN tagging and encapsulation parameters for the ae12 aggregated bundle and assign the inner and outer VLAN tags for the bundle.

```
[edit interfaces]
user@PE1# set ae13 flexible-vlan-tagging
user@PE1# set ae13 encapsulation flexible-ethernet-services
user@PE1# set ae13 unit 0 vlan-tags outer 1300
user@PE1# set ae13 unit 0 vlan-tags inner 13
```

7. Configure the Router PE1 interface toward the single-homed customer site, Router CE1.
 - a. Configure the VLAN tagging and encapsulation parameters for the ge-1/3/0 PE1 interface.

```
[edit interfaces]
user@PE1# set ge-1/3/0 flexible-vlan-tagging
user@PE1# set ge-1/3/0 encapsulation flexible-ethernet-services
```

- b. Configure a trunk interface on the bridge network for the ge-1/3/0 PE1 interface.

```
[edit interfaces]
user@PE1# set ge-1/3/0 unit 0 encapsulation vlan-bridge
user@PE1# set ge-1/3/0 unit 0 vlan-id 10
user@PE1# set ge-1/3/0 unit 100 family bridge interface-mode trunk
user@PE1# set ge-1/3/0 unit 100 family bridge vlan-id-list 110
user@PE1# set ge-1/3/0 unit 100 family bridge vlan-id-list 120
user@PE1# set ge-1/3/0 unit 100 family bridge vlan-id-list 130
user@PE1# set ge-1/3/0 unit 200 family bridge interface-mode trunk
user@PE1# set ge-1/3/0 unit 200 family bridge vlan-id-list 210
user@PE1# set ge-1/3/0 unit 200 family bridge vlan-id-list 220
user@PE1# set ge-1/3/0 unit 200 family bridge vlan-id-list 230
```

8. Configure the Router PE1 interface toward Router P (RR) and enable the MPLS and IS-IS protocol families for the interface.

```
[edit interfaces]
user@PE1# set ge-1/1/4 unit 0 family inet address 192.0.2.1/24
user@PE1# set ge-1/1/4 unit 0 family iso
user@PE1# set ge-1/1/4 unit 0 family mpls
```

9. Configure an IRB interface for Router PE1.

```
[edit interfaces]
user@PE1# set irb unit 0 family inet address 192.0.2.9/24
user@PE1# set irb unit 0 mac 00:99:99:99:01:99
user@PE1# set irb unit 1 family inet address 192.0.2.10/24
user@PE1# set irb unit 1 mac 00:99:99:99:02:99
user@PE1# set irb unit 2 family inet address 192.0.2.11/24
user@PE1# set irb unit 2 mac 00:99:99:99:03:99
```

```

user@PE1# set irb unit 10 family inet address 192.0.2.12/24
user@PE1# set irb unit 10 mac 00:99:99:99:01:90

```

10. Configure the loopback interface for Router PE1.

```

[edit interfaces]
user@PE1# set lo0 unit 0 family inet address 198.51.100.1/24 primary
user@PE1# set lo0 unit 0 family iso

```

11. Assign a router ID and the autonomous system number for Router PE1.

```

[edit routing-options]
user@PE1# set router-id 198.51.100.1
user@PE1# set autonomous-system 65221

```

12. Assign a load-balancing policy to the forwarding table of Router PE1.

```

[edit routing-options]
user@PE1# set forwarding-table export load-balancing-policy

```

13. Configure IS-IS on Router PE1.

```

[edit protocols]
user@PE1# set isis level 1 disable
user@PE1# set isis interface all level 2 metric 10
user@PE1# set isis interface fxp0.0 disable
user@PE1# set isis interface lo0.0 level 2 metric 0

```

14. Configure an internal BGP group for Router PE1 to peer with route reflector, Router P.

```

[edit protocols]
user@PE1# set bgp group RR type internal
user@PE1# set bgp group RR local-address 198.51.100.1
user@PE1# set bgp group RR neighbor 203.0.113.0

```

15. Enable EVPN signaling for the RR BGP group on Router PE1.

```
[edit protocols]
user@PE1# set bgp group RR family evpn signaling
```

16. Configure RSVP, LDP, MPLS, EVPN, and L2 learning on Router PE1.

```
[edit protocols]
user@PE1# set rsvp interface all
user@PE1# set rsvp interface fxp0.0 disable
user@PE1# set ldp deaggregate
user@PE1# set ldp interface all
user@PE1# set ldp interface fxp0.0 disable
user@PE1# set mpls interface all
user@PE1# set mpls interface fxp0.0 disable
user@PE1# set evpn
user@PE1# set l2-learning global-mac-table-aging-time 18000
```

17. Configure label-switched paths between the PE routers.

```
[edit protocols]
user@PE1# set mpls label-switched-path pe1tope2 from 198.51.100.1
user@PE1# set mpls label-switched-path pe1tope2 to 198.51.100.2
user@PE1# set mpls label-switched-path pe1tope2 primary direct_to_pe2
user@PE1# set mpls label-switched-path pe1tope3 from 198.51.100.1
user@PE1# set mpls label-switched-path pe1tope3 to 198.51.100.3
user@PE1# set mpls label-switched-path pe1tope3 primary direct_to_pe3
user@PE1# set mpls label-switched-path pe1tope4 from 198.51.100.1
user@PE1# set mpls label-switched-path pe1tope4 to 198.51.100.4
user@PE1# set mpls label-switched-path pe1tope4 primary direct_to_pe4
```

18. Configure MPLS paths from Router PE1 to other PE routers.

```
user@PE1# set mpls path pe4_to_pe3 198.51.100.4 strict
user@PE1# set mpls path pe4_to_pe3 198.51.100.3 strict
user@PE1# set mpls path direct_to_pe2 198.51.100.5 strict
user@PE1# set mpls path direct_to_pe3 198.51.100.6 strict
user@PE1# set mpls path direct_to_pe4 198.51.100.9 strict
```

```
user@PE1# set mpls path pe2_to_pe3 198.51.100.2 strict
user@PE1# set mpls path pe2_to_pe3 198.51.100.3 strict
```

19. Configure the load-balancing policy to enable load balancing per packet.

```
[edit policy-options]
user@PE1# set policy-statement load-balancing-policy then load-balance per-packet
```

20. Configure the first virtual switch routing instance.

- a. Configure the routing-instance type and assign Router PE1 interfaces to the routing instance.

```
[edit routing-instances]
user@PE1# set VS-1 instance-type virtual-switch
user@PE1# set VS-1 interface ge-1/2/1.130
user@PE1# set VS-1 interface ge-1/2/2.120
user@PE1# set VS-1 interface ge-1/3/0.100
user@PE1# set VS-1 interface ae0.110
```

- b. Configure the route distinguisher and the VPN routing and forwarding (VRF) target for the VS-1 routing instance.

```
[edit routing-instances]
user@PE1# set VS-1 route-distinguisher 198.51.100.1:101
user@PE1# set VS-1 vrf-target target:100:101
```

- c. Configure EVPN and assign VLANs to the VS-1 routing instance.

```
[edit routing-instances]
user@PE1# set VS-1 protocols evpn extended-vlan-list 110
user@PE1# set VS-1 protocols evpn extended-vlan-list 120
user@PE1# set VS-1 protocols evpn extended-vlan-list 130
user@PE1# set VS-1 protocols evpn default-gateway do-not-advertise
```

- d. Configure the bridge domains and their associated VLANs and IRB interfaces for the VS-1 routing instance.

```
[edit routing-instances]
user@PE1# set VS-1 bridge-domains bd-110 vlan-id 110
user@PE1# set VS-1 bridge-domains bd-110 routing-interface irb.0
user@PE1# set VS-1 bridge-domains bd-120 vlan-id 120
user@PE1# set VS-1 bridge-domains bd-120 routing-interface irb.1
user@PE1# set VS-1 bridge-domains bd-130 vlan-id 130
user@PE1# set VS-1 bridge-domains bd-130 routing-interface irb.2
```

21. Configure the second virtual switch routing instance.

- a. Configure the routing-instance type and assign Router PE1 interfaces to the routing instance.

```
[edit routing-instances]
user@PE1# set VS-2 instance-type virtual-switch
user@PE1# set VS-2 interface ge-1/2/1.230
user@PE1# set VS-2 interface ge-1/2/2.220
user@PE1# set VS-2 interface ge-1/3/0.200
user@PE1# set VS-2 interface ae0.210
```

- b. Configure the route distinguisher and the VPN routing and forwarding (VRF) target for the VS-2 routing instance.

```
[edit routing-instances]
user@PE1# set VS-2 route-distinguisher 198.51.100.1:201
user@PE1# set VS-2 vrf-target target:100:201
```

- c. Configure EVPN and assign VLANs to the VS-2 routing instance.

```
[edit routing-instances]
user@PE1# set VS-2 protocols evpn extended-vlan-list 210
user@PE1# set VS-2 protocols evpn extended-vlan-list 220
user@PE1# set VS-2 protocols evpn extended-vlan-list 230
```

- d. Configure the bridge domains and their associated VLANs for the VS-2 routing instance.

```
[edit routing-instances]
user@PE1# set routing-instances VS-2 bridge-domains bd-a vlan-id-list 210
user@PE1# set routing-instances VS-2 bridge-domains bd-a vlan-id-list 220
user@PE1# set routing-instances VS-2 bridge-domains bd-a vlan-id-list 230
```

22. Configure the multihomed EVPN routing instance.

- a. Configure the routing-instance type and assign VLANs and Router PE1 interfaces to the routing instance.

```
[edit routing-instances]
user@PE1# set mhevpn instance-type evpn
user@PE1# set mhevpn vlan-id 10
user@PE1# set mhevpn interface ge-1/2/1.0
user@PE1# set mhevpn interface ge-1/2/2.0
user@PE1# set mhevpn interface ge-1/3/0.0
user@PE1# set mhevpn interface ae0.0
user@PE1# set mhevpn routing-interface irb.10
```

- b. Configure the route distinguisher and the VPN routing and forwarding (VRF) target for the mhevpn routing instance.

```
[edit routing-instances]
user@PE1# set mhevpn route-distinguisher 198.51.100.1:1
user@PE1# set mhevpn vrf-target target:100:1
```

- c. Configure EVPN to the mhevpn routing instance.

```
[edit routing-instances]
user@PE1# set routing-instances mhevpn protocols evpn default-gateway do-not-advertise
```

23. Configure the default routing instance.

- a. Configure the routing-instance type and assign IRB interfaces to the routing instance.

```
[edit routing-instances]
user@PE1# set vrf instance-type vrf
```



```

user@PE1# set vrf interface irb.0
user@PE1# set vrf interface irb.1
user@PE1# set vrf interface irb.2
user@PE1# set vrf interface irb.10

```

- b. Configure the route distinguisher and the VPN routing and forwarding (VRF) target for the vrf routing instance.

```

[edit routing-instances]
user@PE1# set vrf route-distinguisher 198.51.100.1:11
user@PE1# set vrf vrf-target target:100:11
user@PE1# set vrf vrf-table-label

```

Configuration on EX9200 Switches

Step-by-Step Procedure

Several configuration statements used to configure active-active mode differ on EX9200 switches from those used on MX Series routers. This procedure shows which configuration statements are specific to EX9200 switches. All other configuration in this example applies both to EX9200 switches and MX Series routers.

1. To configure a trunk interface, include the family `ethernet-switching` statements instead of the family `bridge` statements in all occurrences.

```

[edit interfaces]
user@PE#1# set interfaces ge-1/2/1 unit 130 family ethernet-switching interface-mode trunk

```

2. To configure the Layer 2 Ethernet switching domain, include the `vlan members` (*vlan-id* | *name*) statement instead of the `vlan-id-list` *vlan-id* statement in all occurrences.

```

[edit interfaces]
user@PE#1# set interfaces ge-1/2/1 unit 130 family ethernet-switching vlan members 130

```

3. To configure the VLAN domain and associated VLANs for each routing instance, include the `vlan name` statement, instead of the `bridge-domains` statement in all occurrences.

```
[edit]
user@PE#1# set routing-instances VS-1 vlans bd-110 vlan-id 110
```

4. To configure the IRB interface in each routing instance, include the `l3-interface irb-interface-name` statement instead of the `routing-interface` statement in all occurrences.

```
[edit]
user@PE#1# set routing-instances VS-1 vlans bd-110 l3-interface irb.0
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, 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 chassis
aggregated-devices {
  ethernet {
    device-count 20;
  }
}
network-services enhanced-ip;
```

```
user@PE1 show interfaces
ge-1/0/3 {
  gigether-options {
    802.3ad ae13;
  }
}
ge-1/0/4 {
  gigether-options {
    802.3ad ae13;
  }
}
ge-1/0/5 {
```

```

    gigaether-options {
        802.3ad ae12;
    }
}
ge-1/1/4 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
        family iso;
        family mpls;
    }
}
ge-1/1/6 {
    gigaether-options {
        802.3ad ae12;
    }
}
ge-1/2/1 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
        00:33:33:33:33:33:33:33:33:33;
        all-active;
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 30;
    }
    unit 130 {
        family bridge {
            interface-mode trunk;
            vlan-id-list 130;
        }
    }
    unit 230 {
        family bridge {
            interface-mode trunk;
            vlan-id-list 230;
        }
    }
}
ge-1/2/2 {

```

```

flexible-vlan-tagging;
encapsulation flexible-ethernet-services;
esi {
    00:22:22:22:22:22:22:22:22;
    all-active;
}
unit 0 {
    encapsulation vlan-bridge;
    vlan-id 20;
}
unit 120 {
    family bridge {
        interface-mode trunk;
        vlan-id-list 120;
    }
}
unit 220 {
    family bridge {
        interface-mode trunk;
        vlan-id-list 220;
    }
}
}
ge-1/2/3 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/2/4 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/3/0 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 10;
    }
    unit 100 {
        family bridge {
            interface-mode trunk;

```

```

        vlan-id-list [ 110 120 130 ];
    }
}
unit 200 {
    family bridge {
        interface-mode trunk;
        vlan-id-list [ 210 220 230 ];
    }
}
}
ae0 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
        00:11:11:11:11:11:11:11:11:11;
        all-active;
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 10;
    }
    unit 110 {
        family bridge {
            interface-mode trunk;
            vlan-id-list 110;
        }
    }
    unit 210 {
        family bridge {
            interface-mode trunk;
            vlan-id-list 210;
        }
    }
}
}
ae12 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    aggregated-ether-options {
        minimum-links 1;
    }
    unit 0 {
        vlan-id 1200;
        family inet {

```

```

        address 198.51.100.12/24;
    }
    family iso;
    family mpls;
}
}
ae13 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    aggregated-ether-options {
        minimum-links 1;
    }
    unit 0 {
        vlan-tags outer 1300 inner 13;
        family inet {
            address 198.51.100.13/24;
        }
        family iso;
        family mpls;
    }
}
}
irb {
    unit 0 {
        family inet {
            address 192.0.2.9/24;
        }
        mac 00:99:99:99:01:99;
    }
    unit 1 {
        family inet {
            address 192.0.2.10/24;
        }
        mac 00:99:99:99:02:99;
    }
    unit 2 {
        family inet {
            address 192.0.2.11/24;
        }
        mac 00:99:99:99:03:99;
    }
    unit 10 {
        family inet {
            address 192.0.2.12/24;

```

```

    }
    mac 00:99:99:99:01:90;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 198.51.100.1/24 {
        primary;
      }
    }
    family iso;
  }
}

```

```

user@PE1# show routing-options
router-id 198.51.100.1;
autonomous-system 65221;
forwarding-table {
  export load-balancing-policy;
}

```

```

user@PE1# show protocols
rsvp {
  interface all;
  interface fxp0.0 {
    disable;
  }
}
mpls {
  label-switched-path pe1tope2 {
    from 198.51.100.1;
    to 198.51.100.2;
    primary direct_to_pe2;
  }
  label-switched-path pe1tope3 {
    from 198.51.100.1;
    to 198.51.100.3;
    primary direct_to_pe3;
  }
}

```

```

label-switched-path pe1tope4 {
    from 198.51.100.1;
    to 198.51.100.4;
    primary direct_to_pe4;
}
path pe4_to_pe3 {
    198.51.100.4 strict;
    198.51.100.3 strict;
}
path direct_to_pe2 {
    198.51.100.5 strict;
}
path direct_to_pe3 {
    198.51.100.6 strict;
}
path direct_to_pe4 {
    198.51.100.9 strict;
}
path pe2_to_pe3 {
    198.51.100.2 strict;
    198.51.100.3 strict;
}
interface all;
interface fxp0.0 {
    disable;
}
}
bgp {
    group RR {
        type internal;
        local-address 198.51.100.1;
        family evpn {
            signaling;
        }
        neighbor 203.0.113.0;
    }
}
isis {
    level 1 disable;
    interface all {
        level 2 metric 10;
    }
    interface fxp0.0 {

```



```

        disable;
    }
    interface lo0.0 {
        level 2 metric 0;
    }
}
ldp {
    deaggregate;
    interface all;
    interface fxp0.0 {
        disable;
    }
}
evpn {
}
l2-learning {
    global-mac-table-aging-time 18000;
}

```

```

user@PE1# show policy-options
policy-statement load-balancing-policy {
    then {
        load-balance per-packet;
    }
}

```

```

user@PE1# show routing-instances
VS-1 {
    instance-type virtual-switch;
    interface ge-1/2/1.130;
    interface ge-1/2/2.120;
    interface ge-1/3/0.100;
    interface ae0.110;
    route-distinguisher 198.51.100.1:101;
    vrf-target target:100:101;
    protocols {
        evpn {
            extended-vlan-list [ 110 120 130 ];
            default-gateway do-not-advertise;
        }
    }
}

```

```

}
bridge-domains {
    bd-110 {
        vlan-id 110;
        routing-interface irb.0;
    }
    bd-120 {
        vlan-id 120;
        routing-interface irb.1;
    }
    bd-130 {
        vlan-id 130;
        routing-interface irb.2;
    }
}
}
VS-2 {
    instance-type virtual-switch;
    interface ge-1/2/1.230;
    interface ge-1/2/2.220;
    interface ge-1/3/0.200;
    interface ae0.210;
    route-distinguisher 198.51.100.1:201;
    vrf-target target:100:201;
    protocols {
        evpn {
            extended-vlan-list [ 210 220 230 ];
        }
    }
    bridge-domains {
        bd-a {
            vlan-id-list [ 210 220 230 ];
        }
    }
}
}
mhevpn {
    instance-type evpn;
    vlan-id 10;
    interface ge-1/2/1.0;
    interface ge-1/2/2.0;
    interface ge-1/3/0.0;
    interface ae0.0;
    routing-interface irb.10;

```

```
route-distinguisher 198.51.100.1:1;
vrf-target target:100:1;
protocols {
    evpn {
        default-gateway do-not-advertise;
    }
}
}
vrf {
    instance-type vrf;
    interface irb.0;
    interface irb.1;
    interface irb.2;
    interface irb.10;
    route-distinguisher 198.51.100.1:11;
    vrf-target target:100:11;
    vrf-table-label;
}
```

Verification

IN THIS SECTION

- [Verifying VPN Services in the Core | 419](#)
- [Verifying the EVPN Instance Status | 421](#)
- [Verifying the Autodiscovery Routes per Ethernet Segment | 428](#)
- [Verifying the Ethernet Segment Route | 431](#)
- [Verifying the DF Status | 434](#)
- [Verifying the BDF Status | 435](#)
- [Verifying the Remote IRB and Host IP | 436](#)

Confirm that the configuration is working properly.

Verifying VPN Services in the Core

Purpose

Ensure that the protocols in the VPN core are functioning properly.

Action

From operational mode, enter the `show isis adjacency` command.

```
user@PE1> show isis adjacency
Interface          System          L State          Hold (secs) SNPA
ge-1/2/4.0         CE10            2 Up              24 5c:5e:ab:e:6f:4
```

From operational mode, enter the `show bgp summary` command.

```
user@PE1> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table          Tot Paths  Act Paths Suppressed  History Damp State  Pending
bgp.evpn.0
              45         45         0         0         0         0
Peer          AS      InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
203.0.113.0   65221      90      26       0       0       3:18 Establ
  bgp.evpn.0: 45/45/45/0
  VS-1.evpn.0: 19/19/19/0
  VS-2.evpn.0: 19/19/19/0
  mhevpn.evpn.0: 13/13/13/0
  __default_evpn__.evpn.0: 4/4/4/0
user@P> show bgp summary
Groups: 1 Peers: 4 Down peers: 0
Table          Tot Paths  Act Paths Suppressed  History Damp State  Pending
bgp.evpn.0
              68         68         0         0         0         0
Peer          AS      InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
198.51.100.1   65221      25      90       0       0       3:04 Establ
  bgp.evpn.0: 22/22/22/0
198.51.100.2   65221      32      80       0       0       6:12 Establ
  bgp.evpn.0: 22/22/22/0
198.51.100.3   65221      31      62       0       0       6:58 Establ
```

```

bgp.evpn.0: 12/12/12/0
198.51.100.4      65221      28      88      0      0      6:04 Establ
bgp.evpn.0: 12/12/12/0

```

From operational mode, enter the `show mpls lsp` command.

```

user@PE1> show mpls lsp
Ingress LSP: 3 sessions
To          From          State Rt P    ActivePath    LSPname
198.51.100.2 198.51.100.1 Up    0 *    direct_to_pe2 pe1tope2
198.51.100.3 198.51.100.1 Up    0 *    direct_to_pe3 pe1tope3
198.51.100.4 198.51.100.1 Up    0 *    direct_to_pe4 pe1tope4
Total 3 displayed, Up 3, Down 0

Egress LSP: 3 sessions
To          From          State Rt Style Labelin Labelout LSPname
198.51.100.1 198.51.100.3 Up    0 1 FF      3      - pe3tope1
198.51.100.1 198.51.100.4 Up    0 1 FF      3      - pe4tope1
198.51.100.1 198.51.100.2 Up    0 1 FF      3      - pe2tope1
Total 3 displayed, Up 3, Down 0

Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

```

From operational mode, enter the `show interface ae* terse` command.

```

user@PE1> show interface ae* terse
Interface          Admin Link Proto  Local          Remote
ae0                up    up
ae0.0              up    up  bridge
ae0.110            up    up  bridge
ae0.210            up    up  bridge
ae0.32767          up    up  multiservice
ae12               up    up
ae12.0             up    up  inet    198.51.100.12/24
                  up    up      iso
                  up    up      mpls
                  up    up  multiservice
ae12.32767         up    up  multiservice
ae13               up    up
ae13.0             up    up  inet    198.51.100.13/24

```

The protocols IS-IS, BGP and MPLS are up and running. The aggregated bundles configured on Router PE1 are up.

Purpose

Action

```

user@PE1> show evpn instance extensive

Instance: VS-1

Route Distinguisher: 198.51.100.1:101
Per-instance MAC route label: 301664

MAC database status          Local  Remote
Total MAC addresses:         0        0
Default gateway MAC addresses: 3        0

Number of local interfaces: 4 (3 up)
Interface name  ESI                               Mode          Status
ae0.110        00:11:11:11:11:11:11:11:11:11  all-active    Up
ge-1/2/1.130   00:33:33:33:33:33:33:33:33:33  all-active    Up
ge-1/2/2.120   00:22:22:22:22:22:22:22:22:22  all-active    Up
ge-1/3/0.100   00:00:00:00:00:00:00:00:00:00  single-homed  Up

Number of IRB interfaces: 3 (3 up)
Interface name  VLAN ID  Status  L3 context
irb.0           110      Up      vrf
irb.1           120      Up      vrf
irb.2           130      Up      vrf

Number of bridge domains: 3
VLAN ID  Intfs / up  Mode          MAC sync  IM route label
110      2    1    Extended    Enabled    301984

```

```

120      2  1  Extended      Enabled  302000
130      2  1  Extended      Enabled  302016

```

Number of neighbors: 3

198.51.100.2

Received routes

```

MAC address advertisement:      0
MAC+IP address advertisement:   0
Inclusive multicast:            3
Ethernet auto-discovery:        6

```

198.51.100.3

Received routes

```

MAC address advertisement:      0
MAC+IP address advertisement:   0
Inclusive multicast:            1
Ethernet auto-discovery:        2

```

198.51.100.4

Received routes

```

MAC address advertisement:      0
MAC+IP address advertisement:   0
Inclusive multicast:            3
Ethernet auto-discovery:        2

```

Number of ethernet segments: 4

ESI: 00:11:11:11:11:11:11:11:11

Status: Resolved by IFL ae0.110

Local interface: ae0.110, Status: Up/Forwarding

Number of remote PEs connected: 2

Remote PE	MAC label	Aliasing label	Mode
198.51.100.3	0	305584	all-active
198.51.100.2	0	306000	all-active

Designated forwarder: 198.51.100.3

Backup forwarder: 198.51.100.1

Backup forwarder: 198.51.100.2

Advertised MAC label: 301792

Advertised aliasing label: 301792

Advertised split horizon label: 301808

ESI: 00:22:22:22:22:22:22:22:22

Status: Resolved by IFL ge-1/2/2.120

Local interface: ge-1/2/2.120, Status: Up/Forwarding

Number of remote PEs connected: 1

Remote PE	MAC label	Aliasing label	Mode
198.51.100.2	0	306032	all-active

Designated forwarder: 198.51.100.1

Backup forwarder: 198.51.100.2

Advertised MAC label: 301824
 Advertised aliasing label: 301824
 Advertised split horizon label: 301840
 ESI: 00:33:33:33:33:33:33:33:33:33
 Status: Resolved by IFL ge-1/2/1.130
 Local interface: ge-1/2/1.130, Status: Up/Forwarding
Number of remote PEs connected: 1

Remote PE	MAC label	Aliasing label	Mode
198.51.100.2	0	306064	all-active

 Designated forwarder: 198.51.100.1
 Backup forwarder: 198.51.100.2
 Advertised MAC label: 301856
 Advertised aliasing label: 301856
 Advertised split horizon label: 301872
 ESI: 00:44:44:44:44:44:44:44:44:44
 Status: Resolved by NH 1048613
 Number of remote PEs connected: 1

Remote PE	MAC label	Aliasing label	Mode
198.51.100.4	0	305152	all-active

Instance: VS-2

Route Distinguisher: 198.51.100.1:201

Per-instance MAC route label: 301696

MAC database status Local Remote

Total MAC addresses: 0 0

Default gateway MAC addresses: 0 0

Number of local interfaces: 4 (3 up)

Interface name	ESI	Mode	Status
ae0.210	00:11:11:11:11:11:11:11:11:11	all-active	Up
ge-1/2/1.230	00:33:33:33:33:33:33:33:33:33	all-active	Up
ge-1/2/2.220	00:22:22:22:22:22:22:22:22:22	all-active	Up
ge-1/3/0.200	00:00:00:00:00:00:00:00:00:00	single-homed	Down

Number of IRB interfaces: 0 (0 up)

Number of bridge domains: 3

VLAN ID	Intfs / up	Mode	MAC sync	IM route label
210	2 1	Extended	Enabled	302032
220	2 1	Extended	Enabled	302048
230	2 1	Extended	Enabled	302064

Number of neighbors: 3

198.51.100.2

Received routes

MAC address advertisement: 0

MAC+IP address advertisement: 0


```

    Inclusive multicast:          3
    Ethernet auto-discovery:      6

```

```
198.51.100.3
```

```
Received routes
```

```

    MAC address advertisement:    0
    MAC+IP address advertisement: 0
    Inclusive multicast:          1
    Ethernet auto-discovery:      2

```

```
198.51.100.4
```

```
Received routes
```

```

    MAC address advertisement:    0
    MAC+IP address advertisement: 0
    Inclusive multicast:          3
    Ethernet auto-discovery:      2

```

```
Number of ethernet segments: 4
```

```
ESI: 00:11:11:11:11:11:11:11:11
```

```
Status: Resolved by IFL ae0.210
```

```
Local interface: ae0.210, Status: Up/Forwarding
```

```
Number of remote PEs connected: 2
```

Remote PE	MAC label	Aliasing label	Mode
198.51.100.3	0	305648	all-active
198.51.100.2	0	306096	all-active

```
Designated forwarder: 198.51.100.1
```

```
Backup forwarder: 198.51.100.2
```

```
Backup forwarder: 198.51.100.3
```

```
Advertised MAC label: 301888
```

```
Advertised aliasing label: 301888
```

```
Advertised split horizon label: 301808
```

```
ESI: 00:22:22:22:22:22:22:22:22
```

```
Status: Resolved by IFL ge-1/2/2.220
```

```
Local interface: ge-1/2/2.220, Status: Up/Forwarding
```

```
Number of remote PEs connected: 1
```

Remote PE	MAC label	Aliasing label	Mode
198.51.100.2	0	306112	all-active

```
Designated forwarder: 198.51.100.1
```

```
Backup forwarder: 198.51.100.2
```

```
Advertised MAC label: 301904
```

```
Advertised aliasing label: 301904
```

```
Advertised split horizon label: 301840
```

```
ESI: 00:33:33:33:33:33:33:33:33
```

```
Status: Resolved by IFL ge-1/2/1.230
```

```
Local interface: ge-1/2/1.230, Status: Up/Forwarding
```

```
Number of remote PEs connected: 1
```

Remote PE	MAC label	Aliasing label	Mode
198.51.100.2	0	306128	all-active

Designated forwarder: 198.51.100.1
 Backup forwarder: 198.51.100.2
 Advertised MAC label: 301920
 Advertised aliasing label: 301920
 Advertised split horizon label: 301872
 ESI: 00:44:44:44:44:44:44:44:44:44
 Status: Resolved by NH 1048616
 Number of remote PEs connected: 1

Remote PE	MAC label	Aliasing label	Mode
198.51.100.4	0	305184	all-active

Instance: __default_evpn__

Route Distinguisher: 198.51.100.1:0

VLAN ID: None

Per-instance MAC route label: 301760

MAC database status	Local	Remote
Total MAC addresses:	0	0
Default gateway MAC addresses:	0	0

Number of local interfaces: 0 (0 up)

Number of IRB interfaces: 0 (0 up)

Number of bridge domains: 0

Number of neighbors: 2

198.51.100.2

Received routes

Ethernet auto-discovery:	0
--------------------------	---

Ethernet Segment:	3
-------------------	---

198.51.100.3

Received routes

Ethernet auto-discovery:	0
--------------------------	---

Ethernet Segment:	1
-------------------	---

Number of ethernet segments: 0

Instance: mhevpn

Route Distinguisher: 198.51.100.1:1

VLAN ID: 10

Per-instance MAC route label: 301728

MAC database status	Local	Remote
Total MAC addresses:	0	0
Default gateway MAC addresses:	1	0

Number of local interfaces: 4 (3 up)

Interface name	ESI	Mode	Status
----------------	-----	------	--------

```

ae0.0          00:11:11:11:11:11:11:11:11:11 all-active Up
ge-1/2/1.0     00:33:33:33:33:33:33:33:33:33 all-active Up
ge-1/2/2.0     00:22:22:22:22:22:22:22:22:22 all-active Up
ge-1/3/0.0     00:00:00:00:00:00:00:00:00:00 single-homed Down
Number of IRB interfaces: 1 (1 up)
  Interface name  VLAN ID  Status  L3 context
  irb.10         10      Up      vrf
Number of bridge domains: 1
  VLAN ID  Intfs / up  Mode          MAC sync  IM route label
  10       4 3      Extended     Enabled   302080
Number of neighbors: 3
198.51.100.2
  Received routes
    MAC address advertisement:      0
    MAC+IP address advertisement:   0
    Inclusive multicast:            1
    Ethernet auto-discovery:        6
198.51.100.3
  Received routes
    MAC address advertisement:      0
    MAC+IP address advertisement:   0
    Inclusive multicast:            1
    Ethernet auto-discovery:        2
198.51.100.4
  Received routes
    MAC address advertisement:      0
    MAC+IP address advertisement:   0
    Inclusive multicast:            1
    Ethernet auto-discovery:        2
Number of ethernet segments: 4
ESI: 00:11:11:11:11:11:11:11:11:11
Status: Resolved by IFL ae0.0
Local interface: ae0.0, Status: Up/Forwarding
Number of remote PEs connected: 2
  Remote PE      MAC label  Aliasing label  Mode
  198.51.100.3   0          305680          all-active
  198.51.100.2   0          306144          all-active
Designated forwarder: 198.51.100.2
Backup forwarder: 198.51.100.1
Backup forwarder: 198.51.100.3
Advertised MAC label: 301936
Advertised aliasing label: 301936
Advertised split horizon label: 301808

```

```

ESI: 00:22:22:22:22:22:22:22:22
Status: Resolved by IFL ge-1/2/2.0
Local interface: ge-1/2/2.0, Status: Up/Forwarding
Number of remote PEs connected: 1
  Remote PE      MAC label  Aliasing label  Mode
  198.51.100.2   0           306160          all-active
Designated forwarder: 198.51.100.1
Backup forwarder: 198.51.100.2
Advertised MAC label: 301952
Advertised aliasing label: 301952
Advertised split horizon label: 301840
ESI: 00:33:33:33:33:33:33:33:33
Status: Resolved by IFL ge-1/2/1.0
Local interface: ge-1/2/1.0, Status: Up/Forwarding
Number of remote PEs connected: 1
  Remote PE      MAC label  Aliasing label  Mode
  198.51.100.2   0           306176          all-active
Designated forwarder: 198.51.100.1
Backup forwarder: 198.51.100.2
Advertised MAC label: 301968
Advertised aliasing label: 301968
Advertised split horizon label: 301872
ESI: 00:44:44:44:44:44:44:44:44
Status: Resolved by NH 1048612
Number of remote PEs connected: 1
  Remote PE      MAC label  Aliasing label  Mode
  198.51.100.4   0           305200          all-active

```

Meaning

The output provides the following information:

- List of EVPN and virtual switch routing instances
- Mode of operation of each interface
- Neighbors of each routing instance
- Number of different routes received from each neighbor
- ESI attached to each routing instance
- Number of Ethernet segments on each routing instance

- DF election roles for each ESI in an EVI
- VLAN ID and MAC labels for each routing instance
- IRB interface details
- Number of default gateway MAC addresses received for the virtual switch routing instance (VS-1 and VS-2)

Verifying the Autodiscovery Routes per Ethernet Segment

Purpose

Verify that the autodiscovery routes per Ethernet segment are received.

Action

From operational mode, run the `show route table mhevpn.evpn.0` command.

Router PE1

```
user@PE1> show route table mhevpn.evpn.0
mhevpn.evpn.0: 17 destinations, 17 routes (17 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:198.51.100.1:1::111111111111111111::0/304
    *[EVPN/170] 00:11:37
    Indirect
1:198.51.100.1:1::222222222222222222::0/304
    *[EVPN/170] 00:11:37
    Indirect
1:198.51.100.1:1::333333333333333333::0/304
    *[EVPN/170] 00:11:37
    Indirect
1:198.51.100.2:0::111111111111111111::FFFF:FFFF/304
    *[BGP/170] 00:11:33, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:0::222222222222222222::FFFF:FFFF/304
    *[BGP/170] 00:11:33, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:0::333333333333333333::FFFF:FFFF/304
```

```

*[BGP/170] 00:11:33, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:1::11111111111111111111::0/304
*[BGP/170] 00:11:33, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:1::22222222222222222222::0/304
*[BGP/170] 00:11:33, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:1::33333333333333333333::0/304
*[BGP/170] 00:11:33, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.3:0::11111111111111111111::FFFF:FFFF/304
*[BGP/170] 00:11:37, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.6 via ae13.0, label-switched-path pe1tope3
1:198.51.100.3:1::11111111111111111111::0/304
*[BGP/170] 00:11:37, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.6 via ae13.0, label-switched-path pe1tope3
3:198.51.100.1:1::10::198.51.100.1/304
*[EVPN/170] 00:13:38
  Indirect
3:198.51.100.2:1::10::198.51.100.2/304
*[BGP/170] 00:11:33, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
3:198.51.100.3:1::10::198.51.100.3/304
*[BGP/170] 00:11:37, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.6 via ae13.0, label-switched-path pe1tope3

```

Router PE2

```

user@PE2> show route table mhevpn.evpn.0
mhevpn.evpn.0: 17 destinations, 17 routes (17 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:198.51.100.1:0::11111111111111111111::FFFF:FFFF/304

```

```

*[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:0::222222222222222222::FFFF:FFFF/304
*[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:0::33333333333333333333::FFFF:FFFF/304
*[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:1::11111111111111111111::0/304
*[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:1::22222222222222222222::0/304
*[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:1::33333333333333333333::0/304
*[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.2:1::11111111111111111111::0/304
*[EVPN/170] 01:10:26
  Indirect
1:198.51.100.2:1::22222222222222222222::0/304
*[EVPN/170] 01:10:26
  Indirect
1:198.51.100.2:1::33333333333333333333::0/304
*[EVPN/170] 01:10:26
  Indirect
1:198.51.100.3:0::11111111111111111111::FFFF:FFFF/304
*[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.7 via ge-0/0/4.0, label-switched-path pe2tope3
1:198.51.100.3:1::11111111111111111111::0/304
*[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified
  > to 198.51.100.7 via ge-0/0/4.0, label-switched-path pe2tope3
1:198.51.100.4:0::44444444444444444444::FFFF:FFFF/304
*[BGP/170] 01:10:17, localpref 100, from 203.0.113.0
  AS path: I, validation-state: unverified

```

```

> to 198.51.100.8 via ge-0/1/8.0, label-switched-path pe2tope4
1:198.51.100.4:1::44444444444444444444444444444444::0/304
    *[BGP/170] 01:10:17, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
> to 198.51.100.8 via ge-0/1/8.0, label-switched-path pe2tope4
3:198.51.100.1:1::10::198.51.100.1/304
    *[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
> to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
3:198.51.100.2:1::10::198.51.100.2/304
    *[EVPN/170] 01:12:14
    Indirect
3:198.51.100.3:1::10::198.51.100.3/304
    *[BGP/170] 01:10:26, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
> to 198.51.100.7 via ge-0/0/4.0, label-switched-path pe2tope3
3:198.51.100.4:1::10::198.51.100.4/304
    *[BGP/170] 01:10:17, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
> to 198.51.100.8 via ge-0/1/8.0, label-switched-path pe2tope4

```

Meaning

The remote type 1 autodiscovery route is received for the ESI attached to Router PE2, which is the other PE router connected to the multihomed CE device.

Verifying the Ethernet Segment Route

Purpose

Verify that the local and advertised autodiscovery routes per Ethernet segment and the Ethernet segment routes are received.

Action

From operational mode, run the `show route table __default_evpn__.evpn.0` command.

Router PE1

```

user@PE1> show route table __default_evpn__.evpn.0
__default_evpn__.evpn.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)

```



```

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

1:198.51.100.1:0::111111111111111111::FFFF:FFFF/304
    *[EVPN/170] 00:25:18
    Indirect
1:198.51.100.1:0::222222222222222222::FFFF:FFFF/304
    *[EVPN/170] 00:25:18
    Indirect
1:198.51.100.1:0::333333333333333333::FFFF:FFFF/304
    *[EVPN/170] 00:25:18
    Indirect
4:198.51.100.1:0::111111111111111111:198.51.100.1/304
    *[EVPN/170] 00:25:18
    Indirect
4:198.51.100.1:0::222222222222222222:198.51.100.1/304
    *[EVPN/170] 00:25:18
    Indirect
4:198.51.100.1:0::333333333333333333:198.51.100.1/304
    *[EVPN/170] 00:25:18
    Indirect
4:198.51.100.2:0::111111111111111111:198.51.100.2/304
    *[BGP/170] 00:25:14, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
4:198.51.100.2:0::222222222222222222:198.51.100.2/304
    *[BGP/170] 00:25:14, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
4:198.51.100.2:0::333333333333333333:198.51.100.2/304
    *[BGP/170] 00:25:14, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
4:198.51.100.3:0::111111111111111111:198.51.100.3/304
    *[BGP/170] 00:25:18, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ae13.0, label-switched-path pe1tope3

```

Router PE2

```

user@PE2> show route table __default_evpn__.evpn.0
__default_evpn__.evpn.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

1:198.51.100.2:0::11111111111111111111::FFFF:FFFF/304
    *[EVPN/170] 01:17:59
    Indirect
1:198.51.100.2:0::22222222222222222222::FFFF:FFFF/304
    *[EVPN/170] 01:17:59
    Indirect
1:198.51.100.2:0::33333333333333333333::FFFF:FFFF/304
    *[EVPN/170] 01:17:59
    Indirect
4:198.51.100.1:0::11111111111111111111:198.51.100.1/304
    *[BGP/170] 01:17:59, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
4:198.51.100.1:0::22222222222222222222:198.51.100.1/304
    *[BGP/170] 01:17:59, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
4:198.51.100.1:0::33333333333333333333:198.51.100.1/304
    *[BGP/170] 01:17:59, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
4:198.51.100.2:0::11111111111111111111:198.51.100.2/304
    *[EVPN/170] 01:17:59
    Indirect
4:198.51.100.2:0::22222222222222222222:198.51.100.2/304
    *[EVPN/170] 01:17:59
    Indirect
4:198.51.100.2:0::33333333333333333333:198.51.100.2/304
    *[EVPN/170] 01:17:59
    Indirect
4:198.51.100.3:0::11111111111111111111:198.51.100.3/304
    *[BGP/170] 01:17:59, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.7 via ge-0/0/4.0, label-switched-path pe2tope3

```

Meaning

The output displays the local and remote type 1 (autodiscovery) and type 4 (Ethernet segment) routes.

Verifying the DF Status

Purpose

Confirm which PE router is the designated forwarder (DF) for each routing instance.

Action

From operational mode, run the `show evpn instance designated-forwarder` command.

```

user@PE1> show evpn instance designated forwarder
Instance: VS-1
  Number of ethernet segments: 4
    ESI: 00:11:11:11:11:11:11:11:11
      Designated forwarder: 198.51.100.3
    ESI: 00:22:22:22:22:22:22:22:22
      Designated forwarder: 198.51.100.1
    ESI: 00:33:33:33:33:33:33:33:33
      Designated forwarder: 198.51.100.1
    ESI: 00:44:44:44:44:44:44:44:44
      Designated forwarder: No local attachment to ethernet segment

Instance: VS-2
  Number of ethernet segments: 4
    ESI: 00:11:11:11:11:11:11:11:11
      Designated forwarder: 198.51.100.1
    ESI: 00:22:22:22:22:22:22:22:22
      Designated forwarder: 198.51.100.1
    ESI: 00:33:33:33:33:33:33:33:33
      Designated forwarder: 198.51.100.1
    ESI: 00:44:44:44:44:44:44:44:44
      Designated forwarder: No local attachment to ethernet segment

Instance: mhevpn
  Number of ethernet segments: 4
    ESI: 00:11:11:11:11:11:11:11:11
      Designated forwarder: 198.51.100.2
    ESI: 00:22:22:22:22:22:22:22:22
      Designated forwarder: 198.51.100.1
    ESI: 00:33:33:33:33:33:33:33:33
      Designated forwarder: 198.51.100.1

```

```
ESI: 00:44:44:44:44:44:44:44:44:44
Designated forwarder: No local attachment to ethernet segment
```

Meaning

The designated forwarder is displayed for each routing instance and ESI.

Verifying the BDF Status

Purpose

Confirm which PE router is the backup designated forwarder (BDF) for each routing instance.

Action

From operational mode, run the `show evpn instance backup-forwarder` command.

```
user@PE1> show evpn instance backup-forwarder
Instance: VS-1
  Number of ethernet segments: 4
    ESI: 00:11:11:11:11:11:11:11:11:11
      Backup forwarder: 198.51.100.1
      Backup forwarder: 198.51.100.2
    ESI: 00:22:22:22:22:22:22:22:22:22
      Backup forwarder: 198.51.100.2
    ESI: 00:33:33:33:33:33:33:33:33:33
      Backup forwarder: 198.51.100.2
    ESI: 00:44:44:44:44:44:44:44:44:44
      Backup forwarder: No local attachment to ethernet segment

Instance: VS-2
  Number of ethernet segments: 4
    ESI: 00:11:11:11:11:11:11:11:11:11
      Backup forwarder: 198.51.100.2
      Backup forwarder: 198.51.100.3
    ESI: 00:22:22:22:22:22:22:22:22:22
      Backup forwarder: 198.51.100.2
    ESI: 00:33:33:33:33:33:33:33:33:33
      Backup forwarder: 198.51.100.2
    ESI: 00:44:44:44:44:44:44:44:44:44
      Backup forwarder: No local attachment to ethernet segment
```

```

Instance: mhevpn
Number of ethernet segments: 4
  ESI: 00:11:11:11:11:11:11:11:11:11
    Backup forwarder: 198.51.100.1
    Backup forwarder: 198.51.100.3
  ESI: 00:22:22:22:22:22:22:22:22:22
    Backup forwarder: 198.51.100.2
  ESI: 00:33:33:33:33:33:33:33:33:33
    Backup forwarder: 198.51.100.2
  ESI: 00:44:44:44:44:44:44:44:44:44
    Backup forwarder: No local attachment to ethernet segment

```

Meaning

The backup designated forwarder is displayed for each routing instance and ESI.

Verifying the Remote IRB and Host IP

Purpose

Verify that the remote IRB IP and the host IP are received.

Action

Router PE1

From operational mode, run the `show route table mhevpn` command.

```

user@PE1> show route table mhevpn
mhevpn.evpn.0: 17 destinations, 17 routes (17 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:198.51.100.1:1::11111111111111111111::0/304
    *[EVPN/170] 01:34:26
    Indirect
1:198.51.100.1:1::22222222222222222222::0/304
    *[EVPN/170] 01:34:26
    Indirect
1:198.51.100.1:1::33333333333333333333::0/304
    *[EVPN/170] 01:34:26

```

```

Indirect
1:198.51.100.2:0::111111111111111111::FFFF:FFFF/304
    *[BGP/170] 01:34:22, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:0::222222222222222222::FFFF:FFFF/304
    *[BGP/170] 01:34:22, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:0::333333333333333333::FFFF:FFFF/304
    *[BGP/170] 01:34:22, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:1::111111111111111111::0/304
    *[BGP/170] 01:34:22, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:1::222222222222222222::0/304
    *[BGP/170] 01:34:22, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.2:1::333333333333333333::0/304
    *[BGP/170] 01:34:22, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
1:198.51.100.3:0::111111111111111111::FFFF:FFFF/304
    *[BGP/170] 01:34:26, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ae13.0, label-switched-path pe1tope3
1:198.51.100.3:1::111111111111111111::0/304
    *[BGP/170] 01:34:26, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.6 via ae13.0, label-switched-path pe1tope3
3:198.51.100.1:1::10::198.51.100.1/304
    *[EVPN/170] 01:36:27
    Indirect
3:198.51.100.2:1::10::198.51.100.2/304
    *[BGP/170] 01:34:22, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.5 via ae12.0, label-switched-path pe1tope2
3:198.51.100.3:1::10::198.51.100.3/304
    *[BGP/170] 01:34:26, localpref 100, from 203.0.113.0

```

```
AS path: I, validation-state: unverified
> to 198.51.100.6 via ae13.0, label-switched-path pe1tope3
```

Router PE2

From operational mode, run the `show route table mhevpn` command.

```
user@PE2> show route table mhevpn
mhevpn.evpn.0: 17 destinations, 17 routes (17 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:198.51.100.1:0::11111111111111111111::FFFF:FFFF/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:0::22222222222222222222::FFFF:FFFF/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:0::33333333333333333333::FFFF:FFFF/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:1::11111111111111111111::0/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:1::22222222222222222222::0/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.1:1::33333333333333333333::0/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
1:198.51.100.2:1::11111111111111111111::0/304
    *[EVPN/170] 01:35:11
    Indirect
1:198.51.100.2:1::22222222222222222222::0/304
    *[EVPN/170] 01:35:11
    Indirect
1:198.51.100.2:1::33333333333333333333::0/304
```

```

*[EVPN/170] 01:35:11
    Indirect
1:198.51.100.3:0::111111111111111111::FFFF:FFFF/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.7 via ge-0/0/4.0, label-switched-path pe2tope3
1:198.51.100.3:1::111111111111111111::0/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.7 via ge-0/0/4.0, label-switched-path pe2tope3
1:198.51.100.4:0::444444444444444444::FFFF:FFFF/304
    *[BGP/170] 01:35:02, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.8 via ge-0/1/8.0, label-switched-path pe2tope4
1:198.51.100.4:1::444444444444444444::0/304
    *[BGP/170] 01:35:02, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.8 via ge-0/1/8.0, label-switched-path pe2tope4
3:198.51.100.1:1::10::198.51.100.1/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.12 via ae12.0, label-switched-path pe2tope1
3:198.51.100.2:1::10::198.51.100.2/304
    *[EVPN/170] 01:36:59
        Indirect
3:198.51.100.3:1::10::198.51.100.3/304
    *[BGP/170] 01:35:11, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.7 via ge-0/0/4.0, label-switched-path pe2tope3
3:198.51.100.4:1::10::198.51.100.4/304
    *[BGP/170] 01:35:02, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.8 via ge-0/1/8.0, label-switched-path pe2tope4

```

Router PE3

From operational mode, run the `show route table mhevpn` command.

```

user@PE3> show route table mhevpn
mhevpn.evpn.0: 19 destinations, 19 routes (19 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```



```

1:198.51.100.1:0::11111111111111111111::FFFF:FFFF/304
    *[BGP/170] 01:36:10, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.13 via ae13.0, label-switched-path pe3tope1
1:198.51.100.1:0::22222222222222222222::FFFF:FFFF/304
    *[BGP/170] 01:36:10, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.13 via ae13.0, label-switched-path pe3tope1
1:198.51.100.1:0::33333333333333333333::FFFF:FFFF/304
    *[BGP/170] 01:36:10, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.13 via ae13.0, label-switched-path pe3tope1
1:198.51.100.1:1::11111111111111111111::0/304
    *[BGP/170] 01:36:10, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.13 via ae13.0, label-switched-path pe3tope1
1:198.51.100.1:1::22222222222222222222::0/304
    *[BGP/170] 01:36:10, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.13 via ae13.0, label-switched-path pe3tope1
1:198.51.100.1:1::33333333333333333333::0/304
    *[BGP/170] 01:36:10, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.13 via ae13.0, label-switched-path pe3tope1
1:198.51.100.2:0::11111111111111111111::FFFF:FFFF/304
    *[BGP/170] 01:36:06, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.10 via ge-2/0/3.0, label-switched-path pe3tope2
1:198.51.100.2:0::22222222222222222222::FFFF:FFFF/304
    *[BGP/170] 01:36:06, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.10 via ge-2/0/3.0, label-switched-path pe3tope2
1:198.51.100.2:0::33333333333333333333::FFFF:FFFF/304
    *[BGP/170] 01:36:06, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.10 via ge-2/0/3.0, label-switched-path pe3tope2
1:198.51.100.2:1::11111111111111111111::0/304
    *[BGP/170] 01:36:06, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified
    > to 198.51.100.10 via ge-2/0/3.0, label-switched-path pe3tope2
1:198.51.100.2:1::22222222222222222222::0/304
    *[BGP/170] 01:36:06, localpref 100, from 203.0.113.0
    AS path: I, validation-state: unverified

```

```

        > to 198.51.100.10 via ge-2/0/3.0, label-switched-path pe3tope2
1:198.51.100.2:1::33333333333333333333333333333333::0/304
    *[BGP/170] 01:36:06, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.10 via ge-2/0/3.0, label-switched-path pe3tope2
1:198.51.100.3:1::11111111111111111111111111111111::0/304
    *[EVPN/170] 01:36:25
        Indirect
1:198.51.100.4:0::44444444444444444444444444444444::FFFF:FFFF/304
    *[BGP/170] 01:35:58, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.11 via ge-2/1/3.0, label-switched-path pe3tope4
1:198.51.100.4:1::44444444444444444444444444444444::0/304
    *[BGP/170] 01:35:58, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.11 via ge-2/1/3.0, label-switched-path pe3tope4
3:198.51.100.1:1::10::198.51.100.1/304
    *[BGP/170] 01:36:10, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.13 via ae13.0, label-switched-path pe3tope1
3:198.51.100.2:1::10::198.51.100.2/304
    *[BGP/170] 01:36:06, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.10 via ge-2/0/3.0, label-switched-path pe3tope2
3:198.51.100.3:1::10::198.51.100.3/304
    *[EVPN/170] 01:37:33
        Indirect
3:198.51.100.4:1::10::198.51.100.4/304
    *[BGP/170] 01:35:58, localpref 100, from 203.0.113.0
        AS path: I, validation-state: unverified
        > to 198.51.100.11 via ge-2/1/3.0, label-switched-path pe3tope4

```

Meaning

The output displays the local and remote IRB interfaces. It also displays the local and remote hosts that are installed in the VRF table.

Example: Configuring LACP for EVPN Active-Active Multihoming

IN THIS SECTION

- [Requirements | 442](#)
- [Overview | 443](#)
- [Configuration | 446](#)
- [Verification | 460](#)

This example shows how to configure the Link Aggregation Control Protocol (LACP) on multihomed customer edge (CE) and provider edge (PE) devices in an Ethernet VPN (EVPN) active-active multihomed network.

Requirements

This example uses the following hardware and software components:

- Four MX Series 5G Universal Routing Platforms with MPC interfaces only, of which:
 - Two routers are configured as PE devices connected to a common multihomed customer site.
 - One router is configured as a remote PE device connected to a single-homed customer site.
 - One router is configured as the core provider device.
- Two CE devices, of which:
 - One CE device is multihomed to the two PE devices.
 - One CE device is single-homed to the remote PE device.
- Junos OS Release 17.1 or later running on all the PE devices.

Before you begin:

1. Configure the device interfaces and enable MPLS on all the interfaces of the PE devices.
2. Configure OSPF or any other IGP between the core provider device and the PE devices.
3. Configure MPLS and a label-switched path (LSP) from the ingress PE device to the remote egress PE device.
4. Configure an internal BGP session between the PE devices.



NOTE: We recommend that you configure Bidirectional Forwarding Detection (BFD) on the BGP session for faster detection of core isolation and better convergence.

Overview

IN THIS SECTION

● [Topology](#) | 444

An extra level of redundancy can be achieved in an EVPN active-active multihoming environment by configuring LACP on both the endpoints of the multihomed CE-PE link. The multihomed devices are configured with aggregated trunk links, where the link aggregation group (LAG) interfaces of the CE-PE link can either be in the active or in the standby state. When the LAG interface is in the active state, data traffic is transmitted over the CE-PE link. When the LAG interface is in the standby state, data traffic is blocked and only control traffic for communicating LAG interface state is transmitted over the link.

The LAG interface state is monitored and operated by LACP to ensure fast convergence on isolation of a multihomed PE device from the core. When there is a core failure, a null route can occur at the isolated PE device. However, with the support for LACP on the CE-PE link, at the time of core isolation, the CE-facing interface of the multihomed PE device is set to the standby state, thereby blocking data traffic transmission from and toward the multihomed CE device. After the core recovers from the failure, the interface state is switched back from standby to active.

The support for LACP for EVPN active-active multihoming is applicable to both EVPN with MPLS and EVPN with VXLAN.

When LACP is configured on the CE-PE link, isolation of the multihomed PE device from the core is handled as follows:

1. When the EVPN BGP peers are null for a multihomed PE device, the PE device is isolated from the core. The CE devices connected to the isolated PE device are notified about the core failure by the isolated PE device.
2. On learning about the core failure, data traffic is not forwarded from the CE device to the isolated multihomed PE device. Instead, the traffic is diverted to the other multihomed PE devices that belong to the same LAG. This helps in preventing traffic black holes at the isolated PE device.
3. If the multihomed CE device uses the LAG for load balancing traffic to multiple active multihomed PE devices, then the LACP configuration along with the same system ID configured on all the

multihomed PE devices for that given LAG, triggers an LACP out-of-sync to all the attached multihomed CE links.

When configuring LACP on the multihomed devices, be aware of the following considerations:

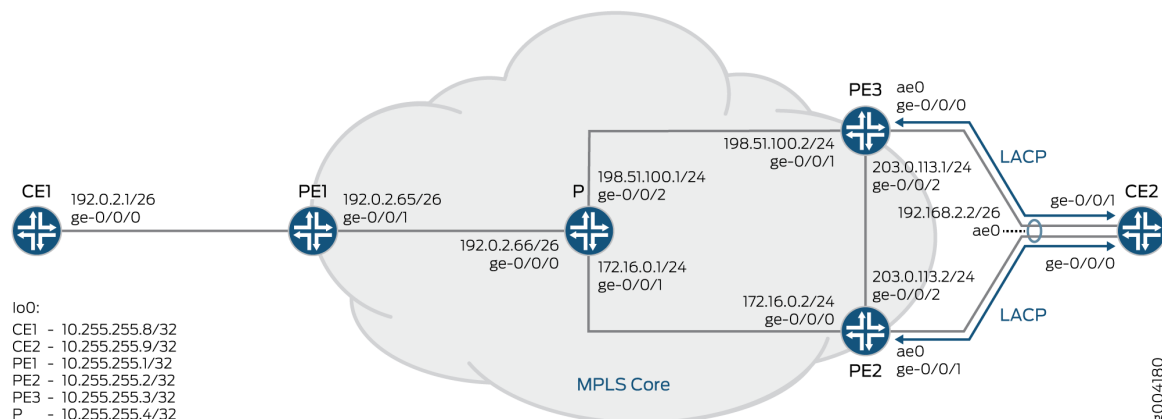
- The LAG link can operate either in the active or in the standby state regardless of the UP/DOWN operational state.
- On system reboot, the LACP link on the multihomed PE device is in the active state.
- When the control plane goes down or when the BGP-EVPN session is down, LACP is notified to run multiplexer state machine for the aggregation port and move the link from active to standby.
- An interface is not treated as up unless it operates in the active state and its operational state is also up.
- The following features are not supported with LACP on EVPN active-active multihoming:
 - Ethernet segment identifier (ESI) value auto-derivation from LACP system ID.
 - Redundancy coverage for non-LAG interfaces.
 - Redundancy coverage for core isolation with static LAGs.
 - Node isolation.
 - Access failures for LACP with EVPN active-active multihoming.

Topology

[Figure 30 on page 445](#) illustrates an EVPN active-active multihoming network with LACP configured on the multihomed CE and PE devices. Device CE1 is single-homed and is connected to a remote PE device, PE1. Device PE2 and Device PE3 connect to a common multihomed CE—CE2. Device P is the core device to which all the PE devices (PE1, PE2, and PE3) are connected.

The endpoints of the multihomed devices—Device CE2, Device PE2, and Device PE3—are configured with LACP on the CE-PE link.

Figure 30: LACP Support in EVPN Active-Active Multihoming



Core isolation of Device PE3 is handled as follows:

1. After LACP is configured on the multihomed CE-PE links, the LACP configuration and operational data is synchronized between the LACP peers to operate as a LAG.

The synchronization between the LACP peers happens with the exchange of control PDUs, and is required for the following reasons:

- To determine the state of the links in the Ethernet bundle—all-active or standby.
 - To detect and handle CE device misconfiguration when LACP Port Key is configured on the PE device.
 - To detect and handle miswiring between CE and PE devices when LACP Port Key is configured on the PE device.
 - To detect and react to actor or partner churn when the LACP speakers are not able to converge.
2. After Device PE2 and Device PE3 establish BGP session with at least one peer, the state of the CE-PE link is set to unblocking mode by LACP.
 3. When there is a core failure, and the BGP EVPN peers of Device PE2 becomes null, Device PE2 is isolated from the core. This can cause a null route at Device PE2. To prevent this situation, the LAG interface of Device PE2 that is facing Device CE2 is changed from the active state to the standby state by LACP.
 4. An out-of-sync notification is sent from LACP on the attached multihomed CE2 link to block traffic transmission between Device CE2 and Device PE2.
 5. When the control plane recovers, that is when Device PE2 establishes BGP session with other EVPN PEs, the LAG interface of Device PE2 is switched back from standby to active by LACP.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 446](#)
- [Procedure | 451](#)
- [Procedure | 457](#)

CLI Quick Configuration

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

Device CE1

```
set interfaces ge-0/0/0 vlan-tagging
set interfaces ge-0/0/0 mac 00:00:00:00:00:01
set interfaces ge-0/0/0 unit 0 vlan-id 100
set interfaces ge-0/0/0 unit 0 family inet address 192.0.2.1/26
set interfaces lo0 unit 0 family inet address 10.255.255.8/32
```

Device CE2

```
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ge-0/0/1 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 mac 00:00:00:00:00:03
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 unit 0 vlan-id 100
set interfaces ae0 unit 0 family inet address 192.0.2.2/26
set interfaces lo0 unit 0 family inet address 10.255.255.9/32
```

Device PE1

```

set interfaces ge-0/0/0 vlan-tagging
set interfaces ge-0/0/0 encapsulation flexible-ethernet-services
set interfaces ge-0/0/0 unit 0 encapsulation vlan-bridge
set interfaces ge-0/0/0 unit 0 vlan-id 100
set interfaces ge-0/0/1 unit 0 family inet address 192.0.2.65/26
set interfaces ge-0/0/1 unit 0 family iso
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.255.1/32
set routing-options router-id 10.255.255.1
set routing-options autonomous-system 65100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls interface ge-0/0/1.0
set protocols bgp local-address 10.255.255.1
set protocols bgp family inet unicast
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.255.1
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 10.255.255.2
set protocols bgp group ibgp neighbor 10.255.255.3
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 ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ldp interface ge-0/0/1.0
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances ALPHA instance-type evpn
set routing-instances ALPHA vlan-id 100
set routing-instances ALPHA interface ge-0/0/0.0
set routing-instances ALPHA route-distinguisher 10.255.255.1:100
set routing-instances ALPHA vrf-target target:65100:100
set routing-instances ALPHA protocols evpn label-allocation per-instance
set policy-options policy-statement load_balance then load-balance per-packet
set routing-options forwarding-table export load_balance

```


Device PE2

```

set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 unit 0 family inet address 172.16.0.2/24
set interfaces ge-0/0/0 unit 0 family iso
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 gigether-options 802.3ad ae0
set interfaces ge-0/0/2 unit 0 family inet address 203.0.113.2/24
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ae0 vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:01:02:03:04:05
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 100
set interfaces lo0 unit 0 family inet address 10.255.255.2/32
set routing-options router-id 10.255.255.2
set routing-options autonomous-system 65100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp local-address 10.255.255.2
set protocols bgp family inet unicast
set protocols bgp family l2vpn signaling
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.255.2
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 10.255.255.3
set protocols bgp group ibgp neighbor 10.255.255.1
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 ALPHA instance-type evpn
set routing-instances ALPHA vlan-id 100
set routing-instances ALPHA interface ae0.0

```

```

set routing-instances ALPHA route-distinguisher 10.255.255.2:100
set routing-instances ALPHA vrf-target target:65100:100
set policy-options policy-statement load_balance then load_balance per-packet
set routing-options forwarding-table export load_balance

```

Device PE3

```

set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 gigether-options 802.3ad ae0
set interfaces ge-0/0/1 unit 0 family inet address 198.51.100.2/24
set interfaces ge-0/0/1 unit 0 family iso
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces ge-0/0/2 unit 0 family inet address 203.0.113.1/24
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ae0 vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:01:02:03:04:05
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 100
set interfaces lo0 unit 0 family inet address 10.255.255.3/32
set routing-options router-id 10.255.255.3
set routing-options autonomous-system 65100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp local-address 10.255.255.3
set protocols bgp family inet unicast
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.255.3
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 10.255.255.2
set protocols bgp group ibgp neighbor 10.255.255.1
set protocols bgp group l2vpn type internal
set protocols bgp group l2vpn family l2vpn signaling
set protocols bgp group l2vpn neighbor 10.255.255.4
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 ALPHA instance-type evpn
set routing-instances ALPHA vlan-id 100
set routing-instances ALPHA interface ae0.0
set routing-instances ALPHA route-distinguisher 10.255.255.3:100
set routing-instances ALPHA vrf-target target:65100:100
set policy-options policy-statement load_balance then load_balance per-packet
set routing-options forwarding-table export load_balance

```

Device P

```

set interfaces ge-0/0/0 unit 0 family inet address 192.0.2.66/26
set interfaces ge-0/0/0 unit 0 family iso
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 unit 0 family inet address 172.16.0.1/24
set interfaces ge-0/0/1 unit 0 family iso
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces ge-0/0/2 unit 0 family inet address 198.51.100.1/24
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.255.4/32
set routing-options router-id 10.255.255.4
set routing-options autonomous-system 100
set protocols rsvp interface all aggregate
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface ge-0/0/1.0
set protocols mpls interface ge-0/0/2.0
set protocols bgp group l2vpn type internal
set protocols bgp group l2vpn local-address 10.255.255.4
set protocols bgp group l2vpn family l2vpn signaling
set protocols bgp group l2vpn neighbor 10.255.255.3
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
set protocols ldp interface all
set routing-instances vpls1 instance-type vpls
set routing-instances vpls1 route-distinguisher 10.255.255.4:100

```

```

set routing-instances vpls1 vrf-target target:200:200
set routing-instances vpls1 protocols vpls no-tunnel-services
set routing-instances vpls1 protocols vpls site vpls-pe site-identifier 3
set routing-instances vpls1 protocols vpls site vpls-pe best-site
set policy-options policy-statement load_balance then load-balance per-packet
set routing-options forwarding-table export load_balance

```

Procedure

Step-by-Step Procedure

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

To configure Device PE2:



NOTE: Repeat this procedure for other multihomed PE devices after modifying the appropriate interface names, addresses, and other parameters.

1. Specify the number of aggregated Ethernet interfaces to be created on Device PE2.

```

[edit chassis]
user@PE2# set aggregated-devices ethernet device-count 1

```

2. Configure Device PE2 interfaces that connect to Device P and Device PE3, respectively.

```

[edit interfaces]
user@PE2# set ge-0/0/0 unit 0 family inet address 172.16.0.2/24
user@PE2# set ge-0/0/0 unit 0 family iso
user@PE2# set ge-0/0/0 unit 0 family mpls
user@PE2# set ge-0/0/2 unit 0 family inet address 203.0.113.2/24
user@PE2# set ge-0/0/2 unit 0 family iso
user@PE2# set ge-0/0/2 unit 0 family mpls

```

3. Configure Device PE2 interfaces within the ae0 aggregated bundle toward the multihomed Device CE2.

```
[edit interfaces]
user@PE2# set ge-0/0/1 gigether-options 802.3ad ae0
user@PE2# set ae0 vlan-tagging
user@PE2# set ae0 encapsulation flexible-ethernet-services
user@PE2# set ae0 unit 0 encapsulation vlan-bridge
user@PE2# set ae0 unit 0 vlan-id 100
```

4. Configure active-active multihoming on the aggregated Ethernet interface.

```
[edit interfaces]
user@PE2# set ae0 esi 00:11:22:33:44:55:66:77:88:99
user@PE2# set ae0 esi all-active
```

5. Configure LACP on the CE-PE link of Device PE2.

```
[edit interfaces]
user@PE2# set ae0 aggregated-ether-options lacp active
user@PE2# set ae0 aggregated-ether-options lacp system-id 00:01:02:03:04:05
```

6. Configure the loopback interface of Device PE2.

```
[edit interfaces]
user@PE2# set lo0 unit 0 family inet address 10.255.255.2/32
```

7. Configure the router ID and autonomous system number for Device PE2.

```
[edit routing-options]
user@PE2# set router-id 10.255.255.2
user@PE2# set autonomous-system 65100
```

8. Enable chained composite next hop for EVPN on Device PE2.

```
[edit routing-options]
user@PE2# set forwarding-table chained-composite-next-hop ingress evpn
```

9. Configure MPLS on all the interfaces of Device PE2, excluding the management interface.

```
[edit protocols]
user@PE2# set mpls interface all
user@PE2# set mpls interface fxp0.0 disable
```

10. Configure an internal BGP group for Device PE1 to peer with the other EVPN PE devices.

```
[edit protocols]
user@PE2# set bgp local-address 10.255.255.2
user@PE2# set bgp family inet unicast
user@PE2# set bgp family l2vpn signaling
user@PE2# set bgp group ibgp type internal
user@PE2# set bgp group ibgp local-address 10.255.255.2
user@PE2# set bgp group ibgp family evpn signaling
user@PE2# set bgp group ibgp neighbor 10.255.255.3
user@PE2# set bgp group ibgp neighbor 10.255.255.1
```

11. Configure OSPF and LDP on all the interfaces of Device PE2, excluding the management interface.

```
[edit protocols]
user@PE2# set ospf area 0.0.0.0 interface all
user@PE2# set ospf area 0.0.0.0 interface fxp0.0 disable
user@PE2# set ospf area 0.0.0.0 interface lo0.0 passive
user@PE2# set ldp interface all
user@PE2# set ldp interface fxp0.0 disable
user@PE2# set ldp interface lo0.0
```

12. Configure an EVPN routing instance on Device PE2 and assign the routing instance parameters.

```
[edit routing-instances]
user@PE2# set ALPHA instance-type evpn
user@PE2# set ALPHA vlan-id 100
```

```

user@PE2# set ALPHA interface ae0.0
user@PE2# set ALPHA route-distinguisher 10.255.255.2:100
user@PE2# set ALPHA vrf-target target:65100:100

```

13. Create and apply an ECMP load balancing policy.

```

[edit]
user@PE2# set policy-options policy-statement load_balance then load-balance per-packet
user@PE2# set routing-options forwarding-table export load_balance

```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, 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 chassis
aggregated-devices {
  ethernet {
    device-count 1;
  }
}

```

```

user@PE2# show interfaces
ge-0/0/0 {
  unit 0 {
    family inet {
      address 172.16.0.2/24;
    }
    family iso;
    family mpls;
  }
}
ge-0/0/1 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/0/2 {

```

```

    unit 0 {
        family inet {
            address 203.0.113.2/24;
        }
        family iso;
        family mpls;
    }
}
ae0 {
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
        00:11:22:33:44:55:66:77:88:99;
        all-active;
    }
    aggregated-ether-options {
        lacp {
            active;
            system-id 00:01:02:03:04:05;
        }
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 100;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.255.2/32;
        }
    }
}
}

```

```

user@PE2# show routing-options
router-id 10.255.255.2;
autonomous-system 65100;
forwarding-table {
    export load_balance;
    chained-composite-next-hop {
        ingress {

```



```

        evpn;
    }
}
}

```

```
user@PE2# show protocols
```

```

mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    local-address 10.255.255.2;
    family inet {
        unicast;
    }
    family l2vpn {
        signaling;
    }
    group ibgp {
        type internal;
        local-address 10.255.255.2;
        family evpn {
            signaling;
        }
        neighbor 10.255.255.3;
        neighbor 10.255.255.1;
    }
}
ospf {
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
        interface lo0.0 {
            passive;
        }
    }
}
}

```

```

ldp {
  interface all;
  interface fxp0.0 {
    disable;
  }
  interface lo0.0;
}

```

```

user@PE2# show routing-instances
ALPHA {
  instance-type evpn;
  vlan-id 100;
  interface ae0.0;
  route-distinguisher 10.255.255.2:100;
  vrf-target target:65100:100;
  protocols {
    evpn {
      traceoptions {
        file evpn-alpha.txt size 4294967295;
        flag all;
      }
    }
  }
}

```

```

user@PE2# show policy-options
then {
  load-balance per-packet;
}
}

```

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

Procedure

Step-by-Step Procedure

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

To configure Device CE2:



NOTE: Repeat this procedure for other multihomed CE devices after modifying the appropriate interface names, addresses, and other parameters.

1. Specify the number of aggregated Ethernet interfaces to be created on Device CE2.

```
[edit chassis]
user@CE2# set aggregated-devices ethernet device-count 1
```

2. Configure Device CE2 interfaces within the ae0 aggregated bundle toward Device PE2 and Device PE3.

```
[edit interfaces]
user@CE2# set ge-0/0/0 gigether-options 802.3ad ae0
user@CE2# set ge-0/0/1 gigether-options 802.3ad ae0
user@CE2# set ae0 flexible-vlan-tagging
user@CE2# set ae0 encapsulation flexible-ethernet-services
user@E2# set ae0 mac 00:00:00:00:00:03
user@E2# set ae0 unit 0 vlan-id 100
user@CE2# set ae0 unit 0 family inet address 192.0.2.2/26
```

3. Configure LACP on the CE-PE links on Device CE2.

```
[edit interfaces]
user@CE2# set ae0 aggregated-ether-options lacp active
```

4. Configure the loopback interface of Device CE2.

```
[edit interfaces]
user@CE2# set interfaces lo0 unit 0 family inet address 10.255.255.9/32
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, 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@CE2# show chassis
aggregated-devices {
  ethernet {
    device-count 1;
  }
}
```

```
user@CE2# show interfaces
ge-0/0/0 {
  gigether-options {
    802.3ad ae0;
  }
}
ge-0/0/1 {
  gigether-options {
    802.3ad ae0;
  }
}
ae0 {
  flexible-vlan-tagging;
  encapsulation flexible-ethernet-services;
  mac 00:00:00:00:00:03;
  aggregated-ether-options {
    lacp {
      active;
    }
  }
  unit 0 {
    vlan-id 100;
    family inet {
      address 192.0.2.2/26;
    }
  }
}
lo0 {
```

```
unit 0 {
    family inet {
        address 10.255.255.9/32;
    }
}
```

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

Verification

IN THIS SECTION

Verifying BGP Session Status | 460

Verifying LACP Interface Status of Multihomed PE Device | 461

Verifying LACP Interface State of Isolated PE Device | 462

Verifying EVPN Routing Instance | 463

Confirm that the configuration is working properly.

Verifying BGP Session Status

Purpose

Verify that Device PE2 has established BGP peering with other EVPN PE devices.

Action

From operational mode, run the `show bgp summary` command.

```
user@PE2> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State   Pending
inet.0
                0          0          0          0          0          0
bgp.l2vpn.0
                0          0          0          0          0          0
bgp.evpn.0
```

```

Peer           5      5      0      0      0      0
                AS    InPkt  OutPkt  OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.255.255.1    65100    6316    6333    0      0 1d 23:33:03 Establ
  bgp.evpn.0: 1/1/1/0
  ALPHA.evpn.0: 1/1/1/0
  __default_evpn__.evpn.0: 0/0/0/0
10.255.255.3    65100    6318    6332    0      0 1d 23:32:56 Establ
  bgp.evpn.0: 4/4/4/0
  ALPHA.evpn.0: 3/3/3/0
  __default_evpn__.evpn.0: 1/1/1/0

```

Meaning

Device PE2 has established BGP peering with the other EVPN PE devices—Device PE1 and Device PE3.

Verifying LACP Interface Status of Multihomed PE Device

Purpose

Verify the LACP interface state on Device PE2.

Action

From operational mode, run the **show lacp interfaces** and **show interfaces ae* terse** commands.

```

user@PE2> show lacp interfaces
Aggregated interface: ae0
  LACP state:      Role  Exp  Def  Dist  Col  Syn  Aggr  Timeout  Activity
    ge-0/0/1      Actor  No   No   Yes  Yes  Yes  Yes     Fast    Active
    ge-0/0/1      Partner No   No   Yes  Yes  Yes  Yes     Fast    Active
LACP protocol:      Receive State  Transmit State      Mux State
  ge-0/0/1          Current  Fast periodic Collecting distributing

```

```

user@PE2> show interfaces ae* terse
Interface      Admin Link Proto  Local      Remote
ae0            up    up

```

```

ae0.0          up    up    bridge
ae0.32767      up    up    multiservice

```

Meaning

The LACP LAG interface state is active when there is BGP peering with other EVPN PE devices. The physical interface state of the aggregated Ethernet interfaces are up and operational.

Verifying LACP Interface State of Isolated PE Device

Purpose

Verify the LACP interface state of Device PE2 when no BGP EVPN peers have been established.

Action

From operational mode, run the **show lacp interfaces** and **show interfaces ae* terse** commands.

```

user@PE2> show lacp interfaces
Aggregated interface: ae0
  LACP state:      Role   Exp   Def   Dist  Col   Syn   Aggr  Timeout  Activity
    ge-0/0/1      Actor  No    No    No    No    No    Yes    Fast    Active
    ge-0/0/1      Partner No    No    No    No    Yes   Yes    Fast    Active
  LACP protocol:   Receive State Transmit State      Mux State
    ge-0/0/1              Current  Fast periodic Waiting

```

```

user@PE2> show interfaces ae* terse
Interface      Admin Link Proto  Local      Remote
ae0            up    down
ae0.0          up    down  bridge
ae0.32767      up    down  multiservice

```

Meaning

Because of core isolation, the LAG interface state is in standby, blocking traffic to-and-from Device CE2. As a result, the physical interface state of the aggregated Ethernet interface is also down until the LACP link state switches back to active on core failure recovery.

Verifying EVPN Routing Instance

Purpose

Verify the EVPN routing instance parameters on Device PE2.

Action

From operational mode, run the **show evpn instance extensive** command.

```

user@PE2> show evpn instance extensive
Instance: ALPHA
Route Distinguisher: 10.255.255.2:100
VLAN ID: 100
Per-instance MAC route label: 299776
MAC database status
MAC advertisements:
MAC+IP advertisements:
Default gateway MAC advertisements:
Number of local interfaces: 1 (1 up)
Interface name  ESI
ae0.0          00:11:22:33:44:55:66:77:88:99
Mode
all-active
Status
Up
Number of IRB interfaces: 0 (0 up)
Number of bridge domains: 1
VLAN  Domain ID  Intfs / up  IRB intf  Mode
100    1          1 1          Extended
MAC sync  IM route label
Enabled  300048
Number of neighbors: 2
Address      MAC    MAC+IP    AD    IM    ES
10.255.255.1  0      0         0     1     0
10.255.255.3  0      0         2     1     0
Number of ethernet segments: 1
ESI: 00:11:22:33:44:55:66:77:88:99
Status: Resolved by IFL ae0.0
Local interface: ae0.0, Status: Up/Forwarding
Number of remote PEs connected: 1
Remote PE    MAC label  Aliasing label  Mode
10.255.255.3  0          299856          all-active
Designated forwarder: 10.255.255.2
Backup forwarder: 10.255.255.3
Last designated forwarder update: Nov 21 01:27:52
Advertised MAC label: 299856
Advertised aliasing label: 299856

```



```

    Advertised split horizon label: 299968

Instance: __default_evpn__
  Route Distinguisher: 10.255.255.2:0
  Number of bridge domains: 0
  Number of neighbors: 1
    Address          MAC      MAC+IP      AD      IM      ES
    10.255.255.3      0        0          0       0       1
  
```

Meaning

The output provides the following information:

- EVPN routing instance
- Mode of operation of each interface
- Neighbors of the routing instance
- Number of different routes received from each neighbor
- ESI attached to the routing instance
- Number of Ethernet segments on the routing instance
- Designated forwarder (DF) election roles for each ESI in an EVPN instance (EVI)
- VLAN ID and MAC labels for the routing instance

RELATED DOCUMENTATION

| [Example: Configuring EVPN Active-Active Multihoming](#) | 375

Example: Configuring LACP for EVPN VXLAN Active-Active Multihoming

IN THIS SECTION

- [Requirements](#) | 465
- [Overview](#) | 465

- [Configuration | 467](#)
- [Verification | 491](#)

This example shows how to configure the Link Aggregation Control Protocol (LACP) on multihomed customer edge (CE) and provider edge (PE) devices in an Ethernet VPN (EVPN) VXLAN active-active multihomed network.

Requirements

This example uses the following hardware and software components:

- Three QFX10002, QFX5100, QFX5110, QFX5200 switches, or QFX5100 Virtual Chassis configured as PE devices, and one QFX5100 switch configured as a CE device.
- Junos OS Release 17.1 or later running on all switches.

Overview

IN THIS SECTION

- [Topology | 467](#)

For another level of redundancy, you can configure EVPN VXLAN active-active multihoming by configuring LACP on both the endpoints of the multihomed CE-PE link. The multihomed devices are configured with aggregated trunk links, where the link aggregation group (LAG) interfaces of the CE-PE link can either be in the active or in the standby state. When the LAG interface is in the active state, data traffic is transmitted over the CE-PE link. When the LAG interface is in the standby state, data traffic is blocked and only control traffic for communicating LAG interface state is transmitted over the link.

LACP monitors and operates the LAG interface to ensure fast convergence on isolation of a multihomed PE device from the core. When there is a core failure, a null route can occur at the isolated PE device. However, with the support for LACP on the CE-PE link, at the time of core isolation, the CE-facing interface of the multihomed PE device is set to the standby state, thereby blocking data traffic transmission from and toward the multihomed PE device. After the core recovers from the failure, the interface state is switched back from standby to active.



NOTE: On QFX10002 and QFX10008 switches, only LACP for EVPN active-active multihoming with VXLAN is supported.

When you configure LACP on the CE-PE link, isolation of the multihomed PE device from the core is handled as follows:

1. The LACP peers synchronize the configuration and operational data.

The LACP peers synchronize by exchanging control PDUs, and is required for the following reasons:

- To determine the state of the links in the Ethernet bundle—all-active or standby.
 - To detect and handle CE device misconfiguration when LACP Port Key is configured on the PE device.
 - To detect and handle miswiring between CE and PE devices when LACP Port Key is configured on the PE device.
 - To detect and react to actor or partner churn when the LACP speakers are not able to converge.
2. When the peers are null for a multihomed PE device, the PE device is isolated from the core. In this case, the isolated PE device notifies the CE devices that are connected to the isolated PE device that there is a core failure.
 3. Data traffic is not forwarded from the CE device to the isolated multihomed PE device. Instead, the traffic is diverted to the other multihomed PE devices that belong to the same LAG. This prevents traffic black holes at the isolated PE device.
 4. If the multihomed CE device uses the LAG for load balancing traffic to multiple active multihomed PE devices, then the LACP configuration along with the same system ID configured on all the multihomed PE devices for that given LAG, triggers an LACP out-of-sync to all the attached multihomed CE links.

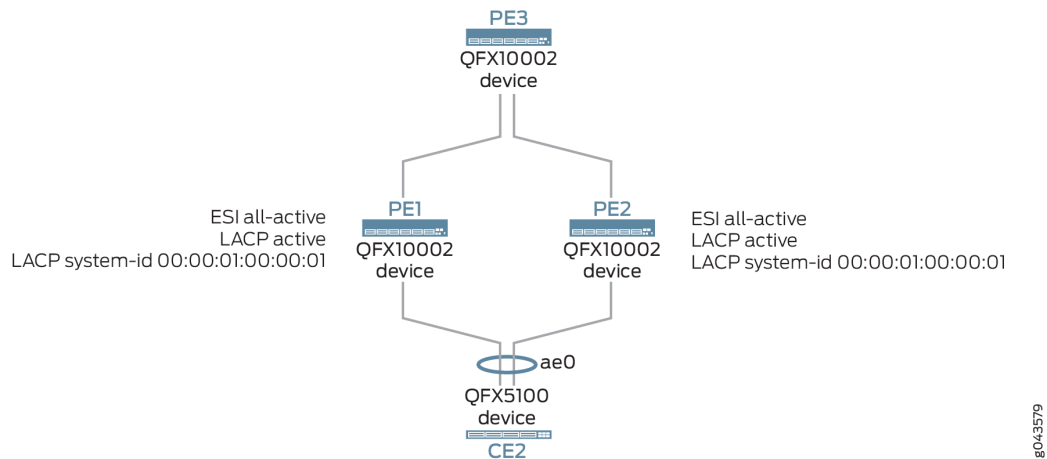
When configuring LACP on the multihomed devices, be aware of the following considerations:

- The LAG link can operate either in the active or in the standby state regardless of the UP/DOWN operational state.
- When you reboot the system, the LACP link on the multihomed PE device is in the active state.
- When the control plane goes down, LACP is notified to run multiplexer state machine for the aggregation port and move the link from active to standby.
- An interface is not treated as up unless it operates in the active state and its operational state is also up.

Topology

Figure 31 on page 467 illustrates an EVPN VXLAN active-active multihoming network with LACP configured on the multi-homed CE and PE devices. Device CE1 is single-homed and is connected to remote PE1 and PE2 devices. Device CE2 is multi-homed to PE1 and PE2 devices.

Figure 31: LACP Support in EVPN Active-Active Multihoming



Core isolation of Device PE1, for example, is handled as follows:

1. After PE2 and PE1 establish a BGP session, LACP sets the state of the CE-PE link to unblocking mode.
2. When there is a core failure, this can cause a null route at Device CE1.

To prevent this situation, the LAG interface that is facing Device CE2 is changed from the active state to the standby state by LACP.

3. LACP sends an out of-sync notification on the attached multihomed CE2 link to block traffic transmission between Device CE2 and Device PE1.
4. When the control plane recovers, Device PE2 is switched back from standby to active by LACP.

Configuration

IN THIS SECTION

● CLI Quick Configuration | 468

- [Configuring LACP for EVPN Active-Active Multihoming on PE3 | 471](#)
- [Configuring LACP for EVPN Active-Active Multihoming on PE1 | 476](#)
- [Configuring LACP for EVPN Active-Active Multihoming on PE2 | 482](#)
- [Configuring LACP for EVPN Active-Active Multihoming on CE2 | 489](#)

CLI Quick Configuration

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

Device PE3

```
set interfaces xe-0/0/0 unit 0 family inet address 10.10.10.1/24
set interfaces xe-0/0/2 unit 0 family inet address 10.12.12.1/24
set interfaces xe-0/0/4 unit 0 family inet address 10.14.14.1/24
set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members all
set interfaces lo0 unit 0 family inet address 10.2.3.1/32 primary
set vlans v100 vlan-id 100
set vlans v200 vlan-id 200
set vlans v100 vxlan vni 100
set vlans v200 vxlan vni 200
set routing-options router-id 10.2.3.1
set routing-options autonomous-system 65011
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.2.3.1:1
set switch-options vrf-target target:1111:11
set protocols bgp group pe type internal
set protocols bgp group pe local-address 10.2.3.1
set protocols bgp group pe family evpn signaling
set protocols bgp group pe neighbor 10.2.3.3
set protocols bgp group pe neighbor 10.2.3.4
set protocols ospf area 0.0.0.0 interface xe-0/0/2
set protocols ospf area 0.0.0.0 interface lo0 passive
set protocols ospf area 0.0.0.0 interface xe-0/0/0
set protocols ospf area 0.0.0.0 interface xe-0/0/4
```

```
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list all
```

Device PE1

```
set chassis aggregated-devices ethernet device-count 1
set interfaces xe-0/0/2 unit 0 family inet address 10.12.12.2/24
set interfaces xe-0/0/3 unit 0 family inet address 10.11.11.2/24
set interfaces xe-0/0/9 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/9 unit 0 family ethernet-switching vlan members v200
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members all
set interfaces xe-0/0/55:0 ether-options 802.3ad ae0
set interfaces ae0 esi 00:03:03:03:03:03:03:03:03
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:01:00:00:01
set interfaces lo0 unit 0 family inet address 10.2.3.3/32 primary
set vlans v100 vlan-id 100
set vlans v200 vlan-id 200
set vlans v100 vxlan vni 100
set vlans v200 vxlan vni 200
set routing-options router-id 10.2.3.3
set routing-options autonomous-system 65011
set switch-options vtep-source-interface lo0
set switch-options route-distinguisher 10.2.3.3:1
set switch-options vrf-target target:1111:11
set protocols bgp group pe type internal
set protocols bgp group pe local-address 10.2.3.3
set protocols bgp group pe family evpn signaling
set protocols bgp group pe neighbor 10.2.3.1
set protocols bgp group pe neighbor 10.2.3.4
set protocols ospf area 0.0.0.0 interface xe-0/0/2
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface xe-0/0/3
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list all
```

Device PE2

```

set chassis aggregated-devices ethernet device-count 1
set interfaces xe-0/0/2 unit 0 family inet address 10.14.14.2/24
set interfaces xe-0/0/5 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/5 unit 0 family ethernet-switching vlan members all
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members all
set interfaces xe-0/0/55:0 ether-options 802.3ad ae0
set interfaces ae0 esi 00:03:03:03:03:03:03:03:03
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:01:00:00:01
set interfaces lo0 unit 0 family inet address 10.2.3.4/32 primary
set vlans v100 vlan-id 100
set vlans v200 vlan-id 200
set vlans v100 vxlan vni 100
set vlans v200 vxlan vni 200
set routing-options router-id 10.2.3.4
set routing-options autonomous-system 65011
set switch-options vtep-source-interface lo0
set switch-options route-distinguisher 10.2.3.4:1
set switch-options vrf-target target:1111:11
set protocols bgp group pe type internal
set protocols bgp group pe local-address 10.2.3.4
set protocols bgp group pe family evpn signaling
set protocols bgp group pe neighbor 10.2.3.1
set protocols bgp group pe neighbor 10.2.3.2
set protocols bgp group pe neighbor 10.2.3.3
set protocols ospf area 0.0.0.0 interface xe-0/0/2
set protocols ospf area 0.0.0.0 interface lo0 passive
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list all

```

Device CE2

```

set vlans v100 vlan-id 100
set vlans v200 vlan-id 200
set interfaces xe-0/0/3 unit 0 family ethernet-switching interface-mode trunk

```

```

set interfaces xe-0/0/3 unit 0 family ethernet-switching vlan members all
set interfaces xe-0/0/5 ether-options 802.3ad ae0
set interfaces xe-0/0/0 ether-options 802.3ad ae0
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members all
set interfaces ae0 aggregated-ether-options lacp active

```

Configuring LACP for EVPN Active-Active Multihoming on PE3

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 PE3:

1. Configure uplink interfaces towards PE1 and PE2 devices.

```

[edit interfaces]
user@CE1# set xe-0/0/0 unit 0 family inet address 10.10.10.1/24
user@CE1# set xe-0/0/2 unit 0 family inet address 12.12.12.1/24
user@CE1# set xe-0/0/4 unit 0 family inet address 14.14.14.1/24

```

2. Configure xe-0/0/8 as a Layer 2 interface.

```

[edit interfaces]
user@CE1# set xe-0/0/8 unit 0 family ethernet-switching interface-mode trunk
user@CE1# set xe-0/0/8 unit 0 family ethernet-switching vlan members all

```

3. Configure a loopback interface.

```

[edit interfaces]
user@CE1# set lo0 unit 0 family inet address 10.2.3.1/32 primary

```


4. Create VLANs v100 and v200.

```
[edit vlans]
user@CE1# set v100 vlan-id 100
user@CE1# set v200 vlan-id 200
```

5. Map VLANs v100 and v200 to VNIs 100 and 200.

```
[edit vlans]
user@CE1# set v100 vxlan vni 100
user@CE1# set v200 vxlan vni 200
```

6. Configure the router ID and autonomous system number.

```
[edit routing-options]
user@CE1# set router-id 10.2.3.1
user@CE1# set autonomous-system 65011
```

7. Specify the loopback interface as the source address for the VTEP tunnel.

```
[edit switch-options]
user@CE1# set vtep-source-interface lo0.0
```

8. Specify a route distinguisher to uniquely identify routes sent from this device.

```
[edit switch-options]
user@CE1# set route-distinguisher 10.2.3.1:1
```

9. Specify the global VRF export policy.

```
[edit switch-options]
user@CE1# set vrf-target target:1111:11
```

10. Configure an internal BGP group for PE3 to peer with PE1 and PE2.

```
[edit protocols]
user@CE1# set bgp group pe type internal
user@CE1# set bgp group pe local-address 10.2.3.1
user@CE1# set bgp group pe family evpn signaling
user@CE1# set bgp group pe neighbor 10.2.3.3
user@CE1# set bgp group pe neighbor 10.2.3.4
```

11. Configure an OSPF area.

```
[edit protocols]
user@CE1# set ospf area 0.0.0.0 interface xe-0/0/2
user@CE1# set ospf area 0.0.0.0 interface lo0 passive
user@CE1# set ospf area 0.0.0.0 interface xe-0/0/0
user@CE1# set ospf area 0.0.0.0 interface xe-0/0/4
```

12. Set VXLAN as the data plane encapsulation for EVPN.

```
[edit protocols]
user@CE1# set evpn encapsulation vxlan
```

13. Specify that all VNI(s) are advertised by EVPN.

```
[edit protocols]
user@CE1# set evpn extended-vni-list all
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, 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@PE3# show interfaces
xe-0/0/0 {
  unit 0 {
    family inet {
```

```

        address 10.10.10.1/24;
    }
}
xe-0/0/2 {
    unit 0 {
        family inet {
            address 10.10.6.1/30;
            address 10.12.12.1/24;
        }
    }
}
xe-0/0/4 {
    unit 0 {
        family inet {
            address 10.14.14.1/24;
        }
    }
}
xe-0/0/8 {
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members all;
            }
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.10.0.1/32;
            address 10.2.3.1/32 {
                primary;
            }
        }
    }
}

```

```
    }
}
```

```
user@PE3# show vlans
```

```
v100 {
    vlan-id 100;
    vxlan {
        vni 100;
    }
}
v200 {
    vlan-id 200;
    vxlan {
        vni 200;
    }
}
```

```
user@PE3# show routing-options
```

```
router-id 10.2.3.1;
autonomous-system 65011;
```

```
user@PE3# show switching-options
```

```
vtep-source-interface lo0;
route-distinguisher 10.2.3.1:1;
vrf-target target:1111:11;
```

```
user@PE3# show protocols bgp
```

```
group pe {
    type internal;
    local-address 10.2.3.1;
    family evpn {
        signaling;
    }
    neighbor 10.2.3.3;
```

```
neighbor 10.2.3.4;
}
```

```
user@PE3# show protocols ospf
area 0.0.0.0 {
    interface lo0.0 {
        passive;
    }
    interface xe-0/0/2;
    interface xe-0/0/0;
    interface xe-0/0/4;
}
```

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

Configuring LACP for EVPN Active-Active Multihoming on 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:

1. Specify the number of aggregated Ethernet interfaces to be created on Device PE1.

```
[edit chassis]
user@PE1# set aggregated-devices ethernet device-count 1
```

2. Configure the interfaces that connect to the CE device.

```
[edit interfaces]
user@PE1# set xe-0/0/2 unit 0 family inet address 10.12.12.2/24
user@PE1# set xe-0/0/2 unit 0 family inet address 10.11.11.2/24
```

3. Configure xe-0/0/9 as a Layer 2 interface.

```
[edit interfaces]
user@PE1# set xe-0/0/9 unit 0 family ethernet-switching interface-mode trunk
user@PE1# set xe-0/0/9 unit 0 family ethernet-switching vlan members v200
```

4. Configure ae0 as a Layer 2 interface.

```
[edit interfaces]
user@PE1# set ae0 unit 0 family ethernet-switching interface-mode trunk
user@PE1# set ae0 unit 0 family ethernet-switching vlan members all
```

5. Configure the interface towards the multihomed device, CE2.

Use the same ESI value on all PE devices where the CE2 is multihomed.

```
[edit interfaces]
user@PE1# set xe-0/0/55:0 ether-options 802.3ad ae0
user@PE1# set ae0 esi 00:03:03:03:03:03:03:03:03
user@PE1# set ae0 esi all-active
```

6. Configure LACP on the ae0.

Use the same system ID value on all PE devices where the CE2 is multihomed.

```
[edit interfaces]
user@PE1# set ae0 aggregated-ether-options lacp active
user@PE1# set ae0 aggregated-ether-options lacp system-id 00:00:01:00:00:01
```

7. Configure a loopback interface.

```
[edit interfaces]
user@PE1# set lo0 unit 0 family inet address 10.2.3.3/32 primary
```

8. Create VLANs v100 and v200.

```
[edit vlans]
user@PE1# set v100 vlan-id 100
user@PE1# set v200 vlan-id 200
```

9. Map VLANs v100 and v200 to VNIs 100 and 200.

```
[edit vlans]
user@PE1# set v100 vxlan vni 100
user@PE1# set v200 vxlan vni 200
```

10. Configure a router ID and autonomous system number.

```
[edit routing-options]
user@PE1# set router-id 10.2.3.3
user@PE1# set autonomous-system 65011
```

11. Specify the loopback interface as the source address for the VTEP tunnel.

```
[edit switch-options]
user@PE1# set vtep-source-interface lo0.0
```

12. Specify a route distinguisher to uniquely identify routes sent from this device.

```
[edit switch-options]
user@PE1# set route-distinguisher 10.2.3.3:1
```

13. Specify the global VRF export policy.

```
[edit switch-options]
user@PE1# set vrf-target target:1111:11
```

14. Configure an internal BGP group for PE3 to peer with PE1 and PE2.

```
[edit protocols]
user@PE1# set bgp group pe type internal
user@PE1# set bgp group pe local-address 10.2.3.3
user@PE1# set bgp group pe family evpn signaling
user@PE1# set bgp group pe neighbor 10.2.3.1
user@PE1# set bgp group pe neighbor 10.2.3.4
```

15. Configure an OSPF area.

```
[edit protocols]
user@PE1# set ospf area 0.0.0.0 interface xe-0/0/2
user@PE1# set ospf area 0.0.0.0 interface lo0 passive
user@PE1# set ospf area 0.0.0.0 interface xe-0/0/3
```

16. Set VXLAN as the data plane encapsulation for EVPN.

```
[edit protocols]
user@PE1# set evpn encapsulation vxlan
```

17. Specify that all VNI(s) are advertised by EVPN.

```
[edit protocols]
user@PE1# set evpn extended-vni-list all
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, 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 chassis
aggregated-devices {
  ethernet {
    device-count 1;
```



```

    }
}

```

```
user@PE1# show interfaces
```

```

xe-0/0/2 {
  unit 0 {
    family inet {
      address 10.10.6.1/30;
      address 10.12.12.1/24;
      address 10.12.12.2/24;
    }
  }
}
xe-0/0/3 {
  unit 0 {
    family inet {
      address 10.11.11.2/24;
    }
  }
}
xe-0/0/9 {
  unit 0 {
    family ethernet-switching {
      interface-mode trunk;
      vlan {
        members v200;
      }
    }
  }
}
xe-0/0/55:0 {
  ether-options {
    802.3ad ae0;
  }
}
ae0 {
  esi {
    00:03:03:03:03:03:03:03:03;
    all-active;
  }
  aggregated-ether-options {

```

```

        lacp {
            active;
            system-id 00:00:01:00:00:01;
        }
    }
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members all;
            }
        }
    }
}

```

```
user@PE1# show vlans
```

```

v100 {
    vlan-id 100;

    vxlan {
        vni 100;
    }
}
v200 {
    vlan-id 200;

    vxlan {
        vni 200;
    }
}

```

```
user@PE1# show routing-options
```

```

router-id 10.2.3.3;
autonomous-system 65011;

```

```
user@PE1# show switching-options
```

```
vtep-source-interface lo0;
```

```
route-distinguisher 10.2.3.3:1;
vrf-target target:1111:11;
```

```
user@PE1# show protocols bgp
group pe {
    type internal;
    local-address 10.2.3.3;
    family evpn {
        signaling;
    }
    neighbor 10.2.3.3;
    neighbor 10.2.3.4;
    neighbor 10.2.3.1;
}
```

```
user@PE1# show protocols ospf
area 0.0.0.0 {
    interface lo0 {
        passive;
    }
    interface xe-0/0/2;
    interface xe-0/0/3;
}
```

```
user@PE1# show protocols evpn
encapsulation vxlan;
extended-vni-list all;
```

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

Configuring LACP for EVPN Active-Active Multihoming on PE2

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 PE2:

1. Specify the number of aggregated Ethernet interfaces to be created on Device PE1.

```
[edit chassis]
user@PE2# set aggregated-devices ethernet device-count 1
```

2. Configure the interface that connects to the CE device.

```
[edit interfaces]
user@PE2# set xe-0/0/2 unit 0 family inet address 10.14.14.2/24
```

3. Configure xe-0/0/5 as a Layer 2 interface.

```
[edit interfaces]
user@PE2# set xe-0/0/5 unit 0 family ethernet-switching interface-mode trunk
user@PE2# set xe-0/0/5 unit 0 family ethernet-switching vlan members v200
```

4. Configure ae0 as a Layer 2 interface.

```
[edit interfaces]
user@PE2# set ae0 unit 0 family ethernet-switching interface-mode trunk
user@PE2# set ae0 unit 0 family ethernet-switching vlan members all
```

5. Configure the interface towards the multihomed device, CE2.

Use the same ESI value on all PE devices where the CE2 is multihomed.

```
[edit interfaces]
user@PE2# set xe-0/0/55:0 ether-options 802.3ad ae0
user@PE2# set ae0 esi 00:03:03:03:03:03:03:03:03
user@PE2# set ae0 esi all-active
```

6. Configure LACP on the ae0.

Use the same system ID value on all PE devices where the CE2 is multihomed.

```
[edit interfaces]
user@PE2# set ae0 aggregated-ether-options lacp active
user@PE2# set ae0 aggregated-ether-options lacp system-id 00:00:01:00:00:01
```

7. Configure a loopback interface.

```
[edit interfaces]
user@PE2# set lo0 unit 0 family inet address 10.2.3.4/32 primary
```

8. Create VLANs v100 and v200.

```
[edit vlans]
user@PE2# set v100 vlan-id 100
user@PE2# set v200 vlan-id 200
```

9. Map VLANs v100 and v200 to VNIs 100 and 200.

```
[edit vlans]
user@PE2# set v100 vxlan vni 100
user@PE2# set v200 vxlan vni 200
```

10. Configure a router ID and autonomous system number.

```
[edit routing-options]
user@PE2# set router-id 10.2.3.4
user@PE2# set autonomous-system 65011
```

11. Specify the loopback interface as the source address for the VTEP tunnel.

```
[edit switch-options]
user@PE2# set vtep-source-interface lo0.0
```

12. Specify a route distinguisher to uniquely identify routes sent from this device.

```
[edit switch-options]
user@PE2# set route-distinguisher 10.2.3.4:1
```

13. Specify the global VRF export policy.

```
[edit switch-options]
user@PE2# set vrf-target target:1111:11
```

14. Configure an internal BGP group for PE3 to peer with PE1 and PE2.

```
[edit protocols]
user@PE2# set bgp group pe type internal
user@PE2# set bgp group pe local-address 10.2.3.4
user@PE2# set bgp group pe family evpn signaling
user@PE2# set bgp group pe neighbor 10.2.3.1
user@PE2# set bgp group pe neighbor 10.2.3.2
user@PE2# set bgp group pe neighbor 10.2.3.3
```

15. Configure an OSPF area.

```
[edit protocols]
user@PE2# set ospf area 0.0.0.0 interface xe-0/0/2
user@PE2# set ospf area 0.0.0.0 interface lo0 passive
```

16. Set VXLAN as the data plane encapsulation for EVPN.

```
[edit protocols]
user@PE2# set evpn encapsulation vxlan
```

17. Specify that all VNI(s) are advertised by EVPN.

```
[edit protocols]
user@PE2# set evpn extended-vni-list all
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, 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 chassis
aggregated-devices {
  ethernet {
    device-count 1;
  }
}
```

```
user@PE2# show interfaces
xe-0/0/2 {
  unit 0 {
    family inet {
      address 10.14.14.2/24;
    }
  }
}
xe-0/0/5 {
  unit 0 {
    family ethernet-switching {
      interface-mode trunk;
      vlan {
        members all;
      }
    }
  }
}
xe-0/0/55:0 {
  ether-options {
    802.3ad ae0;
  }
}
ae0 {
  esi {
    00:03:03:03:03:03:03:03:03;
    all-active;
  }
}
```

```

    aggregated-ether-options {
        lacp {
            active;
            system-id 00:00:01:00:00:01;
        }
    }
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members all;
            }
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.2.3.4/32 {
                primary;
            }
        }
    }
}
}

```

```

user@PE2# show vlans

```

```

v100 {
    vlan-id 100;

    vxlan {
        vni 100;
    }
}
v200 {
    vlan-id 200;

    vxlan {
        vni 200;
    }
}

```



```

    }
}

```

```

user@PE2# show routing-options
router-id 10.2.3.4;
autonomous-system 65011;

```

```

user@PE2# show switching-options
vtep-source-interface lo0;
route-distinguisher 10.2.3.4:1;
vrf-target target:1111:11;

```

```

user@PE2# show protocols bgp
group pe {
    type internal;
    local-address 10.2.3.4;
    family evpn {
        signaling;
    }
    neighbor 10.2.3.3;
    neighbor 10.2.3.4;
    neighbor 10.2.3.1;
}

```

```

user@PE2# show protocols ospf
area 0.0.0.0 {
    interface lo0 {
        passive;
    }
    interface xe-0/0/2;
}

```

```

user@PE2# show protocols evpn
encapsulation vxlan;
extended-vni-list all;

```

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

Configuring LACP for EVPN Active-Active Multihoming on CE2

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 CE1:

1. Configure VLANs v100 and v200.

```
[edit vlans]
user@CE2# set v100 vlan-id 100
user@CE2# set v200 vlan-id 200
```

2. Configure xe-0/0/3 as a Layer 2 interface.

```
[edit interfaces]
user@CE2# set xe-0/0/3 unit 0 family ethernet-switching interface-mode trunk
user@CE2# set xe-0/0/3 unit 0 family ethernet-switching vlan members all
```

3. Add member interfaces to ae0.

```
[edit interfaces]
user@CE2# set xe-0/0/0 ether-options 802.3ad ae0
user@CE2# set xe-0/0/5 ether-options 802.3ad ae0
```

4. Configure ae0 as a Layer 2 interface.

```
[edit interfaces]
user@CE2# set ae0 unit 0 family ethernet-switching interface-mode trunk
user@CE2# set ae0 unit 0 family ethernet-switching vlan members all
```

5. Configure LACP as active for ae0.

```
[edit interfaces]
user@CE2# set ae0 aggregated-ether-options lacp active
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, 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@CE2# show interfaces
xe-0/0/0 {
  ether-options {
    802.3ad ae0;
  }
}
xe-0/0/3 {
  unit 0 {
    family ethernet-switching {
      interface-mode trunk;
      vlan {
        members all;
      }
    }
  }
}
```

```
user@CE2# show vlans
v100 {
  vlan-id 100;
  vxlan {
    vni 100;
  }
}
v200 {
  vlan-id 200;
  vxlan {
    vni 200;
```

```
}  
}
```

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

Verification

IN THIS SECTION

[Verifying LACP Interface Status of PE1 | 491](#)

[Verifying LACP Interface Status of PE2 | 492](#)

Confirm that the configuration is working properly.

Verifying LACP Interface Status of PE1

Purpose

Verify the LACP interface state on Device PE1.

Action

From operational mode, run the **show lacp interfaces** command.

```
user@PE1> show lacp interfaces  
Aggregated interface: ae0  
LACP state:      Role  Exp  Def  Dist  Col  Syn  Aggr  Timeout  Activity  
xe-0/0/55:0      Actor No   No   Yes  Yes  Yes  Yes   Fast    Active  
xe-0/0/55:0      Partner No   No   Yes  Yes  Yes  Yes   Fast    Active  
LACP protocol:  Receive State  Transmit State  Mux State  
xe-0/0/55:0      Current      Fast periodic  Collecting distributing
```

Meaning

The LACP LAG interface state is active.



NOTE: In our example all links and protocols are operational. When all BGP sessions used for Ethernet VPN (EVPN) are down the core isolation feature activates. In this state the leaf brings down LACP to prevent the attached device from sending data. On the leaf the LACP interface status is updated to show core isolation down (CDN). To demonstrate we deactivated BGP on the leaf. The LACP interface status is updated to show core isolation is in effect:

```
user@PE1> show lacp interfaces
```

```
Aggregated interface: ae0
```

LACP state:	Role	Exp	Def	Dist	Col	Syn	Aggr	Timeout	Activity
xe-0/0/55:0 CDN	Actor	No	No	No	No	No	Yes	Fast	Active
xe-0/0/55:0 CDN	Partner	No	No	No	No	Yes	Yes	Fast	Active

LACP protocol:	Receive State	Transmit State	Mux State
xe-0/0/55:0	Current	Fast periodic	Waiting

For details on the core isolation feature see ["Understanding When to Disable EVPN-VXLAN Core Isolation" on page 514](#).

Verifying LACP Interface Status of PE2

Purpose

Verify the LACP interface state on Device PE2.

Action

From operational mode, run the **show lacp interfaces** command.

```
user@PE2> show lacp interfaces
```

```
Aggregated interface: ae0
```

LACP state:	Role	Exp	Def	Dist	Col	Syn	Aggr	Timeout	Activity
xe-0/0/55:0	Actor	No	No	Yes	Yes	Yes	Yes	Fast	Active
xe-0/0/55:0	Partner	No	No	Yes	Yes	Yes	Yes	Fast	Active

LACP protocol:	Receive State	Transmit State	Mux State
xe-0/0/55:0	Current	Fast periodic	Collecting distributing

Meaning

The LACP LAG interface state is active.

RELATED DOCUMENTATION

[Understanding LACP for EVPN Active-Active Multihoming](#)

Example: Configuring an ESI on a Logical Interface With EVPN-MPLS Multihoming

IN THIS SECTION

- [Requirements | 494](#)
- [Overview and Topology | 494](#)
- [EVPN Multihoming Active-Standby Configuration | 497](#)
- [Verification | 507](#)

When a customer edge (CE) device in an Ethernet VPN-Multiprotocol Label Switching (EVPN-MPLS) environment is multihomed to two or more provider edge (PE) devices, the set of Ethernet links that connect the devices comprise an Ethernet segment. An Ethernet segment identifier (ESI) is a 10-octet integer that identifies this segment. A sample ESI is 00:11:22:33:44:55:66:77:88:99.

In releases before Junos OS Release 15.1F6 and 16.1R4 for MX Series routers and in releases before Junos OS Release 17.3R1 for EX9200 switches, you can specify an ESI only on a physical or aggregated Ethernet interface, for example, set interfaces ae0 esi 00:11:22:33:44:55:66:77:88:99. If you specify an ESI on a physical or aggregated Ethernet interface, keep in mind that an ESI is a factor in the designated forwarder (DF) election process. For example, assume that you configure EVPN multihoming active-standby on aggregated Ethernet interface ae0, and given the ESI configured on ae0 and other determining factors, the DF election results in ae0 being in the down state. Further, all logical interfaces configured on ae0, for example, set interfaces ae0 unit 1 and set interfaces ae0 unit 2 are also in the down state, which renders logical interfaces ae0.1 and ae0.2 unable to provide services to their respective customer sites (VLANs).

You can specify an ESI on a logical interface. If you specify an ESI on a logical interface, the DF election process now occurs at the individual logical interface level, which enables you to better utilize logical

interfaces. For example, assume that you configure logical interfaces ae0.1 and ae0.2 on aggregated Ethernet interface ae0. You configure EVPN multihoming active-standby on both logical interfaces and given the ESI configured on ae0.1 and other determining factors, the DF election results in ae0.1 being in the down state. Despite logical interface ae0.1 being down, logical interface ae0.2 and other logical interfaces configured on ae0 can be in the up state and provide services to their respective customer sites (VLANs).

This topic shows how to configure an ESI on logical interfaces in both EVPN multihoming active-standby and active-active modes.



NOTE: We support active-standby multihoming in EVPN fabrics only with MPLS. Please refer to [EVPN-MPLS: Single-active multihoming support](#) for a complete list of the products that support this feature.

Requirements

Both EVPN multihoming active-standby and multihoming active-active examples use the following hardware and software components:

- An EX9200 switch running Junos OS Release 17.3R1 or later (PE1)
- An MX Series router running Junos OS Release 15.1F6 or later, or Junos OS Release 17.1R1 or later (PE2)

Overview and Topology

EVPN Multihoming Active-Standby

[Figure 32 on page 495](#) shows an EVPN-MPLS topology in which CE1 is multihomed to PE1 and PE2 to provide redundant paths to CE2. On CE1, the connections to PE1 and PE2 are configured as separate aggregated Ethernet interfaces. [Table 21 on page 495](#) shows how the connections with CE1 are configured on PE1 and PE2. Note that the EVPN multihoming mode, ESI, and VLAN ID are actually configured on logical interfaces ae0.1 on each PE device.

Figure 32: EVPN-MPLS Topology with Multihoming Active-Standby

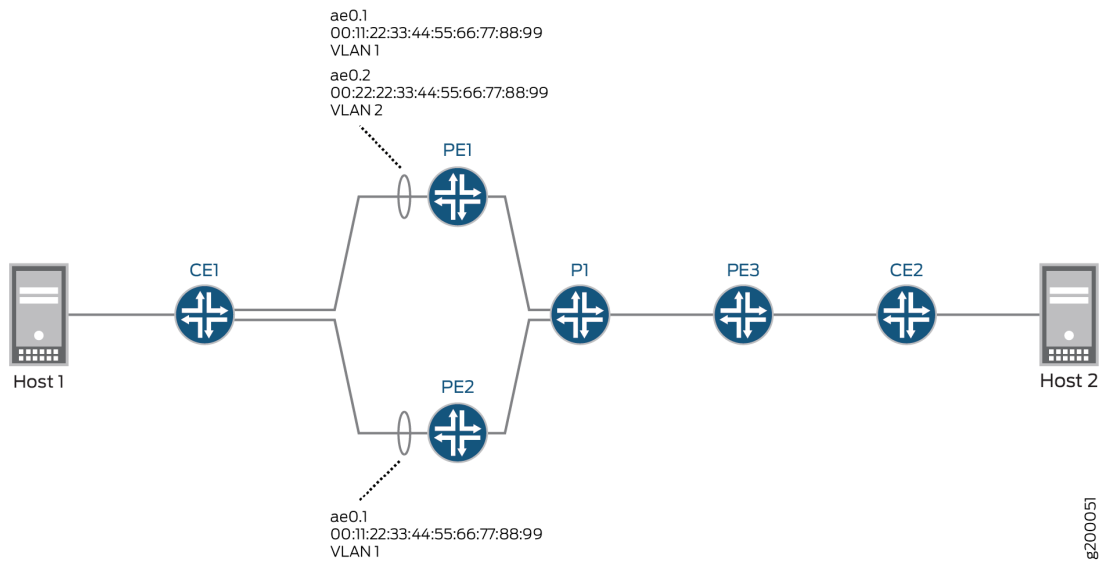


Table 21: EVPN Multihoming Active-Standby: Configuring the Connection with CE1 on PE1 and PE2

Device	Physical Interface	Aggregated Ethernet Interface	Logical Interface	EVPN Multihoming Mode	ESI	VLAN ID
PE1	xe-2/0/0	ae0	ae0.1	single-active	00:11:22:33:44:55:66:77:88:99	1
PE2	xe-3/0/2	ae0	ae0.1	single-active	00:11:22:33:44:55:66:77:88:99	1

Based on the DF election, logical interface ae0.1 on PE2 is up, and logical interface ae0.1 on PE1 is down.

[Table 22 on page 496](#) also shows the configuration for logical interface ae0.2 on PE1. Note that logical interface ae0.2 provides services for a different VLAN and is configured with a different ESI than logical interface ae0.1, which is configured on the same aggregated Ethernet interface. As a result, logical interface ae0.2 is up and providing services to VLAN 2 despite the fact that logical interface ae0.1 is in the down state.

Table 22: Multihoming Active-Standby: Configuring Logical Interface on PE1 that Provides Services to a Different VLAN

Device	Physical Interface	Aggregated Ethernet Interface	Logical Interface	EVPN Multihoming Mode	ESI	VLAN ID
PE1	xe-2/0/0	ae0	ae0.2	single-active	00:22:22:33:44:55:66:77:88:99	2

EVPN Multihoming Active-Active

Figure 33 on page 496 shows an EVPN-MPLS topology in which CE1 is multihomed to PE1 and PE2 to provide redundant paths to CE2. On CE1, the connections to PE1 and PE2 are configured as one aggregated Ethernet interface. Table 23 on page 497 shows how the connections with CE1 are configured on PE1 and PE2. Note that the EVPN multihoming mode, ESI, and VLAN ID are actually configured on logical interfaces ae0.1 on each device.

Figure 33: EVPN-MPLS Topology with Multihoming Active-Active

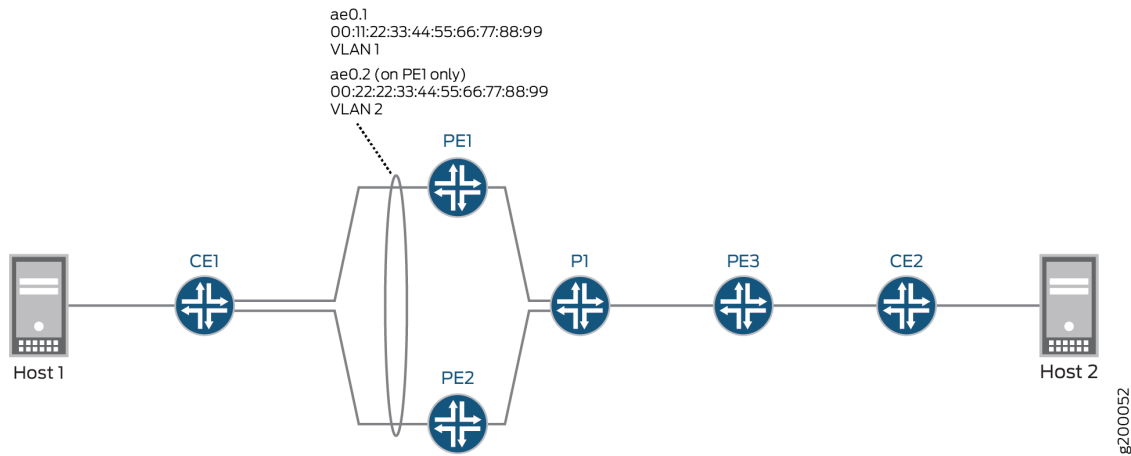


Table 23: Multihoming Active-Active: Configuring the Connection with CE1 on PE1 and PE2

Device	Physical Interface	Aggregated Ethernet Interface	Logical Interface	EVPN Multihoming Mode	ESI	VLAN ID
PE1	xe-2/0/0	ae0	ae0.1	all-active	00:11:22:33:44:55:66:77:88:99	1
PE2	xe-3/0/2	ae0	ae0.1	all-active	00:11:22:33:44:55:66:77:88:99	1

Based on the DF election, logical interface ae0.1 on PE1 is in the up state, and logical interface ae0.1 on PE2 is in the down state.

[Table 24 on page 497](#) also shows the configuration for logical interface ae0.2 on PE1. Note that logical interface ae0.2 provides services for a different VLAN and is configured with a different ESI than logical interface ae0.1, which is in the same aggregated Ethernet interface. As a result, logical interface ae0.2 is down and unable to provide services to VLAN 2 despite the fact that logical interface ae0.1 is in the up state.

Table 24: EVPN Multihoming Active-Active: Configuring Interface on PE1 that Provides Services to a Different VLAN

Device	Physical Interface	Aggregated Ethernet Interface	Logical Interface	EVPN Multihoming Mode	ESI	VLAN ID
PE1	xe-2/0/0	ae0	ae0.2	all-active	00:22:22:33:44:55:66:77:88:99	2

EVPN Multihoming Active-Standby Configuration

IN THIS SECTION

- [CLI Quick Configuration | 498](#)
- [Procedure | 499](#)
- [EVPN Multihoming Active-Active Configuration | 503](#)



NOTE: The configurations for PE1 (EX9200) and PE2 (MX Series router) focus on configuring EVPN multihoming active-standby and ESIs on logical interfaces. The configurations do not include all EVPN-related configurations for physical interfaces, aggregated Ethernet interfaces, logical interfaces, and routing instances. For a more comprehensive configuration of EVPN multihoming active-standby in an EVPN-MPLS environment, see ["Example: Configuring Basic EVPN-MPLS Active-Standby Multihoming" on page 284.](#)

CLI Quick Configuration

PE1

```

set interfaces xe-2/0/0 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 unit 1 encapsulation vlan-bridge
set interfaces ae0 unit 1 vlan-id 1
set interfaces ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 unit 1 esi single-active
set interfaces ae0 unit 2 encapsulation vlan-bridge
set interfaces ae0 unit 2 vlan-id 2
set interfaces ae0 unit 2 esi 00:22:22:33:44:55:66:77:88:99
set interfaces ae0 unit 2 esi single-active
set interfaces irb unit 1 family inet address 192.0.2.1/24
set interfaces irb unit 2 family inet address 192.0.2.2/24
set routing-instances blue instance-type evpn
set routing-instances blue vlan-id 1
set routing-instances blue interface ae0.1
set routing-instances blue l3-interface irb.1
...
set routing-instances blue protocols evpn interface ae0.1
set routing-instances green instance-type evpn
set routing-instances green vlan-id 2
set routing-instances green interface ae0.2
set routing-instances green l3-interface irb.2
...
set routing-instances green protocols evpn interface ae0.2
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.1

```

```
set routing-instances vrf interface irb.2
...
```

PE2

```
set interfaces xe-3/0/2 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 unit 1 encapsulation vlan-bridge
set interfaces ae0 unit 1 vlan-id 1
set interfaces ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 unit 1 esi single-active
set interfaces irb unit 1 family inet address 192.0.2.1/24
set routing-instances blue instance-type evpn
set routing-instances blue vlan-id 1
set routing-instances blue interface ae0.1
set routing-instances blue routing-interface irb.1
...
set routing-instances blue protocols evpn interface ae0.1
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.1
...
```

Procedure

Step-by-Step Procedure

To configure EVPN multihoming active-standby on PE1:

1. Specify an Ethernet interface as a member of aggregated Ethernet interface ae0.

```
[edit interfaces]
user@switch# set xe-2/0/0 gigether-options 802.3ad ae0
```

2. Configure aggregated Ethernet interface ae0 to simultaneously transmit 802.1Q VLAN single-tag and dual-tag frames and to support different types of Ethernet encapsulation at the logical interface level.

```
[edit interfaces]
user@switch# set ae0 flexible-vlan-tagging
user@switch# set ae0 encapsulation flexible-ethernet-services
```

3. On aggregated Ethernet interface ae0, configure logical interfaces ae0.1 and ae0.2. Configure the logical interfaces to use VLAN bridge encapsulation, and map the logical interfaces to VLANs 1 and 2, respectively. Also, assign an ESI to the logical interfaces, and enable EVPN multihoming active-standby.

```
[edit interfaces]
user@switch# set ae0 unit 1 encapsulation vlan-bridge
user@switch# set ae0 unit 1 vlan-id 1
user@switch# set ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
user@switch# set ae0 unit 1 esi single-active
user@switch# set ae0 unit 2 encapsulation vlan-bridge
user@switch# set ae0 unit 2 vlan-id 2
user@switch# set ae0 unit 2 esi 00:22:22:33:44:55:66:77:88:99
user@switch# set ae0 unit 2 esi single-active
```

4. Configure IRB interfaces irb.1 and irb.2, and assign an IP address to each interface.

```
[edit interfaces]
set interfaces irb unit 1 family inet address 192.0.2.1/24

set interfaces irb unit 2 family inet address 192.0.2.2/24
```

5. Configure an EVPN routing instance named blue. Map the routing instance to VLAN 1, logical interface ae0.1, and IRB interface irb.1. Configure EVPN logical interface ae0.1 for the EVPN routing instance.

```
[edit routing-instances]
user@switch# set blue instance-type evpn
user@switch# set blue vlan-id 1
user@switch# set blue interface ae0.1
```

```

user@switch# set blue l3-interface irb.1
user@switch# set blue protocols evpn interface ae0.1

```

6. Configure an EVPN routing instance named green. Map the routing instance to VLAN 2, logical interface ae0.2, and IRB interface irb.2. Configure logical interface ae0.2 for the EVPN routing instance.

```

[edit routing-instances]
user@switch# set green instance-type evpn
user@switch# set green vlan-id 2
user@switch# set green interface ae0.2
user@switch# set green l3-interface irb.2
user@switch# set green protocols evpn interface ae0.2

```

7. Configure a VRF routing instance, and add IRB interfaces irb.1 and irb.2 to the routing instance.

```

[edit routing-instances]
set vrf instance-type vrf

set vrf interface irb.1

set vrf interface irb.2

```

Step-by-Step Procedure

To configure EVPN multihoming active-standby on PE2:

1. Specify an Ethernet interface as a member of aggregated Ethernet interface ae0.

```

[edit interfaces]
user@router# set xe-3/0/2 gigether-options 802.3ad ae0

```

2. Configure aggregated Ethernet interface ae0 to simultaneously transmit 802.1Q VLAN single-tag and dual-tag frames and to support different types of Ethernet encapsulation at the logical interface level.

```
[edit interfaces]
user@router# set ae0 flexible-vlan-tagging
user@router# set ae0 encapsulation flexible-ethernet-services
```

3. On aggregated Ethernet interface ae0, configure logical interface ae0.1. Configure the logical interface to use VLAN bridge encapsulation, and map the logical interface to VLAN 1 and 2. Also, assign an ESI to the logical interface, and enable EVPN multihoming active-standby.

```
[edit interfaces]
user@router# set ae0 unit 1 encapsulation vlan-bridge
user@router# set ae0 unit 1 vlan-id 1
user@router# set ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
user@router# set ae0 unit 1 esi single-active
```

4. Configure IRB interface irb.1, and assign an IP address to the interface.

```
[edit interfaces]
user@router# set irb unit 1 family inet address 192.0.2.1/24
```

5. Configure an EVPN routing instance named blue. Map the routing instance to VLAN 1, logical interface ae0.1, and IRB interface irb.1. Configure EVPN logical interface ae0.1 for the EVPN routing instance.

```
[edit routing-instances]
user@router# set blue instance-type evpn
user@router# set blue vlan-id 1
user@router# set blue interface ae0.1
user@router# set blue routing-interface irb.1
user@router# set blue protocols evpn interface ae0.1
```

6. Configure a VRF routing instance, and add IRB interface irb.1 to the routing instance.

```
[edit routing-instances]
user@router# set vrf instance-type vrf
user@router# set vrf interface irb.1
```

EVPN Multihoming Active-Active Configuration

CLI Quick Configuration



NOTE: The configurations for PE1 (EX9200) and PE2 (MX Series router) focus on configuring EVPN multihoming active-active and ESIs on logical interfaces. The configurations do not include all EVPN-related configurations for physical interfaces, aggregated Ethernet interfaces, logical interfaces, and routing instances. For a more comprehensive configuration of EVPN multihoming active-active, see ["Example: Configuring EVPN Active-Active Multihoming" on page 375](#). Note that the referenced example shows how to configure ESIs on physical and aggregated Ethernet interfaces.

PE1

```
set interfaces xe-2/0/0 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 unit 1 encapsulation vlan-bridge
set interfaces ae0 unit 1 vlan-id 1
set interfaces ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 unit 1 esi all-active
set interfaces ae0 unit 2 encapsulation vlan-bridge
set interfaces ae0 unit 2 vlan-id 2
set interfaces ae0 unit 2 esi 00:22:22:33:44:55:66:77:88:99
set interfaces ae0 unit 2 esi all-active
set interfaces irb unit 1 family inet address 192.0.2.1/24
set interfaces irb unit 2 family inet address 192.0.2.2/24
set routing-instances blue instance-type evpn
set routing-instances blue vlan-id 1
set routing-instances blue interface ae0.1
set routing-instances blue l3-interface irb.1
...
```



```

set routing-instances blue protocols evpn interface ae0.1
set routing-instances green instance-type evpn
set routing-instances green vlan-id 2
set routing-instances green interface ae0.2
set routing-instances green l3-interface irb.2
...
set routing-instances green protocols evpn interface ae0.2
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.1
set routing-instances vrf interface irb.2
...

```

PE2

```

set interfaces xe-3/0/2 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 unit 1 encapsulation vlan-bridge
set interfaces ae0 unit 1 vlan-id 1
set interfaces ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 unit 1 esi all-active
set interfaces irb unit 1 family inet address 192.0.2.1/24
set routing-instances blue instance-type evpn
set routing-instances blue vlan-id 1
set routing-instances blue interface ae0.1
set routing-instances blue routing-interface irb.1
...
set routing-instances blue protocols evpn interface ae0.1
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.1
...

```

Step-by-Step Procedure

To configure EVPN multihoming active-active on PE1:

1. Specify an Ethernet interface as a member of aggregated Ethernet interface ae0.

```

[edit interfaces]
user@switch# set xe-2/0/0 gigether-options 802.3ad ae0

```

2. Configure aggregated Ethernet interface ae0 to simultaneously transmit 802.1Q VLAN single-tag and dual-tag frames and to support different types of Ethernet encapsulation at the logical interface level.

```
[edit interfaces]
user@switch# set ae0 flexible-vlan-tagging
user@switch# set ae0 encapsulation flexible-ethernet-services
```

3. On aggregated Ethernet interface ae0, configure logical interfaces ae0.1 and ae0.2. Configure the logical interfaces to use VLAN bridge encapsulation, and map the logical interfaces to VLANs 1 and 2, respectively. Also, assign an ESI to the logical interfaces, and enable EVPN multihoming active-active.

```
[edit interfaces]
user@switch# set ae0 unit 1 encapsulation vlan-bridge
user@switch# set ae0 unit 1 vlan-id 1
user@switch# set ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
user@switch# set ae0 unit 1 esi all-active
user@switch# set ae0 unit 2 encapsulation vlan-bridge
user@switch# set ae0 unit 2 vlan-id 2
user@switch# set ae0 unit 2 esi 00:22:22:33:44:55:66:77:88:99
user@switch# set ae0 unit 2 esi all-active
```

4. Configure IRB interfaces irb.1 and irb.2, and assign an IP address to each interface.

```
[edit interfaces]
set interfaces irb unit 1 family inet address 192.0.2.1/24

set interfaces irb unit 2 family inet address 192.0.2.2/24
```

5. Configure an EVPN routing instance named blue. Map the routing instance to VLAN 1, logical interface ae0.1, and IRB interface irb.1. Configure logical interface ae0.1 for the EVPN routing instance.

```
[edit routing-instances]
user@switch# set blue instance-type evpn
user@switch# set blue vlan-id 1
user@switch# set blue interface ae0.1
```

```

user@switch# set blue l3-interface irb.1
user@switch# set blue protocols evpn interface ae0.1

```

6. Configure an EVPN routing instance named green. Map the routing instance to VLAN 2, logical interface ae0.2, and IRB interface irb.2. Configure logical interface ae0.2 for the EVPN routing instance.

```

[edit routing-instances]
user@switch# set green instance-type evpn
user@switch# set green vlan-id 2
user@switch# set green interface ae0.2
user@switch# set green l3-interface irb.2
user@switch# set green protocols evpn interface ae0.2

```

7. Configure a VRF routing instance, and add IRB interfaces irb.1 and irb.2 to the routing instance.

```

[edit routing-instances]
set vrf instance-type vrf
set vrf interface irb.1
set vrf interface irb.2

```

Step-by-Step Procedure

To configure EVPN multihoming active-active on PE2:

1. Specify Ethernet interface xe-3/0/2 as a member of aggregated Ethernet interface ae0.

```

[edit interfaces]
user@router# set xe-3/0/2 gicether-options 802.3ad ae0

```

2. Configure aggregated Ethernet interface ae0 to simultaneously transmit 802.1Q VLAN single-tag and dual-tag frames and to support different types of Ethernet encapsulation at the logical interface level.

```

[edit interfaces]
user@router# set ae0 flexible-vlan-tagging
user@router# set ae0 encapsulation flexible-ethernet-services

```

3. On aggregated Ethernet interface ae0, configure logical interface ae0.1. Configure the logical interface to use VLAN bridge encapsulation, and map the logical interface to VLANs 1. Also, assign an ESI to the logical interface, and enable EVPN multihoming active-active.

```
[edit interfaces]
user@router# set ae0 unit 1 encapsulation vlan-bridge
user@router# set ae0 unit 1 vlan-id 1
user@router# set ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
user@router# set ae0 unit 1 esi all-active
```

4. Configure IRB interface irb.1, and assign an IP address to the interface.

```
[edit interfaces]
user@router# set irb unit 1 family inet address 192.0.2.1/24
```

5. Configure an EVPN routing instance named blue. Map the routing instance to VLAN 1, logical interface ae0.1, and IRB interface irb.1. Configure logical interface ae0.1 for the EVPN routing instance.

```
[edit routing-instances]
user@router# set blue instance-type evpn
user@router# set blue vlan-id 1
user@router# set blue interface ae0.1
user@router# set blue routing-interface irb.1
user@router# set blue protocols evpn interface ae0.1
```

6. Configure a VRF routing instance, and add IRB interface irb.1 to the routing instance.

```
[edit routing-instances]
user@router# set vrf instance-type vrf
user@router# set vrf interface irb.1
```

Verification

IN THIS SECTION

 [Verifying that a Logical Interface Has Correct ESI and EVPN Multihoming Mode | 508](#)

● Verifying the EVPN Routing Instance Status | 508

Verifying that a Logical Interface Has Correct ESI and EVPN Multihoming Mode

Purpose

Verify that logical interface ae0.1 is configured with the correct ESI and EVPN multihoming mode.

Action

From operational mode, enter the `show interfaces ae0.1` command.

```
user@switch> show interfaces ae0.1
  Logical interface ae0.1 (Index 332) (SNMP ifIndex 706)
  Flags: Up SNMP-Traps 0x20004000 VLAN-Tag [ 0x8100.100 ] Encapsulation: VLAN-Bridge
  Input packets : 0
  Output packets: 0
  Protocol bridge, MTU: 1522
    Flags: Is-Primary
  Ethernet segment value: 00:11:22:33:44:55:66:77:88:99, Mode: Single-active
```

Meaning

The output shows that logical interface ae0.1 is configured with ESI 00:11:22:33:44:55:66:77:88:99 and the EVPN multihoming mode is single-active for multihoming active-standby mode. For a scenario with multihoming active-active mode, the output would display all-active.

Verifying the EVPN Routing Instance Status

Purpose

Verify the status of the various elements configured in an EVPN routing instance.

Action

From operational mode, enter the `show evpn instance extensive` command.

```

user@switch> show evpn instance extensive
Instance: blue
...
Number of local interfaces: (1 up)
  Interface name  ESI                               Mode           Status    AC-Role
  ae0.1           00:11:22:33:44:55:066:77:88:99  single-active   Up        Root
Number of IRB interfaces: 1 (1 up)
  Interface name  VLAN ID  Status  L3 context
  irb.1           1        Up      vrf
Number of protect interfaces: 0
Number of bridge domains: 1
  VLAN  Domain ID  Intfs / up  IRB intf  Mode           MAC sync  IM route label  SG
sync  IM core nexthop
  1           0      0           Local switching
...
Number of ethernet segments: 1
ESI: 00:11:22:33:44:55:066:77:88:99
Status: Resolved by IFL ae0.1
Local interface: ae0.1, Status: Up/Forwarding
Number of remote PEs connected: 1
  Remote PE      MAC label  Aliasing label  Mode
  10.255.0.1     0          300928          all-active
DF Election Algorithm: MOD based
Designated forwarder: 10.255.0.1
Backup forwarder: 10.255.0.2
Last designated forwarder update: Jul 28 13:04:29
Advertised split horizon label: 300128

```

Meaning

The output shows that for the EVPN routing instance named `blue`, the logical interface `ae0.1` and IRB interface `irb.1` are up and running. It also shows that VLAN 1 and Ethernet segment `00:11:22:33:44:55:066:77:88:99` are properly mapped to the routing instance. It also shows the status of the DF election.

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
15.1F6	Starting with Junos OS Releases 15.1F6 and 16.1R4 for MX Series routers and Junos OS Release 17.3R1 for EX9200 switches, you can specify an ESI on a logical interface.

RELATED DOCUMENTATION

[EVPN Multihoming Overview | 162](#)

[Configuring EVPN-MPLS Active-Standby Multihoming | 280](#)

Configuring Dynamic List Next Hop

The routing table on a remote PE has a next hop entry for Ethernet segment identifier (ESI) routes with multiple next-hop elements for multihomed PE devices. For EVPN active-active multihoming device, the ESI route points to two next hop elements.

Prior to dynamic list next hop, the routing protocol process (rpd) removed the next-hop entry for the ESI route when the link between the CE device and a multihome PE device goes down. The rpd would then create a new next hop entry for the ESI causing mass MAC route withdrawals and additions.

When you enable dynamic list next-hop, the rpd removes the affected next hop element from the dynamic list next-hop entry for the ESI route while retaining the unaffected next hop element to maintain continuity of service. Dynamic list next hop provides the benefit of reducing mass MAC route withdrawals, improving the device performance, and reducing network convergence time.

To enable the dynamic list next-hop feature, include the `dynamic-list-next-hop` statement in the `[edit routing-options forwarding-table]` hierarchy.

To disable the dynamic list next-hop feature when it is enabled, use the `delete routing-options forwarding-table dynamic-list-next-hop` statement.

To display the next-hop elements from the Routing Engine's forwarding table, use the `show route label` and `show route forwarding-table` commands.

The following sample output from the `show route label detail` command shows two indirect next hops for an ESI with the dynamic list next-hop feature enabled.


```

Label operation: Push 301632, Push 299776(top)
Label TTL action: no-prop-ttl, no-prop-ttl(top)
Load balance label: Label 301632: None; Label 299776: None;
Label element ptr: 0xb7a3720
Label parent element ptr: 0xb7a3420
Label element references: 1
Label element child references: 0
Label element lsp id: 0
State: <Active Int>
Age: 1:18
Validation State: unverified
Task: evpn global task
Announcement bits (2): 1-KRT 2-evpn global task
AS path: I
Routing Instance blue, Route Type Egress-MAC, ESI 00:11:22:33:44:55:66:77:88:99

```

The following sample output from the `show route forwarding-table` command shows two next-hop entries for a destination with a multihomed route.

```

user@host> show route forwarding-table label 299952 extensive
MPLS:

Destination: 299952
Route type: user
Route reference: 0                      Route interface-index: 0
Multicast RPF nh index: 0
P2mpidx: 0
Flags: sent to PFE, rt nh decoupled
Next-hop type: indirect                 Index: 1048575 Reference: 2
Nexthop:
Next-hop type: composite                 Index: 601      Reference: 2
Next-hop type: indirect                 Index: 1048574 Reference: 3
Nexthop: 1.0.0.4
Next-hop type: Push 301632, Push 299776(top) Index: 600 Reference: 2
Load Balance Label: None
Next-hop interface: ge-0/0/1.0
Next-hop type: indirect                 Index: 1048577 Reference: 3
Nexthop: 1.0.0.4
Next-hop type: Push 301344, Push 299792(top) Index: 619 Reference: 2
Load Balance Label: None
Next-hop interface: ge-0/0/1.0

```

The following sample shows the `show route forwarding-table` command output after one of the PE devices has been disabled. It shows one next-hop element and one empty next-hop element.

```
user@host> show route forwarding-table label 299952 extensive
Routing table: default.mpls [Index 0]
MPLS:

Destination: 299952
Route type: user
Route reference: 0                      Route interface-index: 0
Multicast RPF nh index: 0
P2mpidx: 0
Flags: sent to PFE, rt nh decoupled
Next-hop type: indirect                 Index: 1048575 Reference: 2
Nexthop:
Next-hop type: composite                Index: 601      Reference: 2
Next-hop type: indirect                 Index: 1048577 Reference: 3
Nexthop: 1.0.0.4
Next-hop type: Push 301344, Push 299792(top) Index: 619 Reference: 2
Load Balance Label: None
Next-hop interface: ge-0/0/1.0
```

Link States and Network Isolation Conditions in EVPN Networks

IN THIS CHAPTER

- [Understanding When to Disable EVPN-VXLAN Core Isolation | 514](#)
- [Backup Liveness Detection on EVPN Dual Homing | 517](#)
- [Determine IRB Interface State Changes from Local L2 Interface or Remote Connectivity Status in EVPN Fabrics | 524](#)
- [Layer 2 Interface Status Tracking and Shutdown Actions for EVPN Core Isolation Conditions | 532](#)
- [EVPN Maintenance Mode for Multihomed Leaf Isolation | 537](#)
- [Uplink Protection for Network Isolation | 546](#)

Understanding When to Disable EVPN-VXLAN Core Isolation

IN THIS SECTION

- [Use Case 1: Example of When to Use the Core Isolation Feature | 515](#)
- [Use Case 2: Example of When to Disable the Core Isolation Feature | 515](#)

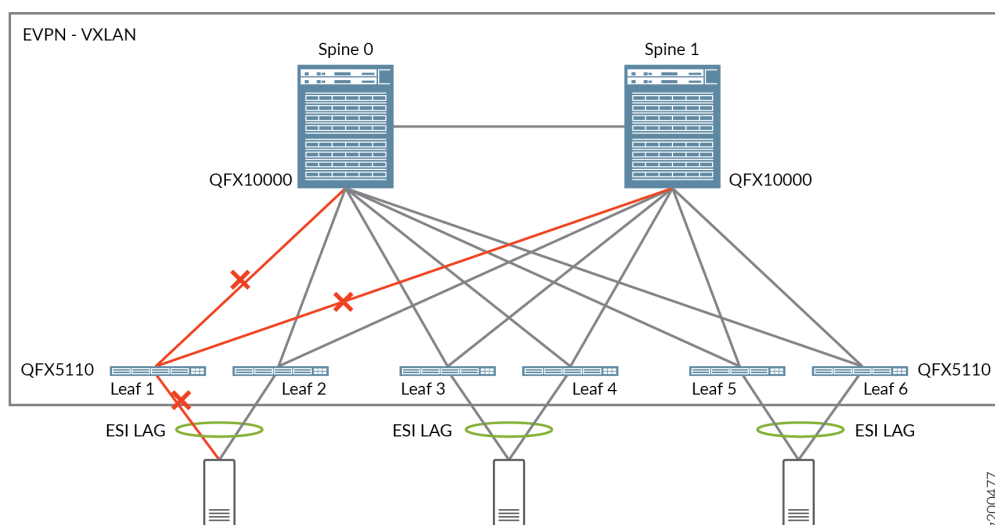
By default, spine and leaf devices in an EVPN network implement the core isolation feature. If one of these devices loses all of its EVPN BGP peering sessions, the core isolation feature, working in conjunction with Link Aggregation Control Protocol (LACP), automatically brings down all Layer 2 Ethernet Segment Identifier (ESI) link aggregation group (LAG) interfaces on the device.

In some situations, the core isolation feature produces a favorable outcome. However, in other situations, the feature produces an undesired outcome, which you can prevent by disabling the feature. The next sections describe an example situation in each case.

Use Case 1: Example of When to Use the Core Isolation Feature

Figure 34 on page 515 displays a topology in which two QFX10000 switches act as spine devices that form an EVPN-VXLAN core. In this topology, six QFX5110 switches that act as leaf devices are multihomed in active-active mode to the spine devices, and in turn, each server is multihomed through ESI-LAG interfaces to two leaf devices.

Figure 34: EVPN-VXLAN Core Isolation Use Case



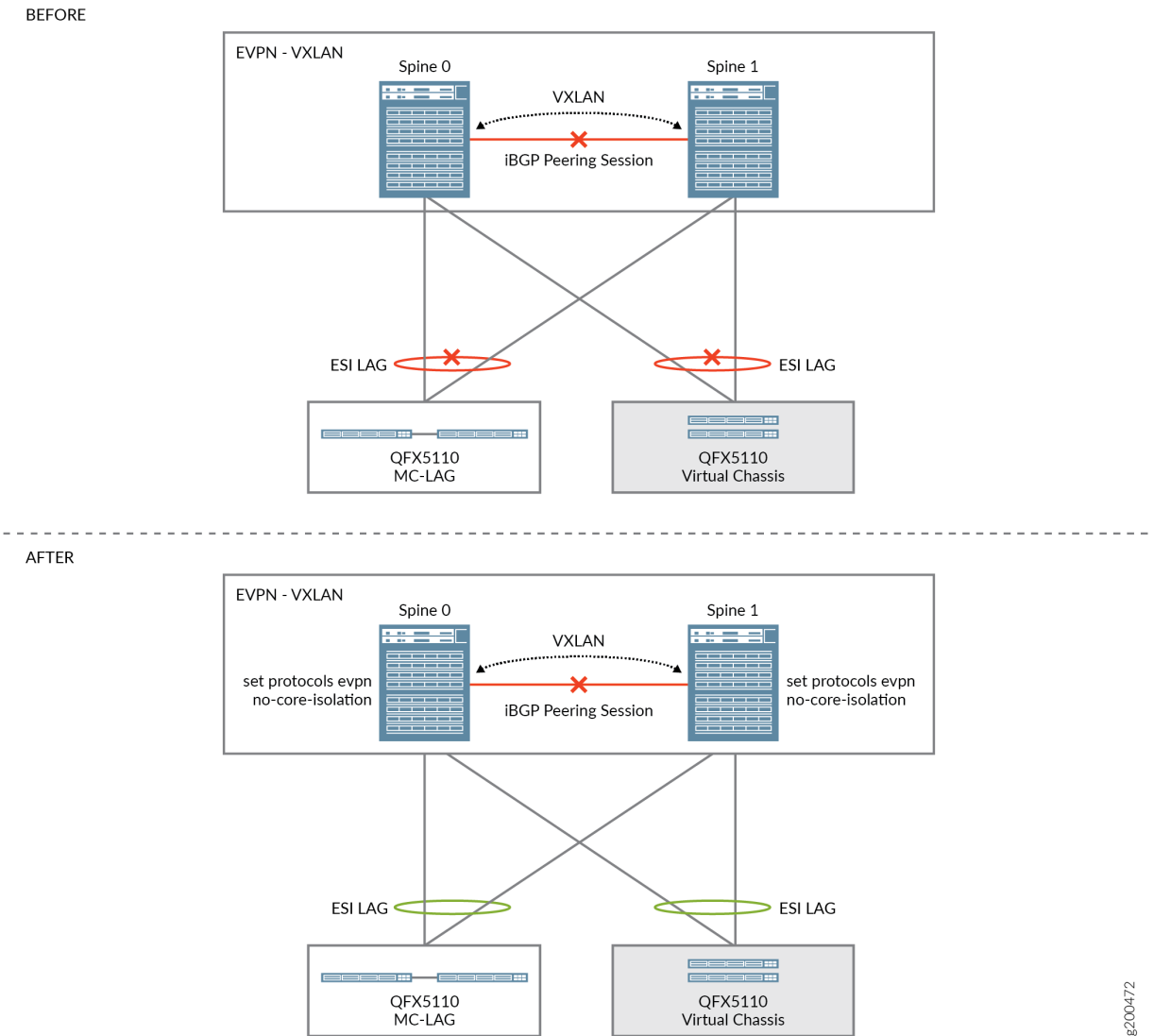
If the links between Leaf 1 and the two spine devices go down, the BGP peering sessions established over the links also go down. With the core isolation feature enabled by default, LACP sets the server-facing interface on Leaf 1 to standby mode, which blocks all traffic from the server. In this situation, the default implementation of the core isolation feature provides the following benefits:

- With the links from Leaf 1 to both spine devices down, it does not make sense for the server to continue forwarding traffic to Leaf 1.
- Traffic from the server is diverted to Leaf 2 until the links between Leaf 1 and the two spine devices are up again.

Use Case 2: Example of When to Disable the Core Isolation Feature

The topology shown in Figure 35 on page 516 is migrating from multichassis link aggregation (MC-LAG) and Virtual Chassis environments to an EVPN-VXLAN environment. In this topology, the only EVPN-VXLAN components are two QFX10000 switches that act as spine devices. The QFX5110 switches that act as leaf (MC-LAG and Virtual Chassis) devices are multihomed in active-active mode through ESI-LAG interfaces to the spine devices.

Figure 35: EVPN No Core Isolation Use Case



If the link between Spine 0 and Spine 1 goes down, the last established BGP peering session also goes down. With the core isolation feature enabled by default, LACP sets the leaf-facing interfaces on Spines 0 and 1 to standby mode, which causes data traffic to and from both leaf devices to be dropped. With the core isolation feature implemented at the leaf device level, traffic within the data center would essentially be halted, which is an undesired outcome.

In cases like this, you can set `no-core-isolation` at the `[edit protocols evpn]` configuration hierarchy level on each spine device to disable the core isolation feature. See the AFTER illustration in [Figure 35 on page 516](#). This statement is available only at the global level, so it applies to either all EVPN routing instances or the default switch instance on devices that don't have multiple routing instances.

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
17.3R3	Starting with Junos OS Release 17.3R3, you can set the no-core-isolation configuration statement at the [edit protocols evpn] hierarchy level on spine devices in the fabric to disable the core isolation feature.

RELATED DOCUMENTATION

| [Example: Configuring LACP for EVPN VXLAN Active-Active Multihoming](#) | [464](#)

Backup Liveness Detection on EVPN Dual Homing

IN THIS SECTION

- [EVPN Core Isolation Configurations](#) | [518](#)
- [Benefits of Backup Liveness Detection](#) | [520](#)
- [Behavior](#) | [520](#)
- [Limitations](#) | [522](#)
- [Configure EVPN Backup Liveness Detection](#) | [522](#)
- [Verifying Status](#) | [523](#)

By default, spine-and-leaf devices in an EVPN network implement the core isolation feature. When a device loses all of its EVPN BGP peering sessions it triggers the core isolation feature. Using the LACP, the core isolation feature shuts down all L2 ESI LAG interfaces.

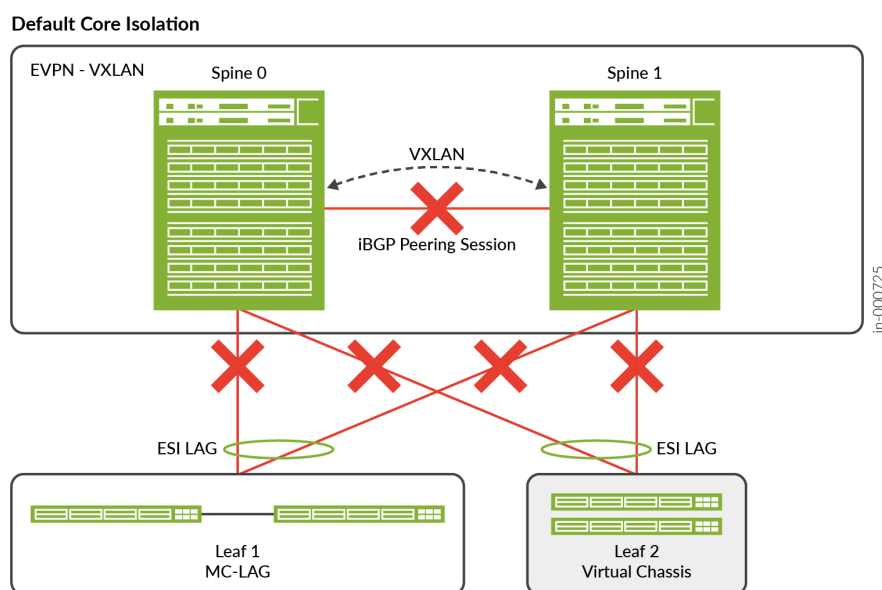
In some situations, the core isolation feature produces a favorable outcome. In other situations, the core isolation feature produces an undesired outcome, which you can prevent by disabling it. However, in a dual-homed architecture, disabling the core isolation feature on one or both peers can cause other issues. You can address these issues by configuring node liveness detection along with the default core isolation feature.

Please refer to [Feature Explorer](#) for a complete list of the products that support the [Backup liveness detection on EVPN dual homed peers](#) feature.

EVPN Core Isolation Configurations

Figure 36 on page 518 displays an EVPN-VXLAN topology with two spine devices with the default configuration for core isolation. The switches acting as leaf devices are multihomed in active/active mode through ESI-LAG interfaces to the spine devices.

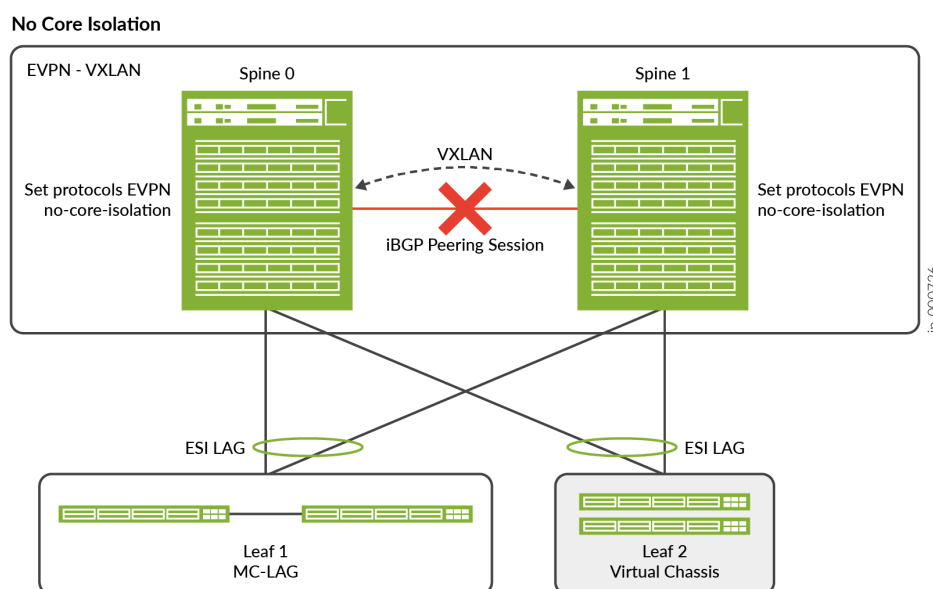
Figure 36: EVPN-VXLAN Default Core Isolation



If the link between Spine 0 and Spine 1 goes down, the last established BGP peering session also goes down. The default core isolation feature triggers LACP to set the leaf-facing interfaces on Spines 0 and 1 to standby mode. This causes data traffic to and from both leaf devices to be dropped halting traffic within the data center, which is an undesired outcome.

Figure 37 on page 519 displays an EVPN-VXLAN topology with two spine devices with `no-core-isolation` configured. The switches acting as leaf devices are multihomed in active/active mode through ESI-LAG interfaces to the spine devices.

Figure 37: EVPN-VXLAN No Core Isolation

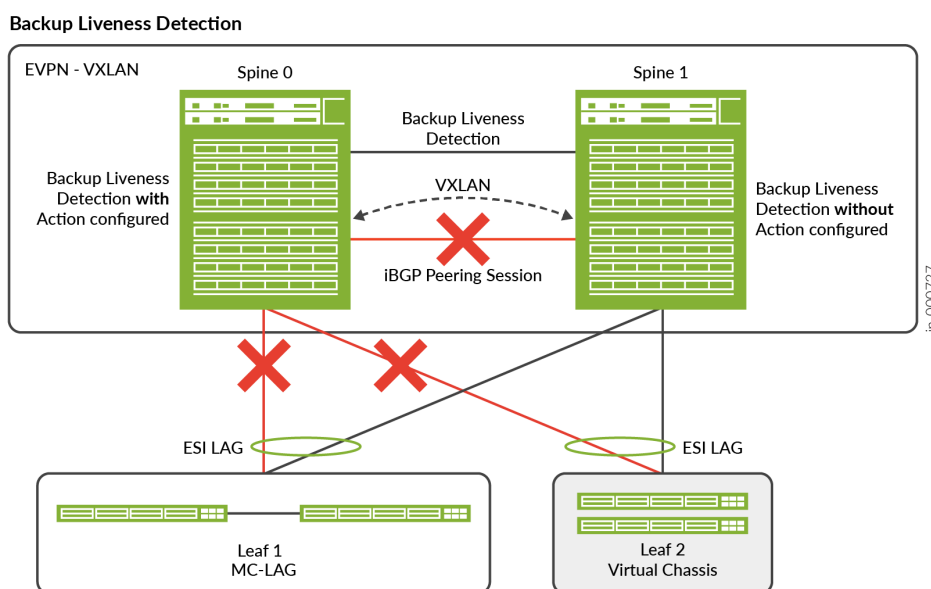


Disabling the core isolation feature could cause the following issues:

- With `no-core-isolation` configured on both Spine devices the links to the leaf devices will remain up even when BGP is down. But L3 traffic might fail because ARP/ND resolution will fail. When Spine 0 generates an ARP/ND request, the ARP/ND reply might get load balanced to Spine 1. But since BGP is down, the control plane will not synchronize resulting in ARP resolution failures.
- With `no-core-isolation` configured only on Spine 0, only Spine 1 would shut down its leaf facing interface when BGP is down. However, if Spine 0 goes down, Spine 1 still brings its leaf facing interfaces down stopping the data traffic to and from both leaf devices.

Figure 38 on page 520 displays an EVPN-VXLAN topology with two spine devices with node-detection (backup liveness detection) configured. The switches acting as leaf devices are multihomed in active/active mode through ESI-LAG interfaces to the spine devices.

Figure 38: EVPN Node Detection (Backup Liveness Detection)



The node liveness detection uses BFD configured between the EVPN peers with one side configured to take action based on the BFD and BGP status. This action ensures traffic forwarding continues through the network. The BFD liveness session can run over standard interfaces or over the management interfaces between the EVPN BGP peer devices. Node liveness needs to be detected even when the link between the Spines is down. So, node liveness detection should run over a separate network.



NOTE: Junos OS Evolved platforms do not support BFD over management ports.

Benefits of Backup Liveness Detection

- Can keep the leaf facing links up on one of the devices in a dual-homed environment during a core isolation event.
- Ensures internal traffic continues even with a BGP failure on the Spine devices.

Behavior

The node liveness feature behaves as follows. When the BGP session is up, both peers keep their leaf facing ESI-LAG interfaces up. When the BGP session is down but the BFD session is up. The Spine with the action configured brings its leaf facing ESI-LAG interfaces down. The Spine which has no action configured will keep its ESI-LAG facing the leaf devices up. When the BGP session and the BFD session tracking the node liveness are down, it indicates one of the Spine devices has gone down. Then the

Spine with the action configured will keep its leaf facing ESI-LAG interfaces up if it is active. When the BFD session comes back up, the Spine with action configured will again bring its leaf facing ESI-LAG interfaces down.

Table 1 summarizes the state on each of the Spine devices in the different scenarios. The actions configured will be either `laser-off` or `trigger-node-isolation`. With `laser-off` as the action the leaf facing ESI-LAG would receive a laser-off signal on core-isolation. If the action configured is `trigger-node-isolation` the I2 process will place the leaf facing ESI-LAG links in link-down state.

Table 25: Backup Liveness Behavior

BGP State	Node Liveness BFD State	On Spine with action Configured	On Spine with action Not Configured	Comments
Up -> Down	Down	ESI-LAG remains up	ESI-LAG remains up	For example, on Spine 0 both BGP and liveness are down Which means that Spine 1 is unavailable. Keep ESI LAG UP regardless of action configuration
Up -> Down	Up	Bring ESI-LAG down	ESI-LAG remains up	BGP state is down. But BFD is up. This indicates a split-brain scenario. Only one ESI LAG must be up. So shut down the ESI LAG on the device with the action configured.
Down -> Up	Down	If ESI LAG is down, bring it up.	ESI-LAG remains up	BGP status takes precedence
Down -> Up	Up	If ESI LAG is down, bring it up.	ESI-LAG remains up	BGP status takes precedence

Limitations

- Junos OS Evolved platforms do not support BFD over management ports, due to a Junos OS Evolved limitation that prevents sending protocol packets over a management port.
- EVPN node liveness detection does not support Multihop BFD. Configure node liveness detection over directly connected interfaces.
- Configuring `no-core-isolation` overrides node-detection configurations.

Configure EVPN Backup Liveness Detection

Configure the following elements to enable the backup liveness detection feature for an EVPN peer:

1. Configure *node-detection* on both EVPN peers with the action configured only on one peer.

- Peer 1 with action configured

```
[edit]
set protocols evpn node-detection next-hop interface;
set protocols evpn node-detection action (laser-off | trigger-node-isolation);
set protocols evpn node-detection bfd-liveness-detection minimum-interval interval;
set protocols evpn node-detection bfd-liveness-detection multiplier number;
set protocols evpn node-detection bfd-liveness-detection neighbor ip address;
```

- Peer 2 without action configured

```
[edit]
set protocols evpn node-detection next-hop interface;
set protocols evpn node-detection bfd-liveness-detection minimum-interval interval;
set protocols evpn node-detection bfd-liveness-detection multiplier number;
set protocols evpn node-detection bfd-liveness-detection neighbor ip address;
```

2. If the `laser-off` statement is used, then you must configure *asynchronous-notification* for the members of the ESI LAG.

```
[edit]
set interfaces interface gigether-options asynchronous-notification;
```

3. If the *trigger-node-isolation* statement is used, then you must configure *network-isolation* with the link-down action to bring the necessary interfaces down on core isolation.

```
[edit]
set protocols network-isolation group name detection hold-time up milliseconds;
set protocols network-isolation group name detection service-tracking core-isolation;
set protocols network-isolation group name service-tracking-action (link-down | lacp-oos);
```

Verifying Status

You can view the BFD from the peer devices by using the *show bfd session* command.

```
> show bfd session
```

Address	State	Interface	Detect Time	Transmit Interval	Multiplier
1.0.0.2	Up	ge-0/0/3.0	0.300	0.100	3

```

1 sessions, 1 clients
Cumulative transmit rate 10.0 pps, cumulative receive rate 10.0 pps
```

When the action set is laser-off, you can examine the system log messages for state information.

```
> show log messages | match isolat
Sep  6 17:48:55 Core1 18956[EVPN_CORE_ISOLATED]:
Sep  6 17:48:55 Core1 18956[EVPN_vmm _ISOLATED]:
Sep  6 17:48:59 Core1 18956[EVPN_CORE_ISOLATED]:
Sep  6 17:48:59 Core1 18956[EVPN_CORE_ISOLATED]:
Sep  6 17:49:55 Core1 18956[EVPN_CORE_ISOLATION_CLEARED]:
Sep  6 17:50:05 Core1 18956[EVPN_CORE_ISOLATION_CLEARED]:
Sep  6 17:50:05 Core1 18956[EVPN_CORE_ISOLATION_CLEARED]:
```

RELATED DOCUMENTATION

[Understanding When to Disable EVPN-VXLAN Core Isolation | 514](#)

[Layer 2 Interface Status Tracking and Shutdown Actions for EVPN Core Isolation Conditions | 532](#)

[Understanding How BFD Detects Network Failures](#)

Determine IRB Interface State Changes from Local L2 Interface or Remote Connectivity Status in EVPN Fabrics

IN THIS SECTION

- [Factors to Determine the State of an IRB Interface | 524](#)
- [Set Local L2 Interface and Remote Device Reachability Status Used to Determine the IRB Interface State | 525](#)
- [Set Network Isolation Status Parameters Used to Determine the IRB Interface State | 529](#)

To provide BGP-enabled services in a reliable way, the provider edge (PE) devices in an EVPN network need to detect when they experience network isolation conditions, and update interface statuses accordingly.

You can ensure that provider edge (PE) devices in an EVPN fabric consider both of the following factors when determining the state of a Layer 3 (L3) integrated routing and bridging (IRB) interface:

- The associated local L2 interface states
- The remote PE device reachability

Factors to Determine the State of an IRB Interface

In an integrated routing and bridging environment, when you allocate L3 interfaces, those interfaces encompass an L2 domain (also called a bridging domain or bridge domain). The L2 domain might span multiple physical or logical ports in your configuration. The device usually determines an L3 interface's state (up or down) based on the state of the corresponding L2 domain. The device sets the L2 domain state based on the status of the ports in the L2 domain, as follows:

- The L2 domain is up if the device detects that at least one of the associated ports is up.
- The L2 domain is down if the device detects that all of the associated ports are down.

When the L2 domain state changes, the device needs to update the operational state of the L3 interface too so the device can reflect the state of the upper layer protocols accordingly.

The EVPN protocol provides an integrated control plane with different data plane options (such as VXLAN or MPLS). In an EVPN environment:

- L2 domains can span multiple PEs in a data center and span across data centers.

- A device can support both L2 and L3 services over the same interfaces.
- A bridge domain or VLAN might include a combination of local physical interfaces and, in the case of EVPN-VXLAN fabrics, remote VXLAN tunnel endpoints (VTEPs).

As a result, PE devices in an EVPN fabric should consider the following factors when determining the state of IRB interfaces that are associated with a bridge domain or a VLAN in an EVPN instance (EVI):

- The states of the underlying local L2 ports or interfaces.
- Remote provider edge (PE) device reachability, based on the availability of routes to remote VTEPs and the network isolation state of the bridge domain or the EVI.

A network isolation condition in an EVPN network is similar to a *core isolation condition* (isolation from the EVPN core network). When a device detects a core isolation condition, it implements default actions to bring down the L2 member interfaces in affected Ethernet Segment Identifier (ESI) link aggregation groups (LAGs).

In contrast, with network isolation conditions, you can configure a network isolation profile and attach that profile to a bridge domain, VLAN, or routing instance. The profile defines the parameters you want the device to use to detect network isolation condition changes that affect the associated IRB interfaces. A core isolation action can affect the network isolation state of the IRB interfaces if the action brings the underlying L2 interfaces up or down.



NOTE: For more information on default core isolation behavior, see ["Understanding When to Disable EVPN-VXLAN Core Isolation"](#) on page 514.

Another factor that indicates a bridge domain, VLAN, or EVI is isolated in an EVPN-VXLAN network is when the device doesn't have a route to a remote VTEP. When the device sends traffic on a bridge domain or VLAN, it also needs to send the traffic on the VXLAN tunnels to the remote VTEPs in the bridge domain or VLAN.

The next sections describe the parameters you can customize so the device uses these factors to determine the state of an IRB interface with EVPN-VXLAN.

Set Local L2 Interface and Remote Device Reachability Status Used to Determine the IRB Interface State

PE devices can use the status of associated local L2 interfaces or remote provider edge (PE) device reachability as factors to determine the state of an IRB interface for a bridge domain or an EVPN instance (EVI). We use the network isolation status of the bridge domain or EVI to determine remote device reachability.

To specify the L2 interface and remote reachability factors the device uses to compute the IRB interface state, configure the `interface-state` statement at the `[edit interfaces irb unit n]` hierarchy.

Specify one of the following options:

- `local`—Use the status of the associated local L2 interfaces:
 - The interface is up if at least one local L2 interface is up.
 - The interface is down if none of the local L2 interfaces are up.
- `remote`—Use remote device reachability (network isolation) status of the bridge domain or EVI:
 - The interface is up if the bridge domain or EVI is not in a network isolation state.
 - The interface is down if the bridge domain or EVI is in a network isolation state.
- `local-remote`—Use a combination of the status of the local L2 interfaces and remote reachability status:
 - The interface is up if at least one local L2 interface is up *or* the bridge domain is not in a network isolation state.
 - The interface is down if no local L2 interfaces are up *and* the bridge domain is in a network isolation state.

You can also customize the minimum number of associated links that need to be up when the device computes whether the L3 interface is up:

- `local-count`—Minimum number of local L2 links that must be up
- `vtep-count`—Minimum number of remote VTEP links that must be up

The next sections show a simple example configuration and how to verify those settings:

Sample Configuration—Interface State Parameters

In the following sample configuration, the device uses both local and remote factors to determine the interface state of the IRB interface `irb.100` in bridge domain (VLAN) `v100`:

```
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 family inet address 10.0.1.11/24 virtual-gateway-address 10.0.1.254
set interfaces irb unit 100 family inet6 address 2001:db8:192:100:1:11/112 virtual-gateway-
address 2001:db8:192:100:1:254
set interfaces irb unit 100 interface-state local-remote
set interfaces irb unit 100 virtual-gateway-v4-mac 40:00:10:11:11:00
set interfaces irb unit 100 virtual-gateway-v6-mac 50:00:10:11:11:00
```

Verify IRB Interface State

To verify the state of the IRB interface irb.100 from the sample configuration in ["Sample Configuration—Interface State Parameters"](#) on page 526, enter the following CLI commands:

1. These commands show that the device determines the IRB interface state for bridge domain (VLAN) v100 using both local and remote factors (Flags = LR). The device detected a network isolation condition, so the IRB interface Link state is down.

```

user@leaf01> show bridge domain v100

Routing instance      Bridge domain      VLAN ID      Interfaces
evpnA                 v100              100          ae0.0

user@leaf01> show interfaces irb.100 terse

Interface      Admin Link Proto      Local      Remote
irb.100        up    down inet      10.0.1.11/24
                                   10.0.1.254/24
                                   inet6      2001:db8:192:100:1:11/112
                                   2001:db8:192:100:1:254/112
                                   fe80::2e6b:f500:644f:3ef0/64
                                   multiservice

user@leaf01> show l2-learning interface-state bridge-domain v100
Information for IRB interface state:

Flags ( L - Local, R - Remote, LR - Local-Remote, LI - Local-interface)

Interface      Bridging      Network  Flags      Hold-time      Hold-time      Loopback
Domain          Isolation      Up          Down          Interface
irb.100        v100          None      LR          0              0

user@leaf01> show l2-learning interface-state bridge-domain v100 extensive
Information for IRB interface state:

Name: irb.100          IRB state: Down      IRB Down Reason: local and vtep interface
below count
Routing Instance: evpnA
Bridge Domain: v100
Network Isolation state: isolated
Number of local Interface in UP state : 0

```



```

Number of local Interface in DOWN state : 0
Number of VTEP Interface in UP state : 0
Number of VTEP Interface in DOWN state : 0

```



NOTE: The Network Isolation field in the show l2-learning interface-state bridge-domain command output shows the names of any network isolation groups you have assigned to the IRB interfaces associated with the bridge domain (VLAN). The field displays None for an interface if you haven't applied any network isolation group to the corresponding bridge domain (VLAN). Don't mistake the value None in that field to mean the device didn't detect a network isolation condition.

See ["Verify Interface State with Network Isolation Link Tracking "](#) on page 530 for sample output from the show l2-learning interface-state bridge-domain command that displays a network isolation group in that field instead of None.

2. These commands show that the device's VTEP interfaces for bridge domain (VLAN) v100 are up and the device is not in a network isolation state, so the IRB interface Link state is up.

```
user@leaf01> show bridge domain
```

Routing instance	Bridge domain	VLAN ID	Interfaces
evpnA	v100	100	ae0.0 esi.903 vtep.32769 vtep.32770 vtep.32771 vtep.32772 vtep.32773 vtep.32774 vtep.32775 vtep.32776 vtep.32777

```

.
.
.

```

```

user@leaf01> show l2-learning interface-state bridge-domain v100 extensive
Information for IRB interface state:

```

```

Name: irb.100          IRB state: Up      IRB Down Reason: none
Routing Instance: evpnA
Bridge Domain: v100
Network Isolation state: not-isolated
Number of local Interface in UP state : 1
Number of local Interface in DOWN state : 0
Number of VTEP Interface in UP state : 9
Number of VTEP Interface in DOWN state : 0

```

```
user@leaf01> show interfaces irb.100 terse
```

Interface	Admin	Link	Proto	Local	Remote
irb.100	up	up	inet	10.0.1.11/24	
				10.0.1.254/24	
			inet6	2001:db8:192:100:1:11/112	
				2001:db8:192:100:1:254/112	
				fe80::2e6b:f500:644f:3ef0/64	
			multiservice		

Set Network Isolation Status Parameters Used to Determine the IRB Interface State

PE devices in an EVPN fabric can consider the network isolation status of a bridge domain or an EVI to decide whether the remote provider edge (PE) devices are currently reachable.

To customize the parameters that determine when a bridge domain or an EVI is in a network isolation state:

1. Define a network isolation profile. To do this, configure the network-isolation group *group-name* statement at the [edit protocols] hierarchy level.

When you create a network isolation group, you can customize parameters such as the following:

- Set hold times for network isolation condition changes (up or down):

After detecting the network isolation status has changed, the device delays for the hold time before acting on the change.

- Track the status of specified logical L3 uplink interfaces to detect a network isolation status change.

To do this, configure the detection link-tracking stanza at the [edit protocols network-isolation group *group-name*]. Include the name of an L3 interface you want to track. You can customize the minimum number of those links that must be up for the device to record the IRB interface state is up.

See *network-isolation* for all of the parameters you can define in a network isolation group.



NOTE: The network isolation group configuration detection stanza also has a service-tracking stanza. The service-tracking options enable the device to track L2 interface status and perform a configured service-tracking-action for L2 interfaces upon detecting a core isolation condition. You can only configure either the link-tracking options or the service-tracking options in a particular network isolation group and assign that as a network isolation profile. You can't configure both options in the same network isolation group.

2. Assign the network isolation group as a network isolation profile to an EVI, a bridge domain, or a VLAN using the `network-isolation-profile group group-name` statement at one of these hierarchy levels:

- [edit routing-instance *instance-name* switch-options]
- [edit routing-instances *name* bridge-domains *name* bridge-options]
- [edit routing-instances *name* vlans *name* switch-options]
- [edit switch-options]
- [edit bridge-domain *name* bridge-options]
- [edit vlans *name* switch-options]

The next sections show a simple example configuration and how to verify those settings:

Sample Configuration—Network Isolation Group Parameters

The following sample configuration defines a network isolation group `grp-v100` with the link-tracking option, and applies that group as a network isolation profile to bridge domain (VLAN) `v100`. As a result, the device tracks the state of interface `ge-0/0/0.0` as a factor to determine the network isolation status of bridge-domain (VLAN) `v100`:

```
set interfaces ge-0/0/0 unit 0 family inet address 192.168.23.34
set protocols network-isolation group grp-v100 detection link-tracking interface ge-0/0/0.0

set routing-instances evpnA bridge-domains v100 bridge-options network-isolation-profile group
grp-v100
```

Verify Interface State with Network Isolation Link Tracking

To verify the state of the IRB interface `irb.100` from the sample configuration in ["Sample Configuration—Network Isolation Group Parameters" on page 530](#), enter the following CLI commands:

1. Use the `show l2-learning interface-state` command to view the network isolation detection parameters assigned to the IRB interfaces on the device. The output here shows the device determines the IRB interface state for bridge domain (VLAN) v100 using both local and remote factors (Flags = LR). The Network Isolation field shows that you applied the network isolation group `grp-v100` to bridge domain (VLAN) v100. As a result, the device uses the parameters from `grp-v100` to determine the network isolation status of v100 and the state of `irb.100`.

```
user@leaf01> show l2-learning interface-state
Information for IRB interface state:

Flags ( L - Local, R - Remote, LR - Local-Remote, LI - Local-interface)

Interface      Bridging      Network  Flags    Hold-time  Hold-time  Loopback
                Domain        Isolation Up        Down       Interface
irb.100        v100          grp-v100 LR        0          0
```

2. View the status of the dependent link (`ge-0/0/0.0`) from the link-tracking stanza in the isolation group `grp-v100`, and the resulting status of the IRB interface. Both are up in this case.

```
user@leaf01> show interfaces ge-0/0/0.0 terse
Interface      Admin Link Proto  Local          Remote
ge-0/0/0.0     up    up    inet    192.168.23.34  --> 0/0
               multiservice

user@leaf01> show interfaces irb.100 terse
Interface      Admin Link Proto  Local          Remote
irb.100        up    up    inet    10.0.1.11/24
               10.0.1.254/24
               inet6    2001:db8:192:100:1:11/112
               2001:db8:192:100:1:254/112
               fe80::2e6b:f500:644f:3ef0/64
               multiservice
```

RELATED DOCUMENTATION

detection (Network Isolation Groups)

interface-state

link-tracking (Network Isolation Groups)

network-isolation

network-isolation-profile

Layer 2 Interface Status Tracking and Shutdown Actions for EVPN Core Isolation Conditions

IN THIS SECTION

- [Benefits | 532](#)
- [Overview | 533](#)
- [Behavior and Limitations | 534](#)
- [Configure Core Isolation Service Tracking and Actions for an L2 Interface | 535](#)

You can configure a device in an EVPN network to track the status of Layer 2 (L2) device interfaces, detect EVPN core isolation conditions, and take action to shut down the associated interfaces.

Starting in Junos OS and Junos OS Evolved Release 23.2R1, you can configure a provider edge (PE) device in an EVPN network to:

- Enable L2 interface service tracking to detect when the device becomes isolated from the EVPN core (a *core isolation* condition).
- Immediately take action to perform a hard shutdown or set Link Aggregation Control Protocol (LACP) out-of-sync state on the associated interface or interfaces to single-homed or multihomed customer edge (CE) devices.

The interfaces might be single physical links, single-homed link aggregation group (LAG) interface bundles, or EVPN segment identifier (ESI) LAG interface bundles.

- Set a timer to delay before bringing associated interfaces up again when the device is no longer isolated from the core.

Benefits

- Provides configurable settings to help avoid traffic loss for single-homed or multihomed CE devices during route convergence when a PE device becomes isolated from the EVPN core.

Overview

In EVPN networks, a PE device can become isolated from the EVPN core. That device might have single-homed or multihomed CE devices connected to it.

When the PE device detects it is in a core isolation condition, usually the LACP protocol sets the ESI LAG interfaces to LACP out-of-sync state. That state should trigger the CE device to bring down the interface on the CE side as well. However, the CE device might not bring down the link immediately. Slower LACP responsiveness on the CE side can cause a delay in route convergence and, as a result, some traffic loss. To similarly avoid traffic loss, single-homed CE devices with active-backup link bundles to a PE device also need an immediate action on the active link to trigger a switchover to the backup link.

With this feature, to avoid traffic loss in these core isolation situations, you can configure an immediate action the PE device takes on the associated interface or interfaces to the attached CE devices. You can set the action to either bring an interface down immediately with a hard shutdown, or continue to rely on LACP out-of-sync signaling.

When the PE device recovers its BGP sessions to the EVPN core and is no longer isolated, the LACP protocol on the device immediately brings the associated ESI LAG interfaces up. However, the PE device might still be synchronizing routes from its multihoming peer PE devices. In that case, the multihomed CE device can also lose traffic.

As a result, with this feature you can also set a delay time before the device brings the interface up again after the device becomes reconnected to the core.

To set up core isolation service tracking, you configure a network isolation group at the [edit protocols network-isolation group *network-isolation-group-name*] hierarchy level. In the network isolation group configuration, set options in the detection and service-tracking-action stanzas, as follows (see the highlighted statements):

```
[edit]
protocols {
  network-isolation {
    group network-isolation-group-name {
      detection {
        hold-time {
          up ms;
        }
        service-tracking {
          core-isolation;
        }
      }
    }
    service-tracking-action {
```

```

        lacp-out-of-sync | link-down;
    }
}
}
}

```



NOTE: The configuration hierarchy above shows only the options relevant to the core isolation service tracking feature. The [edit protocols network-isolation group *network-isolation-group-name*] hierarchy includes other options for other features that we don't cover here.

You assign the network isolation group as a network isolation profile to the interface or interfaces the device should shut down upon detecting a core isolation condition. To do this, you configure the `network-isolation-profile network-isolation-group-name` statement at the [edit interfaces *interface-name*] hierarchy.

You can enable core isolation service tracking for the following types of Layer 2 (L2) interfaces:

- A logical single link or aggregated Ethernet (AE) link bundle (LAG) to a single-homed customer edge device in the network.
- An Ethernet segment identifier (ESI) LAG interface to a multihomed customer edge device in the network.
- A physical interface.

See ["Configure Core Isolation Service Tracking and Actions for an L2 Interface" on page 535](#) for the steps to configure this feature.

Behavior and Limitations

Note the following runtime behaviors and limitations with this feature:

- The network isolation group configuration detection stanza also has a link-tracking option to determine the state of a Layer 3 (L3) integrated routing and bridging (IRB) interface after network isolation conditions change. (See ["Set Network Isolation Status Parameters Used to Determine the IRB Interface State" on page 529](#)).

In contrast, the service-tracking option tracks L2 interface status and performs the configured service-tracking-action on those L2 interfaces.

You can only configure either the link-tracking option or the service-tracking option in a particular network isolation group and assign that as a network isolation profile. You can't configure both options in the same network isolation group.

- When you configure this feature on a LAG or ESI LAG interface bundle, the device takes the configured action on each of its member links.
- If the device is already in a core isolation state and you configure core isolation service tracking with action `link-down` for a new interface, the device immediately brings that interface down.
- The network isolation group configuration includes a hold timer (`hold-time up`) to delay before bringing the interfaces up when the core isolation state is resolved.

The device maintains only one `hold-time up` timer for each network isolation group you configure. As a result, if you associate the network isolation group with a new interface while the timer is already running, the device brings up all of the interfaces in the group when the timer expires. The device doesn't reset the timer when you add another interface to the group.



NOTE: We don't support the `hold-time down` timer option in the detection stanza with this core isolation service tracking feature.

- If you enable core isolation service tracking when you also have configured the `no-core-isolation` option, the device will not bring any interfaces down upon detecting a core isolation condition. See ["Understanding When to Disable EVPN-VXLAN Core Isolation" on page 514](#) for details on using that option.
- If you delete a network isolation group profile configuration from an interface, the device brings the interface up again immediately if it had previously brought the interface down due to a core isolation condition.

Configure Core Isolation Service Tracking and Actions for an L2 Interface

Configure the following elements to enable the core isolation service tracking feature for an interface:

1. Configure the network isolation group:

- a. Enable core isolation service tracking in the detection stanza.

```
set protocols network-isolation group network-isolation-group-name detection service-  
tracking core-isolation
```

- b. Set a `hold-time up` timer value in the detection stanza. The device waits this number of milliseconds before setting the interface state to *up* again upon detecting that the core isolation condition is resolved.

This feature has no default hold time, so you usually want to set a hold time greater than 0 so the device isn't excessively processing rapid status changes. Specify a value of 1 or more milliseconds.

```
set protocols network-isolation group network-isolation-group-name detection hold-time up
ms
```

- c. Set the service tracking action in the network-isolation group *network-isolation-group-name* stanza. The available core isolation interface actions are to either do a hard shutdown on the interface or interfaces, or set interface status to LACP out-of-sync.

```
set protocols network-isolation group network-isolation-group-name service-tracking-action
(link-down | lacp-out-of-sync)
```

2. Assign the network isolation group to the desired interface or interfaces using the network-isolation-profile *network-isolation-group-name* statement at the [edit interfaces *interface-name*] hierarchy level.

```
set interfaces interface-name network-isolation-profile network-isolation-group-name
```

For example, the following sample configuration stanza shows a network isolation profile assigned simply to a physical interface:

```
[edit]
interfaces {
  et-0/0/1 {
    network-isolation-profile network-isolation-group-name;
  }
}
```

and the following sample configuration stanza shows a network isolation profile assigned to an ESI-LAG interface bundle:

```
[edit]

interfaces {
  ae1 {
    esi {
      00:11:22:33:44:55:66:77:88:99;
      all-active;
    }
  }
}
```

```

        network-isolation-profile network-isolation-group-name;
    }
}

```

RELATED DOCUMENTATION

network-isolation

detection (Network Isolation Groups)

service-tracking (Network Isolation Groups)

service-tracking-action (Network Isolation Groups)

[Understanding When to Disable EVPN-VXLAN Core Isolation | 514](#)

EVPN Maintenance Mode for Multihomed Leaf Isolation

SUMMARY

EVPN Maintenance Mode enables seamless maintenance and upgrades in EVPN environments by isolating multihomed ERB Leaf nodes, rerouting traffic to maintain network stability.

IN THIS SECTION

- [Benefits of Maintenance Mode CLI | 538](#)
- [Overview | 538](#)
- [EVPN Maintenance Mode Prerequisites | 539](#)
- [Command Utilization and Best Practices | 542](#)
- [Upgrade an ERB Leaf Node | 544](#)

EVPN maintenance mode streamlines maintenance and upgrade operations in EVPN environments by minimizing traffic disruption. By isolating devices in a controlled manner, service interruptions during upgrades can be prevented. This feature operates seamlessly with multihomed edge-routed bridging (ERB) leaf nodes, rerouting traffic to alternate nodes to maintain network stability. It includes pre-checks and validations to ensure multihomed configurations, service tracking, and non-revertive designated forwarder (DF) settings are correctly configured. Non-revertive DF and egress link protection (ELP) enhance resilience, prevent traffic loss and improve convergence times, while network isolation (NISO) profiles aid in maintaining service continuity through core isolation management. You use the command-line interface (CLI) commands to configure, validate, and monitor maintenance mode, and to ensure that all pre-check requirements are met before network changes.

Benefits of Maintenance Mode CLI

- Minimizes network disruption by isolating multihomed ERB leaf nodes during upgrades, ensuring continued network operations.
- Enhances system resilience with non-revertive DF election, maintaining stability even during node outages.
- Improves route convergence times through ELP, ensuring efficient traffic rerouting and reduced downtime.
- Supports multihoming configurations, providing robust connectivity options that maintain network performance during maintenance activities.
- Ensures compatibility and optimal performance by adhering to recommended pre-check and post-check procedures, reducing the risk of upgrade failures.
- Facilitates service continuity and stability during maintenance using NISO profiles to manage core isolation and interface actions effectively.

Overview

EVPN maintenance mode is a sophisticated feature that facilitates seamless maintenance and upgrades in EVPN networks by isolating multihomed ERB leaf nodes and shutting down access-side interfaces. Activating this mode minimizes traffic disruption by ensuring effective traffic rerouting, maintaining network performance during maintenance and system upgrades. You initiate maintenance mode using the CLI command `set protocols evpn maintenance-mode erb-leaf action-type (validate-only|validate-and-set|force)`. The options `validate-only`, `validate-and-set`, or `force` give you precise control over the process, ensuring that pre-checks and configurations are accurate before proceeding.

You can validate configurations before and after starting maintenance mode with the `request evpn validate-maintenance-mode-pre-checks` command. Then use `show evpn maintenance-mode-status` to view results of the pre-checks and monitor the current status. If you change any maintenance mode related configurations, rerun `request evpn validate-maintenance-mode-pre-checks` to update the validation.

The egress link protection (ELP) and fast reroute (FRR) features are integral to maintaining network reliability, especially during link failures. When a link to a customer edge device fails, these features facilitate rapid route convergence, rerouting traffic swiftly to avoid significant data loss. This proactive approach ensures that during maintenance or unexpected failures, the network remains stable and operational. The non-revertive DF election further stabilizes the network by maintaining consistent designated forwarder status even after node reboots, preventing unnecessary traffic switching and ensuring a seamless transition back to normal operations. Network isolation (NISO) profiles also play a crucial role in core isolation management, vital during maintenance activities. These profiles manage interface actions and service-tracking mechanisms to ensure continuity and stability.

EVPN Maintenance Mode Prerequisites

Before initiating EVPN maintenance mode, prepare your network by ensuring out-of-band management capabilities and bandwidth availability to handle potential traffic loads when traffic is diverted to the multihomed peer. This feature does not support single-homed configurations or in-band network management during maintenance.

Adhering to operational guidelines is crucial; these include conducting pre-maintenance health checks, backing up configurations and cleaning system storage. After completing the maintenance activity verify the system and network status to confirm stability and performance.

The maintenance mode CLI feature has the following prerequisites for implementation.

- Multihomed EVPN instances with the ESI configured with `df-election-type preference non-revertive`.
- NISO profiles with `service-tracking core-isolation` and `service-tracking-action link-down` configurations.
- Access interfaces configured with `hold-time up` for better traffic convergence post maintenance.
- (EVPN-MPLS) ELP (`evpn-egress-link-protection`) configured under the `routing-options forwarding-table` hierarchy.
- (EVPN-VXLAN) FRR (`reroute-address`) configured under the `forwarding-options evpn-vxlan` hierarchy.
- Minimum software version of Junos OS 25.3R1 or later.



NOTE: You manage EVPN maintenance mode using configuration statements. When you exit maintenance mode by removing the `maintenance-mode` configurations the `hold-time` configured in the NISO groups is not triggered. Therefore, the `hold-time up` should be configured on the access interfaces.

Non-revertive DF

The non-revertive option for "[Preference-Based DF Election](#)" on [page 192](#) ensures that once a DF is elected, it will not be preempted by the previously designated DF coming back online after a failure. This non-revertive mode is key in maintaining a stable network environment and avoiding unnecessary service disruptions.

```
interfaces {
  ge-0/1/0 {
    esi {
      00:01:02:03:04:05:06:07:08:09;
```

```

single-active | all-active;
df-election-type {
    preference {
        value (0-65535);
        non-revertive;
        least;
    }
}
}
}
}
}
}
}

```

Network Isolation Groups and Service Tracking

Network isolation (NISO) groups define what to track to detect core isolation and what action to take on layer 2 interfaces. Core isolation tracking can be based on core side link tracking or service tracking.

You define NISO groups using the set protocols `network-isolation group` statement. Then you associate the group name with CE facing interfaces using the `network-isolation-profile network-isolation-group-name` statement under the edit interfaces `interface name` hierarchy. This configures those interfaces for fast switchover of traffic in event of core isolation.

The maintenance mode CLI disables access-facing interfaces by sending an EVPN maintenance-mode message to the Layer 2 Address Learning Daemon (L2ALD). This causes the interfaces connected to the multihomed CE to go into an LACP out-of-sync (OOS) state. However, the LACP interfaces may not go down right away. So, you must configure the `service-tracking core-isolation` and `service-tracking-action link-down` statements in the NISO group to bring down the maintenance node's interfaces immediately, allowing better and faster traffic convergence. Additionally, setting the interfaces `name` hold-time down to 0 ensures the interface shuts down immediately when the Maintenance Mode CLI is enabled.

You can use the interfaces `name` hold-time up statement to configure a short delay when bringing the access facing interfaces back up. This improves traffic convergence by enabling the core facing interfaces to come up and synchronize routes with the neighboring PEs first. The hold-time configured in NISO group is not triggered when removing the maintenance mode CLI configuration.

```

interfaces {
    ae1 {
        hold-time up 240000 down 0;
        esi {
            00:11:00:11:11:11:11:11:11;
            (all-active | single-active);
        }
    }
}

```

```

    }
    network-isolation-profile network isolation group name;
  }
}

protocols {
  network-isolation {
    group network isolation group name {
      detection {
        hold-time {
          up time in milliseconds;
        }
        service-tracking {
          core-isolation;
        }
      }
      service-tracking-action {
        (link-down | lacp-oos);
      }
    }
  }
}

```

ELP and FRR Configurations

You configure the `evpn-egress-link-protection` statement for EVPN-MPLS or the `reroute-address` statement for EVPN-VXLAN to enable the egress link protection fast reroute (ELP/FRR) feature on multihoming peer provider edge (PE) devices. This feature helps to minimize load-balanced traffic loss when the link from a PE device to a multihomed CE device goes down.

The ELP/FRR check passes when there is a valid EVPN-MPLS or EVPN-VXLAN style configuration as shown below.

EVPN-MPLS configuration:

```

routing-options {
  forwarding-table {
    evpn-egress-link-protection;
  }
}

```

EVPN-VXLAN configuration:

```
forwarding-options {
  evpn-vxlan {
    reroute-address {
      inet {
        re-route IP address;
      }
    }
  }
}
```

Command Utilization and Best Practices

To effectively implement the maintenance mode CLI, it is essential to understand the commands and their usage.

The request `evpn validate-maintenance-mode-pre-checks erb-leaf` command verifies that the device configuration meets the `maintenance-mode` prerequisites. This command initiates the same validation checks as the `set protocols evpn maintenance-mode erb-leaf action-type (validate-only|validate-and-set)` configuration and can be ran before, during or after configuring `maintenance-mode`. You should run this command after any configuration changes to ensure those changes are considered during re-validation. Then use the `show evpn maintenance-mode-status` command to check the status.

The `show evpn maintenance-mode-status` displays the results of the most recent validation checks and the whether the current EVPN Maintenance Status: `status` is `Under Maintenance` or `Not-under Maintenance`.

- **Under Maintenance**—Indicates the access facing interfaces are disabled/down and all EVPN T2 and T4 routes have been withdrawn and the node is ready for maintenance or upgrade.
- **Not-under Maintenance**—Indicates the node is not in `maintenance-mode`. Use the `set protocols evpn maintenance-mode erb-leaf action-type` command with the `validate-and-set` or `force` option to enter `maintenance-mode`.

You should regularly consult `show evpn maintenance-mode-status` to assess the current mode and review validation results. These insights are crucial for making informed decisions during maintenance activities.

`show evpn maintenance-mode-status`

```
user@Leaf-1> show evpn maintenance-mode-status
```

```

Pre-check Validation Status      : PASS
      ELP / FRR                  : PASS
      NISO-Profile               : PASS
      Non-revertive DF           : PASS

EVPN Maintenance Status : (Under Maintenance | Not-under Maintenance)

```

The `show evpn maintenance-mode-status extensive` output provides status details on individual EVPN instances. This output also displays untracked interfaces. These are non-EVPN interfaces that might go down during the maintenance if they have a NISO profile configured.

show evpn maintenance-mode-status extensive

```

user@Leaf-1> show evpn maintenance-mode-status extensive

Pre-check Validation Status      : PASS
      ELP / FRR                  : PASS
      NISO-Profile               : PASS
      Non-revertive DF           : PASS
EVPN Maintenance Status         : Under Maintenance

Instance: rd_pr_mac_vrf_1
      Encapsulation type: VXLAN
      FRR                    : FAIL
Instance: test
      Encapsulation type: VXLAN
      FRR                    : FAIL
Interfaces : CE-facing-Intf-Name
      ae0.0      ESI: 00:01:01:01:01:01:01:01:01
      Multihomed           : Yes (all-active)
      NISO Status          : PASS
      Non-revertive DF Status : PASS

Untracked Interfaces: ae1
                        ae2
                        ge-0/0/1

```

You configure the `set protocols evpn maintenance-mode erb-leaf action-type (validate-only|validate-and-set|force)` statement to initiate maintenance-mode. The `validate-only` option allows you to run pre-checks without altering the network, while `validate-and-set` executes the checks and transitions the node into maintenance-mode if the checks succeed. The `force` option bypasses the validation options. Use this option with caution as it can cause traffic loss if the required configurations are missing.

The validation options certify that your network is adequately configured before entering maintenance mode. These pre-checks verify multihomed configurations, enforce service tracking actions, and confirm non-revertive DF settings. By ensuring these conditions are met, the risk of service disruption during maintenance is reduced. Additionally, utilizing non-revertive DF election and ELP with fast reroute (FRR) enhances network convergence times and resilience, preventing traffic loss during maintenance operations. NISO profiles play a crucial role in core isolation management, vital during maintenance activities. These profiles manage interface actions and service-tracking mechanisms to ensure continuity and stability.



CAUTION: The maintenance mode CLI feature assumes that the configuration includes NISO profiles attached to all CE facing L2 interfaces and not to attached to any core facing interfaces. It does not validate whether interfaces with NISO profiles are CE facing or core facing. Incorrectly configured NISO profiles can cause traffic loss.



NOTE: The `set protocols evpn maintenance-mode` and the `set protocols l2-learning niso-maintenance down` commands cannot be configured at the same time.

Use the `delete protocols evpn maintenance-mode` command to remove the maintenance-mode configuration and bring the EVPN instances back online and resume normal operations. You can implement this once maintenance is completed to restart EVPN immediately. Or you can leave the maintenance-mode configured and remove it after a reboot if necessary.

For example, if you reboot the node and maintenance-mode is configured with the `force` option all the CE facing interfaces would remain down. If maintenance-mode is configured with the `validate-and-set` option it would validate the pre-checks and act accordingly.

Alternatively, if you change the maintenance-mode to use the `validate-only` option it will clear the maintenance-mode and bring the CE facing interfaces up.

Upgrade an ERB Leaf Node

In the edge-routed bridging (ERB) design the spine devices provide only IP connectivity between the leaf devices. They require no VXLAN functionality and are referred to as lean spines. The leaf devices provide connectivity to attached workloads and provide layer 2 (L2) and layer 3 (L3) VXLAN functionality in the overlay network. L2 gateways provide bridging within the same VLAN and L3 gateways handle the traffic between VLANs using integrated routing and bridging (IRB) interfaces.

You define NISO groups and attach them to layer 2 or CE facing interfaces for fast switchover of traffic in the event of core isolation. These groups define what to track to detect core isolation and what action to take on layer 2 interfaces. You can use the maintenance mode CLI to leverage the NISO groups during maintenance or upgrades to simplify isolating multihomed nodes and minimize traffic disruption.

The [Junos OS Upgrade for EVPN VXLAN Network](#) document provides detailed recommended procedures for upgrading an EVPN-VXLAN ERB leaf node. Chapter 2 covers the guidelines for planning the upgrade and includes the recommended pre-upgrade and post-upgrade health checks. Chapter 3 provides an overview and specific steps for upgrading an ERB leaf node. You can use the maintenance mode CLI to isolate the multihomed access interfaces for the steps that disable and enable the end-device facing access interfaces.

Enter EVPN Maintenance Mode

EVPN maintenance mode leverages the NISO configurations to disable the end-device facing access interfaces and divert traffic towards multihomed peer devices. This helps to isolate the node and prepare it for maintenance.

Use the following steps to enable maintenance mode.

- Use the `request evpn validate-maintenance-mode-pre-checks erb-leaf` and `show evpn maintenance-mode-status` commands to ensure the device configuration and state meets the prerequisites for EVPN maintenance mode.
- Enter maintenance mode by configuring the `set protocols evpn maintenance-mode erb-leaf action-type (validate-only|validate-and-set|force)` statement with either the `validate-and-set` or the `force` option.
 - `validate-only`—initiate the EVPN maintenance-mode pre-check process and provide the validation results.
 - `validate-and-set`—initiate the EVPN maintenance-mode pre-checks and if the validations pass then enter maintenance-mode.
 - `force`—skip the EVPN maintenance-mode pre-checks and enter maintenance-mode immediately.

Exit EVPN Maintenance Mode

You exit maintenance mode to enable the end-device facing access interfaces that maintenance mode disabled. The maintenance mode CLI configurations remain active as long as they are configured, even after a reboot. For example:

- If you configure maintenance-mode with the `validate-and-set` option, it will validate the pre-checks after rebooting and act accordingly.
- If you configure maintenance-mode with the `force` option, it will skip the pre-checks after rebooting and re-enter maintenance-mode immediately.

You can exit maintenance-mode by changing the maintenance-mode option to `validate-only` or by removing the maintenance-mode CLI configuration as follows:

- Use set protocols evpn maintenance-mode erb-leaf action-type validate-only to exit maintenance-mode but still run the validation checks.
- Use the delete protocols evpn maintenance-mode command to exit maintenance-mode by removing the configuration.



NOTE: When you exit Maintenance Mode the hold-time configured in the NISO groups is not triggered. Therefore, the hold-time up should be configured on the access interfaces.

RELATED DOCUMENTATION

[EVPN Multihoming Overview](#)

[Layer 2 Interface Status Tracking and Shutdown Actions for EVPN Core Isolation Conditions](#)

[Convergence in a Multihomed EVPN-MPLS Network](#)

[Fast Reroute for Egress Link Protection with EVPN-VXLAN Multihoming](#)

Uplink Protection for Network Isolation

SUMMARY

Uplink protection for network isolation automatically shuts down L2 interfaces when core-isolation is detected, and brings them back up when the isolation state is cleared.

IN THIS SECTION

- [Benefits of Uplink Protection for Network Isolation | 547](#)
- [Overview | 547](#)
- [Configuration Example | 548](#)

Uplink protection for network isolation ensures network stability by managing Layer 2 (L2) interfaces based on the state of Layer 3 (L3) or core-facing interfaces. This feature detects core isolation on Provider Edge (PE) devices and triggers the shutdown of associated L2 interfaces to prevent traffic loops and enhance failover for multihomed Customer Edge (CE) devices. When isolation is resolved, these interfaces are brought back up after a configurable delay to allow for protocol convergence. Key functionalities include the management of aggregate and physical interfaces, an option for simultaneous L2 interface control, and service tracking actions to optimize network operations.

Benefits of Uplink Protection for Network Isolation

- Prevents L2 traffic loops by automatically shutting down L2 interfaces connected to isolated PE devices, ensuring network stability.
- Ensures quick and reliable failover for multihomed CE devices, minimizing downtime during network isolation events.
- Facilitates seamless reconnection of interfaces using configurable hold times to ensure all routing updates are synchronized before interfaces come back online.
- Simplifies network maintenance through a dedicated CLI command that enables administrators to bring all L2 ports down or up simultaneously.
- Integrates with service tracking to automatically manage server/CE-facing L2 interfaces, optimizing network operations and maintaining service availability.

Overview

Uplink Protection for Network Isolation dynamically manages L2 interfaces based on the state of L3 interfaces facing the core network. This functionality is critical in preventing traffic loops and ensuring network stability by automatically shutting down L2 interfaces connected to isolated PE devices. When core isolation is detected, L2 interfaces are brought down and back up after the isolation state is resolved, adhering to a configurable hold timer. This hold timer allows time for L3 protocols to converge and routes to synchronize, ensuring that the network is stable before L2 interfaces become active again.

To configure Uplink Protection for Network Isolation, define network isolation groups and associate them with relevant interfaces. Key CLI commands involved in setting up this feature include defining detection hold times for network isolation (both up and down), specifying link-tracking interfaces, and associating interfaces with network isolation profiles.

For example, use the following commands to configure the network isolation group, define detection parameters, and associate interfaces with the group, ensuring that L2 interface states are managed based on L3 interface conditions:

```
set protocols network-isolation group group-name detection hold-time up milliseconds down
milliseconds
set protocols network-isolation group group-name detection link-tracking interface interface
set interfaces interface network-isolation-profile group-name
```

Additionally, the feature integrates service tracking actions to manage L2 interface states, improving network consistency. During maintenance windows, specific CLI commands can bring down or up all L2

interfaces associated with network isolation profiles, facilitating controlled and efficient maintenance operations.

For example, the following command allows you to shut down the L2 interfaces associated with a particular network isolation group during maintenance, ensuring a smooth operational workflow.

```
set protocols l2-learning niso-maintenance down group-name
```

Configuration Example

The following configuration example sets up a network isolation group named `grp-red` to monitor the L3 interface `ge-0/0/0.0` with a hold time of 1000 milliseconds for bringing interfaces up. It then associates the L2 interfaces `ae0` and `et-0/0/1` with this group.

This configuration ensures that when core isolation is detected, the specified L2 interfaces will be brought down and back up after a 1000 milliseconds hold time once the isolation state is cleared. This setup allows for route synchronization and protocol convergence, maintaining network stability and preventing traffic loops.

```
set protocols network-isolation group grp-red detection link-tracking interface ge-0/0/0.0
set protocols network-isolation group grp-red detection hold-time up 1000
set protocols network-isolation group grp-red service-tracking-action link-down
set interfaces ae0 network-isolation-profile grp-red
set interfaces et-0/0/1 network-isolation-profile grp-red
```

RELATED DOCUMENTATION

[Layer 2 Interface Status Tracking and Shutdown Actions for EVPN Core Isolation Conditions | 532](#)

niso-maintenance

network-isolation

network-isolation-profile

EVPN Proxy ARP and ARP Suppression, and NDP and NDP Suppression

IN THIS CHAPTER

- [EVPN Proxy ARP and ARP Suppression, and Proxy NDP and NDP Suppression | 549](#)
- [ARP and NDP Request with a proxy MAC address | 552](#)

EVPN Proxy ARP and ARP Suppression, and Proxy NDP and NDP Suppression

Proxy Address Resolution Protocol (ARP) and ARP suppression, and proxy Neighbor Discovery Protocol (NDP) and NDP suppression are supported as follows:

- MX Series routers and EX9200 switches
 - Starting with Junos OS Release 17.2R1, MX Series routers and EX9200 switches that function as provider edge (PE) devices in an Ethernet VPN-MPLS (EVPN-MPLS) or Layer 3 VXLAN gateways in an Ethernet VPN-Virtual Extensible LAN (EVPN-VXLAN) environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on an integrated and routing (IRB) interface.
 - Starting with Junos OS Release 17.4R2, MX Series routers and EX9200 switches support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on non-IRB interfaces. Junos OS Release 17.4R2 also introduces the ability to limit the number of media-access-control (MAC)-IP address bindings that can be learned on these Juniper Networks devices.
- QFX10000 switches
 - Starting with Junos OS Release 17.3R1, QFX10000 switches that function as Layer 3 VXLAN gateways in an EVPN-VXLAN environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on an IRB interface.
 - Starting with Junos OS Release 19.1R1, QFX10000 switches that function as Layer 2 VXLAN gateways in an EVPN-VXLAN environment support proxy ARP and ARP suppression, and proxy

NDP and NDP suppression on a non-IRB interface. You can also limit the number of MAC-IP address bindings that can be learned on these switches.

- QFX5100, QFX5200, and QFX5110 switches—Starting with Junos OS Release 18.1R1, QFX5100 and QFX5200 switches that function as Layer 2 VXLAN gateways in an EVPN-VXLAN environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on non-IRB interfaces. QFX5110 switches that function as Layer 2 or 3 VXLAN gateways in an EVPN-VXLAN environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on IRB interfaces and non-IRB interfaces. You can also limit the number of MAC-IP address bindings that can be learned on these switches.



NOTE: In Junos OS Releases you could configure the `no-arp-suppression` statement to turn off suppressing ARP packets and NDP packets. However, starting in Junos OS Release 19.1R1, that statement is hidden in the Junos CLI.

This feature reduces the flooding of ARP and NDP messages in the EVPN network, resulting in a more efficient use of core bandwidth. By default, proxy ARP and ARP suppression and proxy NDP and NDP suppression are enabled.

When limiting the number of MAC-IP address bindings that can be learned, the limit can be configured globally or for a specific routing instance, bridge domain, VLAN, or interface. After the specified limit is reached, no additional entries are added to the MAC-IP binding database. You can also specify a timeout interval for MAC-IP address bindings.

Proxy ARP and NDP snooping are enabled by default for all EVPN-MPLS or EVPN-VXLAN bridge domains and VLANs. ARP or NDP packets generated from a local customer edge (CE) device or Layer 2 VXLAN gateway are snooped. ARP and NDP packets generated from a remote PE device or Layer 3 VXLAN gateway through core-facing interfaces, however, are not snooped.

Both IRB and non-IRB interfaces configured on a PE device or a Layer 2 or 3 VXLAN gateway deliver ARP requests and NDP requests. When one of these devices receives an ARP request or NDP request, the device searches its MAC-IP address bindings database for the requested IP address. If the device finds the MAC-IP address binding in its database, it responds to the request. If the device does not find the MAC-IP address binding, it takes the following action:

- If the device is running Junos OS Releases 17.2Rx or 17.3Rx, the device swaps the source MAC address with the MAC address of the interface on which the request was received and sends the request to all interfaces.
- If the device is running Junos OS Releases 17.4R1 or later, the device leaves the source MAC address as is and sends the request to all interfaces.

Even when a PE device or a Layer 2 or 3 VXLAN gateway responds to an ARP request or NDP request, ARP packets and NDP might still be flooded across the WAN. ARP suppression and NDP suppression prevent this flooding from occurring.

Proxy ARP and ARP suppression and proxy NDP and NDP suppression are supported in the following scenarios:

- Single-homed devices in active mode—EVPN-MPLS and EVPN-VXLAN
- Multihomed devices in active-active mode—EVPN-MPLS and EVPN-VXLAN
- Multihomed devices in single-active mode—EVPN-MPLS only

In a multihoming active-active scenario, the database of MAC-IP address bindings are synchronized between the PE device or Layer 2 or 3 VXLAN gateway that act as the designated forwarder (DF) and non-designated forwarder (non-DF).

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
19.1R1	Starting in Junos OS Release 19.1R1, the <code>no-arp-suppression</code> statement is hidden in the Junos CLI.
18.2R1	Starting with Junos OS Release 19.1R1, QFX10000 switches that function as Layer 2 VXLAN gateways in an EVPN-VXLAN environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on a non-IRB interface. You can also limit the number of MAC-IP address bindings that can be learned on these switches.
18.1R1	Starting with Junos OS Release 18.1R1, QFX5100 and QFX5200 switches that function as Layer 2 VXLAN gateways in an EVPN-VXLAN environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on non-IRB interfaces. QFX5110 switches that function as Layer 2 or 3 VXLAN gateways in an EVPN-VXLAN environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on IRB interfaces and non-IRB interfaces. You can also limit the number of MAC-IP address bindings that can be learned on these switches.
17.4R2	Starting with Junos OS Release 17.4R2, MX Series routers and EX9200 switches support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on non-IRB interfaces.
17.4R2	Junos OS Release 17.4R2 also introduces the ability to limit the number of media-access-control (MAC)-IP address bindings that can be learned on these Juniper Networks devices.
17.3R1	Starting with Junos OS Release 17.3R1, QFX10000 switches that function as Layer 3 VXLAN gateways in an EVPN-VXLAN environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on an IRB interface.

17.2R1	Starting with Junos OS Release 17.2R1, MX Series routers and EX9200 switches that function as provider edge (PE) devices in an Ethernet VPN-MPLS (EVPN-MPLS) or Layer 3 VXLAN gateways in an Ethernet VPN-Virtual Extensible LAN (EVPN-VXLAN) environment support proxy ARP and ARP suppression, and proxy NDP and NDP suppression on an integrated and routing (IRB) interface.
--------	--

RELATED DOCUMENTATION

global-mac-ip-limit

interface-mac-ip-limit

mac-ip-table-size

ARP and NDP Request with a proxy MAC address

IN THIS SECTION

- [Benefits of using a virtual gateway or IRB MAC address in an ARP request](#) | 557

Starting in Junos OS Release 19.1R1, PE devices support the substitution of a source MAC address with a proxy MAC address in the ARP or NDP reply. When the PE device receives an ARP or NDP request, the PE device searches the MAC-IP address binding database and if there is an entry, it replaces the source MAC address with the proxy MAC address in the ARP reply.

For EVPN networks with an edge-routed bridging overlay, you can enable proxy MAC address using the `irb` option. Junos OS uses the virtual gateway MAC address as the proxy MAC address on each leaf device. When virtual gateway is not configured, Junos uses the IRB MAC address as the proxy MAC address on the leaf device.

If the entry is not found in the MAC-IP address binding database in an edge-routed bridging EVPN network, the source MAC and IP address in the ARP request are replaced with the proxy MAC and IP address of the IRB interface and the ARP request is flooded.

In edge-routed bridging, Junos specifies the proxy MAC address using the following precedence order:

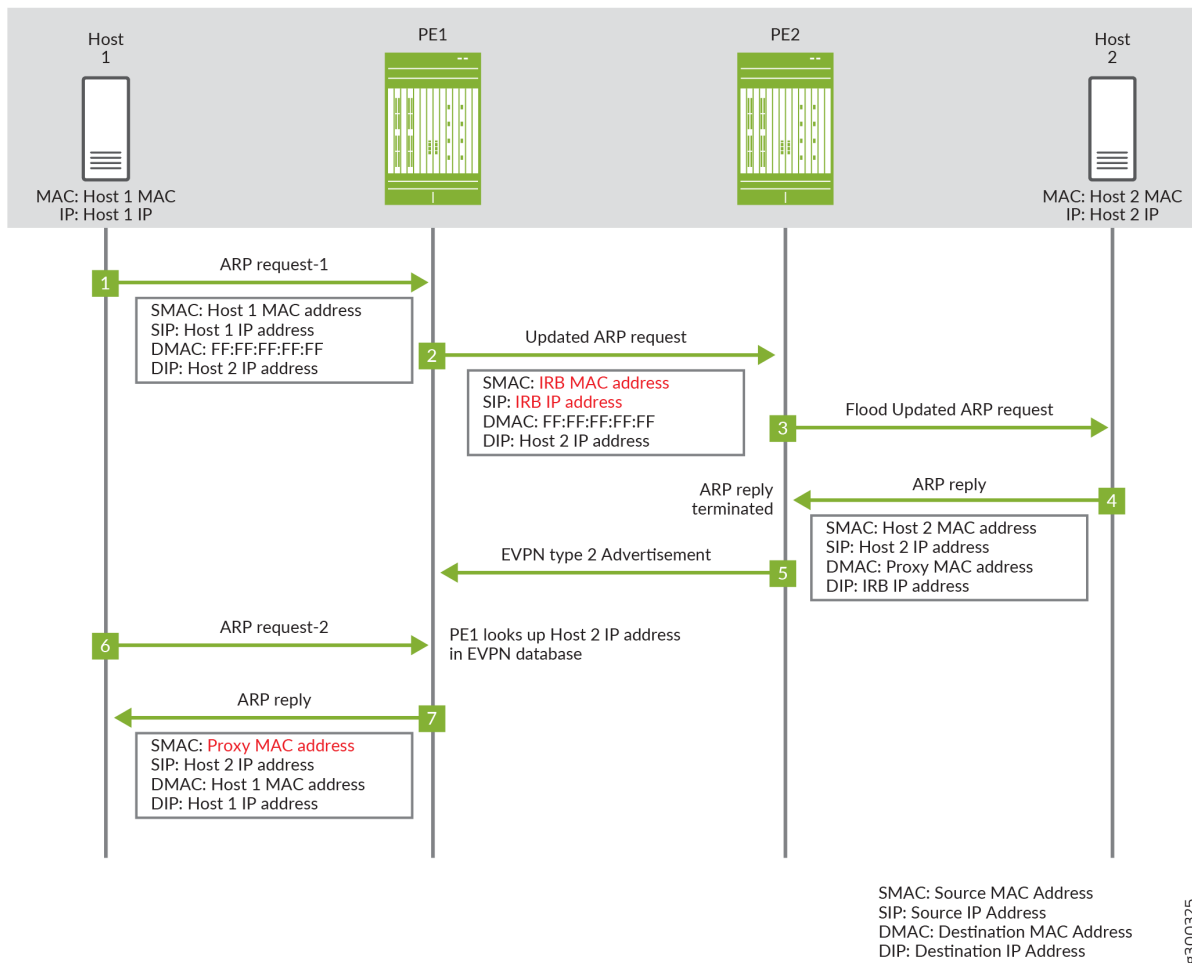
1. A configured virtual gateway MAC address.
2. An automatically derived virtual gateway MAC address.

3. A configured IRB MAC address.
4. Automatically assigned MAC address based on the physical interface when neither the virtual gateway MAC address nor the IRB MAC address is configured.

[Figure 39 on page 554](#) illustrates how proxy MAC address is applied in an edge-routed bridging EVPN network. The sequence of events for an ARP request is as follows:

1. Host 1 sends the first ARP request message to locate Host 2. The ARP request message has the MAC and IP address of host 1 as the source MAC and IP address.
2. Device PE1 receives the ARP request message. Since there are no matching entries in its MAC-IP address binding database, PE1 appropriates the ARP request message and replaces the MAC and IP address of Host 1 with the MAC and IP address of the IRB interface.
3. Device PE2 forwards the ARP request message.
4. Host 2 sends the ARP response with its MAC and IP address.
5. Device PE2 adds an entry for Host 2 to its MAC-IP address binding database and sends an EVPN type 2 advertisement message for Host 2. When PE1 receives the EVPN type 2 message, it adds an entry for Host 2 to its MAC-IP address binding database.
6. ARP request 1 times out and host 1 sends another ARP request message.
7. Device PE1 receives the second ARP request message. Since there is an entry in the MAC-IP address binding database, it responds as the ARP proxy with the proxy MAC address.

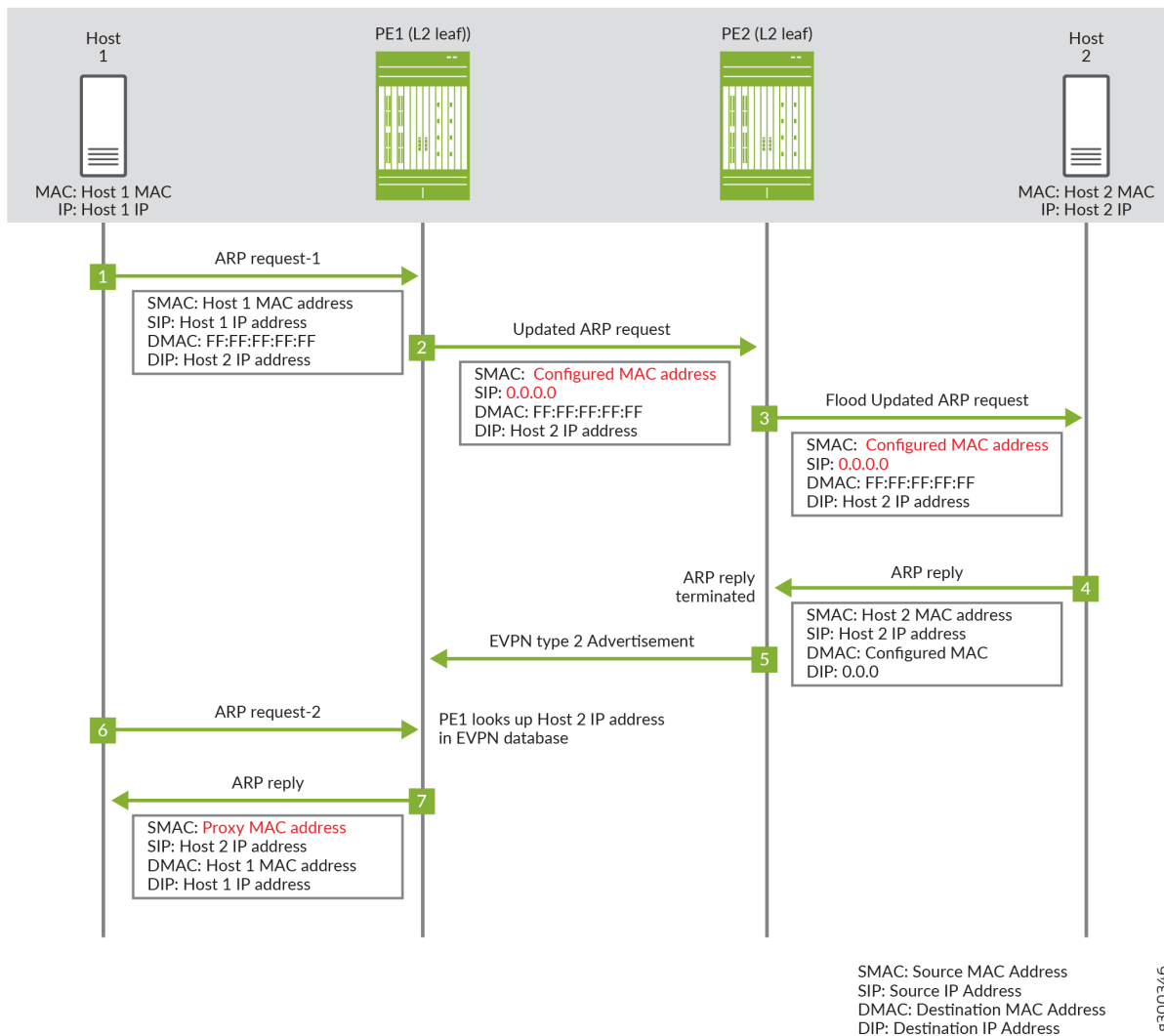
Figure 39: Proxy MAC ARP Request in an Edge-Routed Bridging EVPN Network



For EVPN networks with a centrally-routed bridging overlay, you assign a configured MAC address as the proxy MAC address. The configured proxy MAC address on the leaf should be the virtual gateway MAC address of the IRB on spine device or gateway. When the virtual gateway MAC address is not configured, configure the proxy MAC address to use the IRB MAC address on the centralized gateway.

If the entry is not found in the MAC-IP address binding database in a centrally-routed bridging EVPN network, the source MAC is replaced with the configured proxy MAC and the source IP address is set to 0.0.0.0 in the ARP request. [Figure 40 on page 555](#) illustrates how proxy MAC address is applied in a centrally-routed bridging EVPN Network where Host 1 and Host 2 are connected to leaf devices PE1 and PE2 respectively. The sequence of events for the ARP request is as follows:

Figure 40: Proxy MAC ARP Request in a Centrally-routed Bridging EVPN Network



1. Host 1 sends the first ARP request message to locate Host 2. The ARP request message has the MAC and IP address of Host 1 as the source MAC and IP address.
2. Leaf device PE1 receives the ARP request message. Since there are no matching entries in its MAC-IP address binding database, PE1 floods the ARP request message and replaces the source MAC address of Host 1 with the configured proxy MAC address and a source IP address to 0.0.0.0.
3. Leaf device PE2 receives and forwards the ARP request message.
4. Host 2 sends the ARP response with its MAC and IP address.

5. Device PE2 adds an entry for Host 2 to its MAC-IP address binding database and sends an EVPN type 2 advertisement message for Host 2. When PE1 receives the EVPN type 2 message, it adds an entry for Host 2 to its MAC-IP address binding database.
6. ARP request 1 times out and host 1 sends another ARP request message.
7. Device PE1 receives the second ARP request message. Since there is an entry in the MAC-IP address binding database, it responds as the ARP proxy with the proxy MAC address.

This feature ensures all inter-VLAN and intra-VLAN traffic are forwarded as Layer 3 traffic to the configured gateway and that the traffic can be routed.

To enable this feature, include the `proxy-mac (irb | proxy-mac-address)` statement in the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy for the evpn instance type or in the `[edit routing-instances routing-instance-name bridge-domains domain_name]` hierarchy for the virtual-switch instance type.

In an EVPN network with edge-routed bridging overlay where CE devices are directly connected to a L3 gateway device, configure `proxy-mac` on the gateway devices with the `irb` option.

In an EVPN network with a centrally-routed bridging overlay where CE devices are connected to an L3 gateway device via layer 2 only leaf devices, configure `proxy-mac` on the leaf devices with the `proxy-mac-address` option using the virtual or physical gateway MAC address on the centralized L3 gateway IRB interface as the MAC address.

The following is a sample configuration for bridge domain with support for proxy MAC for an edge-routed bridging EVPN network.

```
routing-instances {
  VS-1 {
    instance-type virtual-switch;
    bridge-domains {
      bd-110 {
        interface ae0.0;
        vlan-id 110;
        routing-interface irb.5;
        proxy-mac {
          irb;
        }
      }
    }
  }
}
```

Benefits of using a virtual gateway or IRB MAC address in an ARP request

With this feature, all inter-VLAN and intra-VLAN traffic for all configured routing instances will be routed to an IRB interface. This allows the following:

- Communication between private VLANs.
- Layer 3 security filtering at the gateway for both intra-VLAN and inter-VLAN traffic.
- MAC address privacy. Host do not learn the MAC addresses of other hosts.

RELATED DOCUMENTATION

[EVPN Proxy ARP and ARP Suppression, and Proxy NDP and NDP Suppression](#) | 549

proxy-mac

Configuring DHCP Relay Agents

IN THIS CHAPTER

- DHCP Relay Agent in EVPN-MPLS Network | 558
- DHCP Relay Agent over EVPN-VXLAN | 561

DHCP Relay Agent in EVPN-MPLS Network

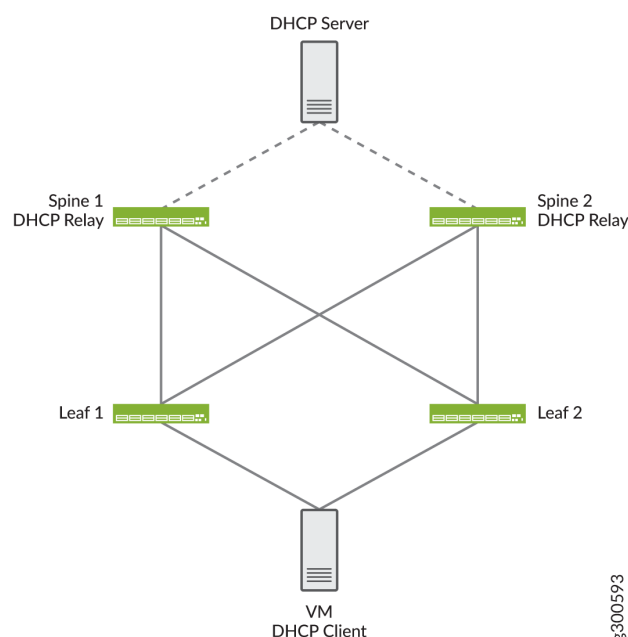
IN THIS SECTION

- Benefits of DHCP Relay Agent | 560
- Configuring DHCP Relay Agents in EVPN-MPLS | 560

Dynamic Host Configuration Protocol (DHCP) is a protocol that allows a DHCP server to dynamically allocate IP addresses to DHCP clients. The DHCP relay agent forward DHCP messages between DHCP clients and DHCP servers when they are on different networks. Junos supports configuring the DHCP relay agent on the spine devices in a centrally bridging overlay on an EVPN-MPLS network.

[Figure 41 on page 559](#) shows a centrally-routed bridging EVPN-MPLS network with a virtual machine on a datacenter server multihomed to leaf 1 and leaf 2. The leaf devices are connected to spine devices that function as DHCP relays and forwards the DHCP messages to the DHCP server.

Figure 41: DHCP Relay Agent on Centrally-Routed Bridging Overlay



The following sequence describes the interaction between the DHCP client, DHCP relay agent, and DHCP server.

1. The DHCP client (VM) initiates a request by sending a DHCP discovery message to find a DHCP server and obtain a lease for an IP address. DHCPv6 clients send DHCPv6 solicit messages.
2. Leaf 1 broadcasts the new locally-learned MAC address and DHCP discovery (or DHCPv6 solicit) message to the other devices in the EVPN core.
3. Spine 1 forwards the DHCP discovery (or DHCPv6 solicit) message to DHCP server.
4. Upon receipt of the DHCP discovery (or DHCPv6 solicit) message, the DHCP server issues a DHCP offer message (or DHCPv6 advertisement) message with the IP address back to Spine 1.
5. Spine 1 forwards the DHCP offer (or DHCPv6 advertisement) message to Leaf 1 where the message is sent back to the client.

Subsequent DHCP request (or DHCPv6 request) and DHCP acknowledgment (or DHCPv6 reply) messages are similarly forwarded. For DHCP renew and DHCP release messages, the DHCP client sends unicast messages to renew or release their IP address. The DHCP V6 clients send multicast messages to renew or release their IP address.

Benefits of DHCP Relay Agent

DHCP allows you to manage IP addresses and other network configurations easily. The DHCP relay on the spine devices of a centrally routed bridging overlay allows you to simplify and centralized the routing of the DHCP messages.

Configuring DHCP Relay Agents in EVPN-MPLS

To configure the DHCP relay agent on the Junos OS device, include the `dhcp-relay` statement at the `[edit forwarding-options]` hierarchy level for either DHCP or DHCPv6 services.

To configure the DHCP server, create a server group with at least one DHCP server IP addresses on the DHCP relay agent and apply the server-group as a member of an active-server-group.

To configure a group of DHCP server addresses and define them as an active server group:

1. Specify the name of the server group.

```
[edit forwarding-options dhcp-relay]
user@host# set server-group server-group-name
```

2. Add the IP addresses of the DHCP servers belonging to the group.

```
[edit forwarding-options dhcp-relay server-group
server-group-name]
user@host# set ip-address1
user@host# set ip-address2
```

3. Define the server group as an active server group.

```
[edit forwarding-options dhcp-relay]
user@host# set active-server-group server-group-name
```

The following configuration snippet shows a sample DHCP relay configuration.

```
forwarding-options {
  dhcp-relay {
    server-group {
      dhcp-server {
        10.0.0.3;
```

```

    }
  }
}
active-server-group dhcp-server;
group v4-relay {
    interface irb.0;
}
}

```



NOTE: DHCP relay agent on EVPN-MPLS does not support active leasequery (ALQ).

RELATED DOCUMENTATION

[active-server-group](#)

[dhcp-relay](#)

[server-group](#)

DHCP Relay Agent over EVPN-VXLAN

IN THIS SECTION

- [DHCP Relay on a Centrally-routed Bridging Overlay | 562](#)
- [DHCP Relay on an Edge-routed Bridging Overlay | 563](#)

Dynamic Host Configuration Protocol (DHCP) is a protocol that enables a DHCP server to dynamically allocate IP addresses to DHCP clients. This allows you to manage IP addresses and other network configurations easily. The DHCP relay agent forward DHCP messages between DHCP clients and DHCP servers when they are on different networks. Junos supports configuring the DHCP relay agent in an EVPN-VXLAN fabric.

In an EVPN-VXLAN fabric, the DHCP server and the DHCP clients connect into the network using access interfaces on leaf devices. The DHCP server and clients can communicate with each other over the existing network without further configuration when the DHCP client and server are in the same

VLAN. When a DHCP client and server are in different VLANs, DHCP traffic between the client and server is forwarded between the VLANs through integrated routing and bridging (IRB) interfaces.

You can configure DHCP relay on a centrally-routed bridging overlay or on an edge-routed bridging overlay in an EVPN-VXLAN fabric. The main difference between the overlay models for DHCP relay configuration is the IRB interfaces. In a centrally-routed bridging overlay, these IRB interfaces are on the spine devices, while in an edge-routed bridging overlay, they are on the leaf devices.



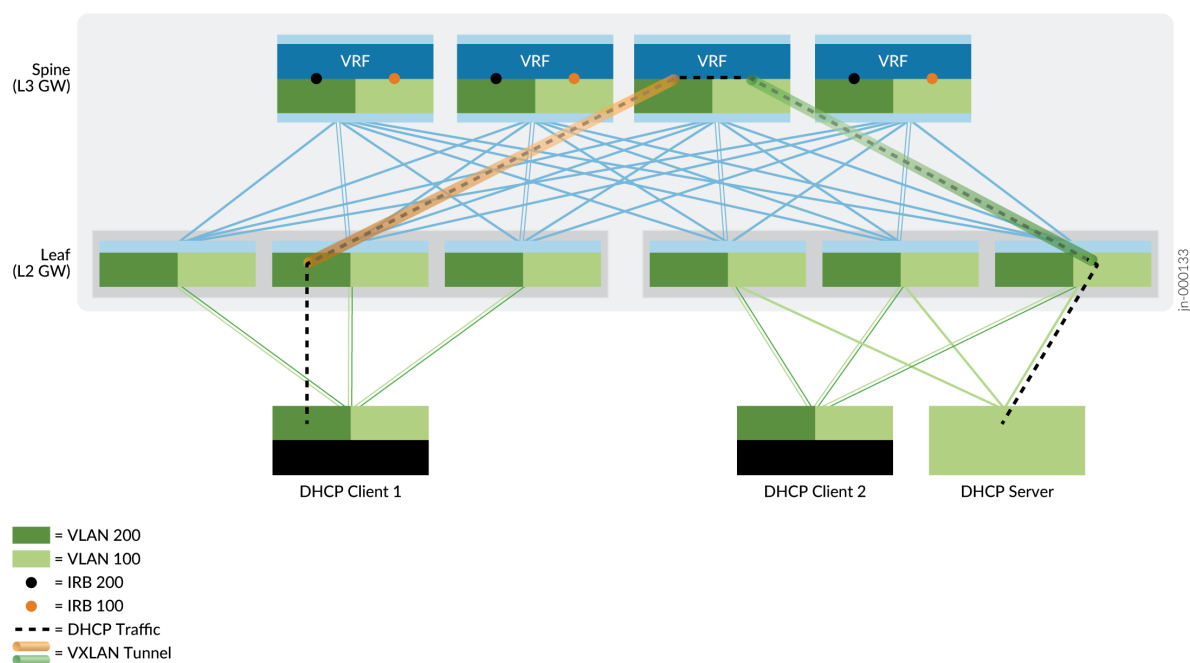
NOTE: DHCP relay is supported in EVPN-VXLAN fabrics with IPv6 underlay. For information on IPv6 underlay, see ["EVPN-VXLAN with an IPv6 Underlay" on page 897](#).

DHCP Relay on a Centrally-routed Bridging Overlay

In a centrally-routed bridging overlay, the spine devices function as Layer 3 gateways that handle traffic between VXLANs while the leaf devices function as Layer 2 gateways that handle traffic within a VXLAN. DHCP clients are connected to pure Layer 2 EVPN leaf devices. DHCP packets are sent over the EVPN core towards spine devices, which function as DHCP relay agents, forwarding the packets to the servers. Configuring DHCP relay on the spine devices of a centrally-routed bridging overlay allows you to simplify and centralize the routing of the DHCP messages.

[Figure 42 on page 563](#) shows how DHCP traffic is forwarded in a centrally-routed bridging overlay where the DHCP client and server connect to access interfaces on different leaf devices in different VLANs. For information on how to configure DHCP relay in a centrally-routed bridging overlay, see [DHCP Relay Design and Implementation](#).

Figure 42: DHCP Relay Agent on Centrally-Routed Bridging Overlay

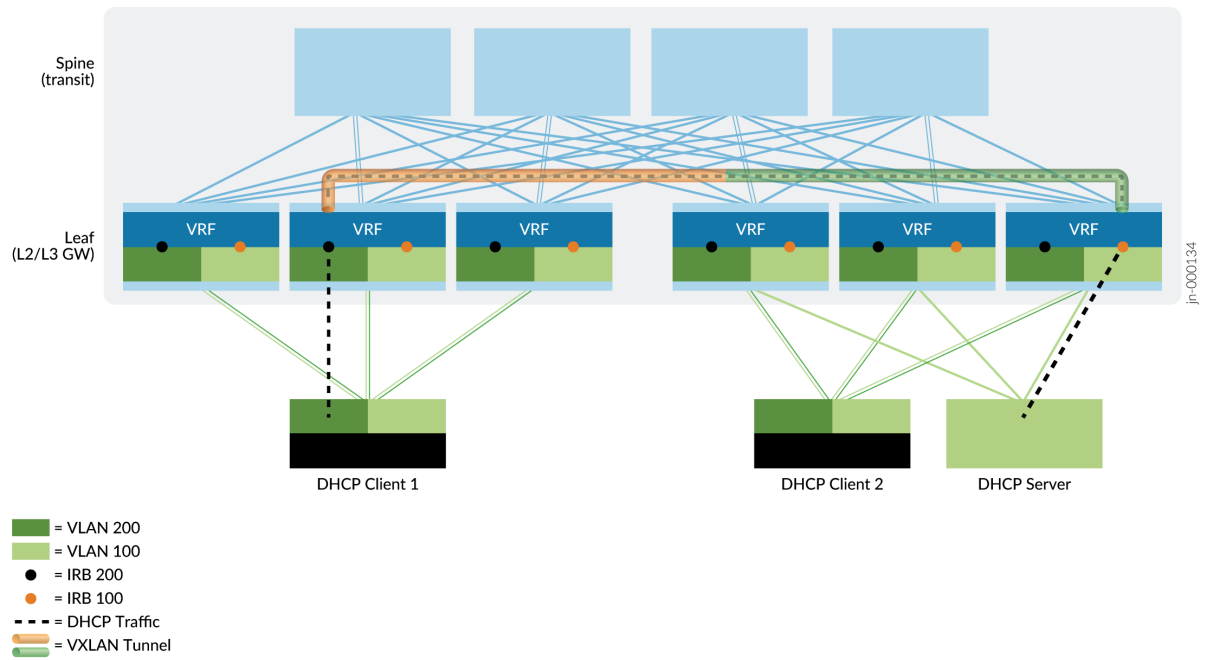


DHCP Relay on an Edge-routed Bridging Overlay

In an edge-routed bridging overlay, the leaf devices serve as both Layer 2 and Layer 3 gateways. All leaf devices support IRB functionality and function as VTEPs to establish VXLAN tunnels. Spine devices are configured to handle only IP traffic, which removes the need to extend the bridging overlays to the spine devices.

[Figure 43 on page 564](#) shows how DHCP traffic is forwarded in an edge-routed bridging overlay where the DHCP client and server connect to access interfaces on different leaf devices in different VLANs. For more information on configuring DHCP relay in an edge-routed bridging overlay, see [Configure a DHCP Relay in EVPN-VXLAN Fabric Architecture](#).

Figure 43: DHCP Relay Agent on Edge-Routed Bridging Overlay



High Availability in EVPN

IN THIS CHAPTER

- [NSR and Unified ISSU Support for EVPN | 565](#)
- [Preventing Traffic Loss in an EVPN-VXLAN Environment With GRES and NSR | 568](#)
- [Graceful Restart in EVPN | 569](#)

NSR and Unified ISSU Support for EVPN

IN THIS SECTION

- [Supported Features on EVPN | 567](#)

Junos OS ensures minimal loss of traffic when a Routing Engine switchover occurs with nonstop active routing (NSR) and graceful Routing Engine switchover (GRES) enabled. The forwarding state of the Packet Forwarding Engine (PFE) remains intact during switchover. The signaling state on the primary Routing Engine and on the standby Routing Engine are built in parallel.

EVPN reproduces dynamically generated data (such as labels and sequence numbers), and data obtained from peers on the primary Routing Engine, and on the standby Routing Engine. EVPN also monitors BGP ingress and egress routing table messages on the standby Routing Engine to populate its signaling plane data structures. Local MAC addresses are obtained by the Layer 2 address learning process (l2ald), which transfers the data to the EVPN module in the route processing software. In the network layer reachability information (NLRI) fields of its packets, BGP transfers the MAC addresses to peers in the network.

In earlier releases, the l2ald did not run on the standby Routing Engine. Consequently, the l2ald did not inform the routing protocol process on the standby Routing Engine about locally learned MAC addresses. With this feature, the routing protocol process learns of newly acquired MAC addresses by listening to the BGP ingress and egress routing table messages and then populates its data structures.

Following a switchover, when the I2ald becomes active on the new primary Routing Engine, it sends all locally-learned MAC addresses to the routing protocol process, which can then verify the MAC addresses learned from the I2ald with MAC addresses derived from the BGP routing table egress messages.



NOTE: Expect a traffic loss pertaining to a topology change if the topology change occurs during a switchover.

This feature mirrors the following data to the standby Routing Engine:

- MAC route labels—EVPN allocates a MAC route label per EVPN instance (EVI) and per Ethernet segment Identifier (ESI). MAC labels, including the EVI and ESI are mirrored to the standby Routing Engine.
- Inclusive multicast (IM) route labels—EVPN allocates an IM label per VLAN. An IM label, including the EVI name and VLAN ID are mirrored to the standby Routing Engine.
- Split horizon labels—EVPN mirrors a split horizon label and the EVI name to the standby Routing Engine.
- Aliasing labels—EVPN mirrors an aliasing label and EVI name to the standby Routing Engine.
- Dummy labels—EVPN creates a dummy label with which it adds the egress route to the mpls.0 table. EVPN mirrors a dummy label and the EVI name to the standby Routing Engine so the mpls.0 table is identical on the primary and standby Routing Engines.

For EVPN ETREE, Junos mirrors the local EVPN ETREE leaf label that is advertised to other PE as part of the ETREE extended community on the standby Routing Engine.

For EVPN P2MP, Junos mirrors the LDP/RSVP transport label on the standby Routing Engine.

NSR Data Flow Pre-Switchover

The EVPN configuration on the standby Routing Engine directs it to operate identically to the primary Routing Engine except that it does not allocate any labels. Label information is updated on the standby Routing Engine when it receives the information from its label information base mirror (libmirror).

BGP updates remote MAC addresses in the instance.EVPN table on the standby Routing Engine. Just as EVPN operates on the primary Routing Engine, the standby Routing Engine derives information from its flash drive to update its data structures.

To gather local MAC addresses and update data structures, EVPN on the primary Routing Engine syncs its MAC address information with the standby routing engine. Following a switchover, when the I2ald becomes active, if it sends MAC addresses that are identical to those marked stale, the addresses are retained. If the I2ald does not send the same MAC addresses learned from BGP RIB egress messages, it deletes the addresses.

NSR Data Flow Post Switchover

Following a switchover, when the standby Routing Engine becomes the primary Routing Engine, it establishes connection with the I2ald. When the I2ald becomes active, it sends local MAC addresses to the routing protocol process. The routing protocol process removes the stale bit from the MAC addresses it receives from the I2ald.

MAC addresses that were not derived from BGP RIB egress messages, but were sent to the routing protocol process by the I2ald, are treated as newly learned MAC addresses. BGP advertises such MAC addresses.

When the I2ald sends the end marker of the MAC address to the routing protocol process, all local MAC addresses marked as stale are deleted.

Unified ISSU Support

Unified in-service software upgrade is supported for VXLAN on MX Series routers. ISSU enables you to upgrade your Junos OS software on your MX Series router with no disruption on the control plane and with minimal disruption of traffic. Unified ISSU is supported only on dual Routing Engine platforms. In addition, the graceful Routing Engine switchover (GRES) and nonstop active routing (NSR) features must be enabled. Unified ISSU allows you to eliminate network downtime, reduce operating costs, and deliver higher levels of services. See [Getting Started with Unified In-Service Software Upgrade](#).

To enable GRES, include the `graceful-switchover` statement at the `[edit chassis redundancy]` hierarchy level.

To enable NSR, include the `nonstop-routing` statement at the `[edit routing-options]` hierarchy level and the `commit synchronize` statement at the `[edit system]` hierarchy level.

Supported Features on EVPN

Junos OS supports NSR/ISSU on the following features for EVPN:

- EVPN with ingress replication for BUM traffic
- EVPN-ETREE
- EVPN-VPWS
- EVPN -VXLAN
- PBB-EVPN
- EVPN with P2MP mLDP replication for BUM traffic

Use [Feature Explorer](#) to confirm platform and release support for specific features.

RELATED DOCUMENTATION

show evpn instance

show evpn database

[Tracing EVPN Traffic and Operations | 80](#)

[Understanding Graceful Routing Engine Switchover](#)

Preventing Traffic Loss in an EVPN-VXLAN Environment With GRES and NSR

The graceful Routing Engine switchover (GRES) feature in Junos OS enables devices with redundant Routing Engines to continue forwarding packets, even if one Routing Engine fails. To preserve routing during a switchover, GRES is combined with nonstop active routing (NSR).

Although GRES preserves interface and kernel information, in an EVPN-VXLAN environment, switches in the QFX10000 line might experience Layer 3 traffic loss if traffic is flowing during the switchover or during the Routing Engine reboot operation.



NOTE: This section doesn't apply to QFX10002 switches because those devices don't have redundant Routing Engines and don't support GRES.

To prevent traffic flows from being dropped in such a scenario, configure a hold-time interval that prevents IP phantom addresses from being added to the routing tables. During this interval, the routes are maintained in the kernel. After the interval expires, the phantom routes are added to the appropriate routing tables before they are deleted from the kernel.



NOTE: We recommended a hold-time value of 300 seconds (5 minutes) in EVPN-VXLAN environments.

1. Enable GRES.
2. Enable NSR.
3. Specify the recommended hold-time interval (300 seconds), which prevents phantom routes from being added to the routing table during a switchover until the interval expires.

```
[edit routing-options]
```

```
user@swtich# set nsr-phantom-holdtime seconds 300.
```



NOTE: The hold-time interval range is 0 through 10000.

RELATED DOCUMENTATION

nsr-phantom-holdtime

[Understanding Graceful Routing Engine Switchover](#)

Graceful Restart in EVPN

Starting in Junos OS 18.4R1 graceful restart is supported on devices in an EVPN-VXLAN network. Graceful restart allows a routing engine undergoing a switchover without NSR to inform its adjacent neighbors and peers of its condition. When you enable graceful-restart, the routing protocol process learns of newly acquired MAC addresses by listening to the BGP ingress and egress routing table messages. It then verifies MAC addresses learned from the I2ald, with MAC addresses derived from the BGP routing table egress messages. Graceful restart enables the device to reduce the overall network traffic loss while the device is passing through the intermediate convergence states that occurs during a restart.

To enable graceful restart, include the `graceful-restart` statement at the `[edit routing-options]` hierarchy level.

RELATED DOCUMENTATION

[Understanding Graceful Restart](#)

Monitoring EVPN Networks

IN THIS CHAPTER

- [Connectivity Fault Management Support for EVPN and Layer 2 VPN Overview | 570](#)
- [Configure a MEP to Generate and Respond to CFM Protocol Messages | 573](#)

Connectivity Fault Management Support for EVPN and Layer 2 VPN Overview

IN THIS SECTION

- [Limitations of CFM on layer 2 VPN and EVPNs | 571](#)

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 measurement, 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 \(MEP\) | 573](#)
- [Configure a Remote Maintenance Association End Point \(MEP\) | 575](#)

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-name mep 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-name mep 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](#)

Layer 2 Control Protocol Transparency

IN THIS CHAPTER

- [Layer 2 Control Protocol Transparency for EVPN | 577](#)

Layer 2 Control Protocol Transparency for EVPN

IN THIS SECTION

- [Benefits of L2CP Transparency | 578](#)
- [MEF Service Definitions | 578](#)
- [EVPN-MPLS E-LAN Services | 580](#)
- [Configure a Filter for EVPN MPLS E-LAN Services | 581](#)
- [EVPN-VPWS E-LINE Service | 582](#)
- [Configure a Filter for EVPN-VPWS Services | 584](#)

Layer 2 Control Protocols (L2CP) are Ethernet control protocols, such as spanning-tree, BPDUs, LACP, pause, and so on. The L2CP Ethernet frames have specific destination MAC addresses. The destination MAC addresses are reserved multicast MAC addresses in the range from 01-80-C2-00-00-00 through 01-80-C2-00-00-0F and from 01-80-C2-00-00-20 through 01-80-C2-00-00-2F.

The Metro Ethernet Forum (MEF) specifies the rules for processing L2CP Ethernet frame when the frames arrive at the L2CP decision point on the user network interface (UNI). The rules provide the mechanism for transparently passing the L2CP frame between a Carrier Ethernet Network and a Subscriber Network. [Table 26 on page 578](#) lists the actions that can be taken at the L2CP decision point and what transpires on Junos devices.

Table 26: MEF Model of Behavior

MEF Actions	Junos Device
Pass	Junos devices forwards L2CP frames without processing them when the protocol is not configured on the interface. The frame is treated like a multicast address and sent to all the local and remote CE.
Peer	When the protocol is configured on the interface, the Packet Forwarding Engine further processes the frame.
Discard	You must explicitly configure a filter on the appropriate physical or logical interface to discard the frame.

When an L2CP frame arrives at the UNI of the ingress PE Junos device, the Packet Forwarding Engine forwards (passes) the L2CP frame unless you configure the protocol on the local physical or logical interface. When a protocol is configured on the interface, the Packet Forwarding Engine further processes (peers) the frame unless you create a filter to drop (discard) it. We support L2CP transparency on EVPN-MPLS and EVPN-VPWS networks.

Benefits of L2CP Transparency

Adopting L2CP transparency promotes the interoperability of equipment between subscriber networks and carrier networks.

MEF Service Definitions

MEF defines Ethernet Private LAN (EP-LAN) as a port-based service that supports all-to-one bundling. This allows multiple VLANs to be mapped into a single Ethernet Virtual Circuit. Ethernet Virtual Private LAN (EVP-LAN) is a VLAN-based service where VLAN ID identifies the traffic. [Table 27 on page 579](#) lists and maps the MEF service type to the supported EVPN configuration on Junos OS Evolved devices.

Table 27: MEF Service Type

MEF Service Type	Junos EVPN Services
EP-LAN	<ul style="list-style-type: none"> • EVPN-MPLS with a MAC-VRF instances for VLAN-Bundle service with Ethernet bridge ports. • EVPN-MPLS with MAC-VRF instances for VLAN-Based service with Ethernet bridge ports. • L2 Basic Switching: default switching instance with Ethernet bridge ports. • L2 Basic Switching: virtual switching instance with Ethernet bridge ports.
EVP-LAN	<ul style="list-style-type: none"> • EVPN-MPLS E-LAN MAC-VRF VLAN-Based service with Service Provider Style interfaces (vlan-id-list). • EVPN-MPLS E-LAN MAC-VRF VLAN-Aware service with Service Provider Style interfaces. • EVPN-MPLS E-LAN MAC-VRF VLAN-Aware service with Enterprise Style interfaces (trunk or access interfaces). • EVPN-MPLS E-LAN MAC-VRF VLAN-Bundle service with vlan-id-list interfaces. • L2 Basic Switching: default switching instance with Service Provider Style interfaces with vlan-id-list. • L2 Basic Switching: default switching instance with Enterprise Style interfaces (trunk or access interfaces). • L2 Basic Switching: virtual switching instance with Service Provider Style interfaces with vlan-id-list. • L2 Basic Switching: virtual switching instance with Enterprise Style interfaces (trunk or access interfaces).

EVPN-MPLS E-LAN Services

Table 28 on page 580 lists the reserved MAC address, the L2CP assigned to those MAC addresses, and the actions required for EVPN E-LAN services.

Table 28: EVPN MPLS E-LAN Service

Destination MAC Address	L2CP	Ethertype /Subtype	EP-LAN	EVP-LAN
01-80-C2-00-00-00	STP RSTP MSTP	—	Pass	Peer or Discard
01-80-C2-00-00-01	Pause	0x8808	Peer or Discard	Peer or Discard
01-80-C2-00-00-02	LACP LAMP	0x8809/01 0x8809/02	Peer or Discard	Peer or Discard
01-80-C2-00-00-02	Link OAM	0x8809/03	Peer or Discard	Peer or Discard
01-80-C2-00-00-02	ESMC	0x8809/0A	Peer or Discard	Peer or Discard
01-80-C2-00-00-03	802.1X	0x888E	Peer or Discard	Peer or Discard
01-80-C2-00-00-04	MAC Specific Control Protocols	—	Peer or Discard	Peer or Discard
01-80-C2-00-00-05	Reserved	—	Peer or Discard	Peer or Discard
01-80-C2-00-00-06	Reserved	—	Peer or Discard	Peer or Discard
01-80-C2-00-00-07	E-LMI	0x88EE	Peer or Discard	Peer or Discard
01-80-C2-00-00-08	Provider Bridge Group Address	—	Peer or Discard	Peer or Discard
01-80-C2-00-00-09	Reserved	—	Peer or Discard	Peer or Discard
01-80-C2-00-00-0A	Reserved	—	Peer or Discard	Peer or Discard

Table 28: EVPN MPLS E-LAN Service (Continued)

Destination MAC Address	L2CP	Ethertype /Subtype	EP-LAN	EVP-LAN
01-80-C2-00-00-0B	Reserved	—	Pass	Peer or Discard
01-80-C2-00-00-0C	Reserved	—	Pass	Peer or Discard
01-80-C2-00-00-0D	Provider Bridge MVRP Address	—	Pass	Peer or Discard
01-80-C2-00-00-0E	LLDP	0x88CC	Peer or Discard	Peer or Discard
01-80-C2-00-00-0E	PTP Peer Delay	0x88F7	Peer or Discard	Peer or Discard
01-80-C2-00-00-0F	Reserved	—	Pass	Peer or Discard
01-80-C2-00-00-20 through 01-80-C2-00-00-2F	GARP GMRP	—	Pass	Pass

Configure a Filter for EVPN MPLS E-LAN Services

Configure a discard filter for EP-LAN or EVP-LAN for EVPN MPLS E-LAN services as follows:

1. Configure a firewall filter to discard traffic that matches the reserved destination MAC address for family ethernet-switching.

```
set firewall family ethernet-switching filter name term name from destination-mac-address mac-address
set firewall family ethernet-switching filter name term name then discard.
```



NOTE: You must configure a firewall filter to match the destination MAC address and ether type for the following protocol.

- Pause
- LACP

- LAMP
- Link OAM
- ESMC
- 802.1X
- E-LMI
- LLDP
- PTP Peer Delay

```
set firewall family ethernet-switching filter name term name from destination-mac-address mac-address
set firewall family ethernet-switching filter name term name from ether-type value.
set firewall family ethernet-switching filter name term name then discard.
```

2. Apply the firewall filter to the interface.

```
set interfaces interface-name unit n family ethernet-switching filter name
```

The following is the sample firewall filter configuration for discarding the LACP Ethernet frames. The filter is applied to the EP-LAN interface.

```
set firewall family ethernet-switching filter eplan term lacp from destination-mac-address 01:80:c2:00:00:02/48
set firewall family ethernet-switching filter eplan term lacp from ether-type 0x8809
set firewall family ethernet-switching filter eplan term lacp then discard
set interfaces et-0/0/1 unit 0 family ethernet-switching filter input eplan
```

EVPN-VPWS E-LINE Service

[Table 29 on page 583](#) lists the reserved MAC address, L2CP assigned to those MAC addresses, and the actions required for EVPN VPWS (E-LINE) services. MEF allows different actions for some EP-LAN services on EVPN-VPWS and defines the two options for those protocols as Option 1 and Option 2.

Table 29: EVPN-VPWS

Destination MAC Address	L2CP	Ethertype/ Subtype	EP-LAN Option 1	EP-LAN Option 2	EVP-LAN
01-80-C2-00-00-00	STP RSTP MSTP	—	Pass	Pass	Peer or Discard
01-80-C2-00-00-01	Pause	0x8808	Peer or Discard	Peer or Discard	Peer or Discard
01-80-C2-00-00-02	LACP LAMP	0x8809/01 0x8809/02	Peer or Discard	Pass	Peer or Discard
01-80-C2-00-00-02	Link OAM	0x8809/03	Peer or Discard	Pass	Peer or Discard
01-80-C2-00-00-02	ESMC	0x8809/0A	Peer or Discard	Pass	Peer or Discard
01-80-C2-00-00-03	802.1X	0x888E	Peer or Discard	Pass	Peer or Discard
01-80-C2-00-00-04	MAC-specific Control Protocols	—	Peer or Discard	Peer or Discard	Peer or Discard
01-80-C2-00-00-05	Reserved	—	Peer or Discard	Peer or Discard	Peer or Discard
01-80-C2-00-00-06	Reserved	—	Peer or Discard	Peer or Discard	Peer or Discard
01-80-C2-00-00-07	E-LMI	0x88EE	Peer or Discard	Pass	Peer or Discard
01-80-C2-00-00-08	Provider Bridge Group Address	—	Peer or Discard	Peer or Discard	Peer or Discard

Table 29: EVPN-VPWS (Continued)

Destination MAC Address	L2CP	Ethertype/Subtype	EP-LAN Option 1	EP-LAN Option 2	EVP-LAN
01-80-C2-00-00-09	Reserved	—	Peer or Discard	Peer or Discard	Peer or Discard
01-80-C2-00-00-0A	Reserved	—	Peer or Discard	Peer or Discard	Peer or Discard
01-80-C2-00-00-0B	Reserved	—	Pass	Pass	Peer or Discard
01-80-C2-00-00-0C	Reserved	—	Pass	Pass	Peer or Discard
01-80-C2-00-00-0D	Provider Bridge MVRP Address	—	Pass	Pass	Peer or Discard
01-80-C2-00-00-0E	LLDP	0x88CC	Peer or Discard	Pass	Peer or Discard
01-80-C2-00-00-0E	PTP Peer Delay	0x88F7	Peer or Discard	Pass	Peer or Discard
01-80-C2-00-00-0F	Reserved	—	Pass	Pass	Peer or Discard
01-80-C2-00-00-20 through 01-80-C2-00-00-2F	GARP GMRP	—	Pass	Pass	Pass

Configure a Filter for EVPN-VPWS Services

Configure a discard filter for EP-LAN (Option 1 or Option 2) or EVP-LAN for EVPN VPWS services as follows:

1. Configure a firewall filter to discard traffic that matches the reserved destination MAC address for family ccc.

```
set firewall family ccc filter name term name from destination-mac-address mac-address
set firewall family ccc filter name term name then discard.
```



NOTE: You must configure a firewall filter to match the destination MAC address and ethertype for the following protocol.

- Pause
- LACP
- LAMP
- Link OAM
- ESMC
- 802.1X
- E-LMI
- LLDP
- PTP Peer Delay

```
set firewall family ccc filter name term name from destination-mac-address mac-address
```

```
set firewall family ccc filter name term name from ether-type value.
```

```
set firewall family ccc filter name term name then discard.
```

2. Apply the firewall filter to the interface.

```
set interfaces interface-name unit n family ccc filter name
```

The following is the sample firewall filter configuration for discarding provider bridge group address Ethernet frames. The filter is applied to the EP-LAN interface.

```
set firewall family ccc filter epl_option_1 term pbga from destination-mac-address
01:80:c2:00:00:08/48
set firewall family ccc filter epl_option_1 term pbga then discard
set interfaces et-0/0/3 unit 0 family ccc filter input epl_option_1
```

RELATED DOCUMENTATION

[MAC-VRF Routing Instance Type Overview | 38](#)

[Configuring VPWS with EVPN Signaling Mechanisms | 1629](#)

Example: Configuring Filtering of Frames by MAC Address

2

PART

EVPN-VXLAN

- Overview | **588**
 - Configuring EVPN-VXLAN Interfaces | **654**
 - Configuring VLAN-Aware Bundle Services, VLAN-Based Services, and Virtual Switch Support | **710**
 - Load Balancing with EVPN-VXLAN Multihoming | **723**
 - Setting Up a Layer 3 VXLAN Gateway | **741**
 - Configuring an EVPN-VXLAN Centrally-Routed Bridged Overlay | **771**
 - Configuring an EVPN-VXLAN Edge-Routed Bridging Overlay | **845**
 - IPv6 Underlay for VXLAN Overlays | **897**
 - Multicast Features with EVPN-VXLAN | **925**
 - Microsegmentation Using Group-Based Policies | **1272**
 - Configuring the Tunneling of Q-in-Q Traffic | **1326**
 - Tunnel Traffic Inspection on SRX Series Devices | **1354**
 - Fault Detection and Isolation in EVPN-VXLAN Fabrics | **1394**
-

CHAPTER 14

Overview

IN THIS CHAPTER

- Understanding EVPN with VXLAN Data Plane Encapsulation | 588
- EVPN-over-VXLAN Supported Functionality | 599
- Understanding VXLANs | 606
- VXLAN Constraints on EX Series, QFX Series, PTX Series, and ACX Series Devices | 616
- EVPN Over VXLAN Encapsulation Configuration Overview for QFX Series and EX4600 Switches | 628
- Implementing EVPN-VXLAN for Data Centers | 630
- PIM NSR and Unified ISSU Support for VXLAN Overview | 634
- Routing IPv6 Data Traffic through an EVPN-VXLAN Network with an IPv4 Underlay | 636
- Understanding How to Configure VXLANs and Layer 3 Logical Interfaces to Interoperate | 651

Understanding EVPN with VXLAN Data Plane Encapsulation

IN THIS SECTION

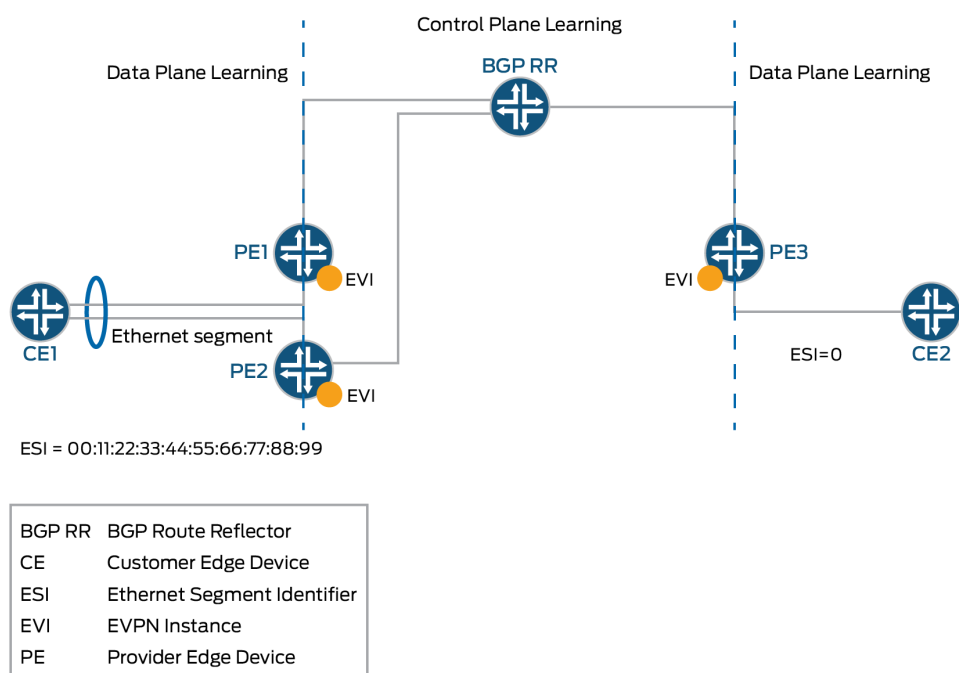
- Understanding EVPN | 589
- Understanding VXLAN | 592
- EVPN-VXLAN Integration Overview | 592
- Firewall Filtering and Policing Support for EVPN-VXLAN | 594
- Understanding Contrail Virtual Networks Use with EVPN-VXLAN | 595
- EVPN-VXLAN Support for VXLAN Underlays on MX Series Routers and the EX9200 Line of Switches | 596
- EVPN-VXLAN Support for VXLAN Underlays on QFX Series Switches | 596
- EVPN-VXLAN Support for VXLAN Underlays on ACX Series Devices | 597
- EVPN-VXLAN Packet Format | 597

Ethernet VPNs (EVPNs) enable you to connect groups of dispersed customer sites using Layer 2 virtual bridges, and Virtual Extensible LANs (VXLANs) allow you to stretch Layer 2 connectivity over an intervening Layer 3 network, while providing network segmentation like a VLAN, but without the scaling limitation of traditional VLANs. EVPN with VXLAN encapsulation handles Layer 2 connectivity at the scale required by cloud server providers and replaces limiting protocols like Spanning Tree Protocol (STP), freeing up your Layer 3 network to use more robust routing protocols. EVPN with VXLAN data plane encapsulation can be used with and without Juniper Networks Contrail virtualization software—use Contrail with EVPN VXLAN data plane encapsulation when you have an environment that includes both virtual and bare-metal devices.

Understanding EVPN

Ethernet VPN (EVPN) is a standards-based technology that provides virtual multipoint bridged connectivity between different Layer 2 domains over an IP or IP/MPLS backbone network. Like other VPN technologies, such as IP VPN and virtual private LAN service (VPLS), EVPN instances are configured on provider edge (PE) routers to maintain logical service separation between customers. The PE routers connect to customer edge (CE) devices, which can be routers, switches, or hosts. The PE routers then exchange reachability information using Multiprotocol BGP (MP-BGP) and encapsulated traffic is forwarded between PE routers. Because elements of the architecture are common with other VPN technologies, you can seamlessly introduce and integrate EVPN into existing service environments, as shown in [Figure 44 on page 590](#).

Figure 44: EVPN Overview



The EVPN is used as a Layer 2 overlay solution to provide Layer 2 connection over an IP underlay for the endpoints within a virtual network whenever Layer 2 connectivity is required by an end station such as bare-metal server (BMS). Otherwise, Layer 3 routing is used through VRF tables between Contrail vRouters and MX Series routers. EVPN technology offers multitenancy, flexible services that can be extended on demand, frequently using compute resources of different physical data centers for a single service (Layer 2 extension).

EVPN's MP-BGP control plane enables you to dynamically move live virtual machines from one data center to another, also known as virtual machine (VM) motion. After you move a VM to a destination server or hypervisor, it transmits a gratuitous ARP, which updates the Layer 2 forwarding table of the PE device at the destination data center. The PE device then transmits a MAC route update to all remote PE devices which in turn update their forwarding tables. An EVPN tracks the movement of the VM, which is also known as MAC mobility.

EVPN also has mechanisms that detect and stop MAC flapping, and prevent the looping of broadcast, unknown unicast, and multicast (BUM) traffic in an all-active multi-homed topology.

The EVPN technology, similar to Layer 3 MPLS VPN, includes the concept of routing MAC addresses using IP/MPLS core. EVPN provides the following benefits:

- Ability to have an active multihomed edge device
- Aliasing

- Fast convergence
- Load balancing across dual-active links
- MAC address mobility
- Multitenancy

In addition, EVPN uses these techniques:

- *Multihoming* provides redundancy in the event that an access link or one of the PE routing devices fails. In either case, traffic flows from the CE device towards the PE router, using the remaining active links. For traffic in the other direction, the remote PE router updates its forwarding table to send traffic to the remaining active PE routers connected to the multihomed Ethernet segment. EVPN provides a fast convergence mechanism, which reduces traffic restoration time so that the time it takes to make this adjustment is independent of the number of media access control (MAC) addresses learned by the PE router. All-active multihoming enables a CE device to connect to two or more PE routers such that traffic is forwarded using all of the links between the devices. This multihoming enables the CE device to load-balance traffic to multiple PE routers. More importantly, multihoming enables a remote PE router to load-balance traffic to the multihomed PE routers across the core network. This load balancing of traffic flows between data centers is known as *aliasing*, which causes different signals to become indistinguishable—they become aliases of one another. Aliasing is used with digital audio and digital images.
- *Split horizon* prevents the looping of broadcast, unknown unicast, and multicast (BUM) traffic in a network. The split horizon basic principle is simple: Information about the routing for a particular packet is never sent back in the direction from which it was received.
- *Local link bias* conserves bandwidth by using local links to forward unicast traffic exiting a Virtual Chassis or Virtual Chassis Fabric (VCF) that has a link aggregation group (LAG) bundle composed of member links on different member switches in the same Virtual Chassis or VCF. A local link is a member link in the LAG bundle that is on the member switch that received the traffic.
- EVPN with VXLAN encapsulation is used for Layer 2 connectivity between virtual machines and a top-of-rack (TOR) switch, for example, a QFX5100 switch, within a Layer 2 domain.

You can use Contrail to provision an MX Series router as a Layer 2 or Layer 3 VXLAN gateway. MX Series routers implement the NETCONF XML management protocol, which is an XML-based protocol that client applications use to request and change configuration information on routing, switching, and security devices. The NETCONF XML management protocol uses an XML based data encoding for the configuration data and remote procedure calls. The NETCONF protocol defines basic operations that are equivalent to configuration mode commands in the command-line interface (CLI). Applications use the protocol operations to display, edit, and commit configuration statements similarly to how administrators use CLI configuration mode commands to perform those same operations.

Understanding VXLAN

Virtual extensible LANs (VXLANs) introduced an overlay scheme that expands the Layer 2 network address space from 4K to 16 million, largely solving the scaling issues seen in VLAN-based environments.

Network overlays are created by encapsulating traffic and tunneling the traffic over a physical network. You can use a number of tunneling protocols in the data center to create network overlays—the most common protocol is VXLAN. VXLAN tunneling protocol encapsulates Layer 2 Ethernet frames in Layer 3 UDP packets. This encapsulation enables you to create virtual Layer 2 subnets or segments that can span physical Layer 3 networks.

In a VXLAN overlay network, a VXLAN network identifier (VNI) uniquely identifies each Layer 2 subnet or segment. A VNI segments traffic the same way that an IEEE 802.1Q VLAN ID segments traffic. As is the case with VLAN, virtual machines on the same VNI can communicate directly with each other, whereas virtual machines on different VNIs need a router to communicate with each other.

The entity that performs the encapsulation and de-encapsulation is called a VXLAN tunnel endpoint (VTEP). In the physical network, a Juniper Networks device that functions as a Layer 2 or Layer 3 VXLAN gateway can encapsulate and de-encapsulate data packets. This type of VTEP is known as a hardware VTEP. In the virtual network, VTEPs can reside in hypervisor hosts, such as kernel-based virtual machine (KVM) hosts. This type of VTEP is known as a software VTEP.

Each VTEP has two interfaces.

- One interface is a switching interface that faces the virtual machines in the host and provides communication between VMs on the local LAN segment.
- The other is an IP interface that faces the Layer 3 network.

Each VTEP has a unique IP address that is used for routing the UDP packets between VTEPs. For example, when VTEP1 receives an Ethernet frame from VM1 addressed to VM3, it uses the VNI and the destination MAC to look up in its forwarding table which VTEP sends the packet to. It then adds a VXLAN header that contains the VNI to the Ethernet frame and encapsulates the frame in a Layer 3 UDP packet and routes the packet to VTEP2 over the Layer 3 network. VTEP2 de-encapsulates the original Ethernet frame and forwards it to VM3. VM1 and VM3 cannot detect the VXLAN tunnel and the Layer 3 network between them.

EVPN-VXLAN Integration Overview

VXLAN defines a tunneling scheme to overlay Layer 2 networks on top of Layer 3 networks. This tunneling scheme allows for optimal forwarding of Ethernet frames with support for multipathing of unicast and multicast traffic with the use of UDP/IP encapsulation for tunneling, and is mainly used for the intra-data center site connectivity.

A unique characteristic of EVPN is that MAC address learning between PE routers occurs in the control plane. The local PE router detects a new MAC address from a CE device and then, using MP-BGP, advertises the address to all the remote PE routers. This method differs from existing Layer 2 VPN solutions such as VPLS, which learn by flooding unknown unicast in the data plane. This control plane MAC learning method is the key enabler of the many useful features that EVPN provides.

Because MAC learning is handled in the control plane, EVPN has the flexibility to support different data plane encapsulation technologies between PE routers. This flexibility is important because not every backbone network might be running MPLS, especially in enterprise networks.

EVPN addresses many of the challenges faced by network operators building data centers to offer cloud and virtualization services. The main application of EVPN is Data Center Interconnect (DCI), which refers to the ability to extend Layer 2 connectivity between different data centers that are deployed to improve the performance of delivering application traffic to end users and for disaster recovery.

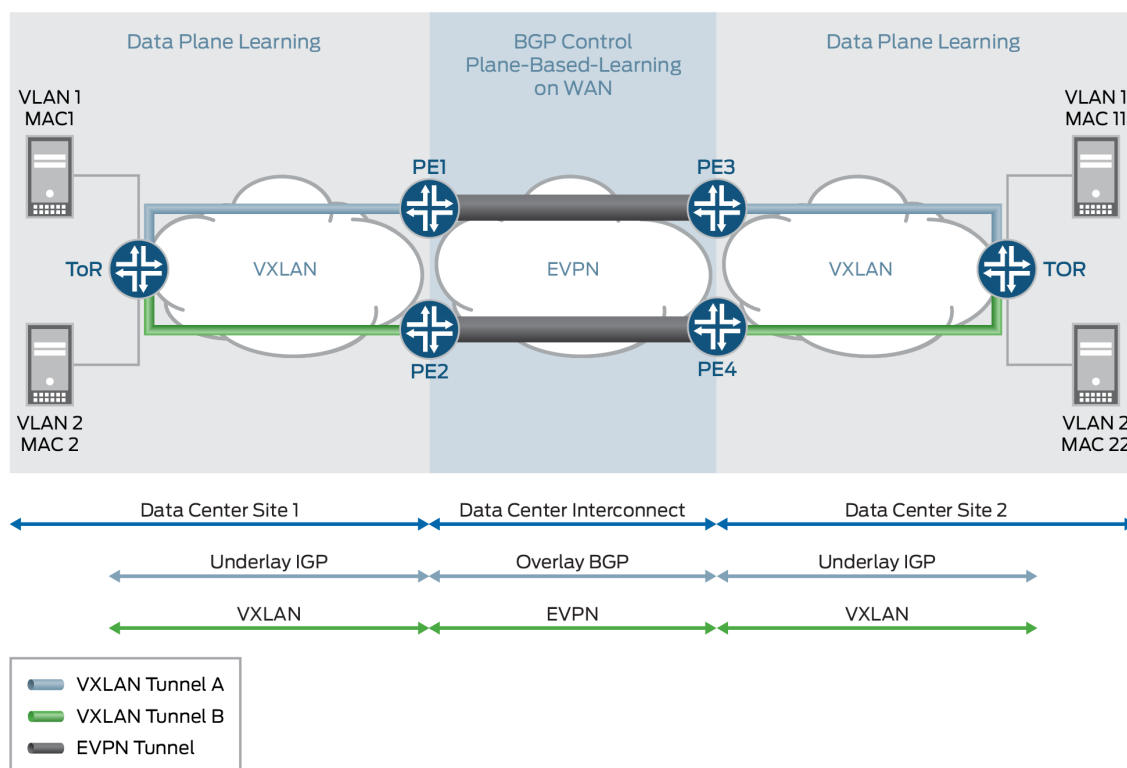
Although various DCI technologies are available, EVPN has an advantage over the other MPLS technologies because of its unique features, such as active/active redundancy, aliasing, and mass MAC withdrawal. As a result, to provide a solution for DCI, VXLAN is integrated with EVPN.

As shown in [Figure 45 on page 594](#), each VXLAN, which is connected to the MPLS or IP core, runs an independent instance of the interior gateway protocol (IGP) control plane. Each PE router participates in the IGP control plane instance of its VXLAN. Each customer is a data center, so each has its own virtual router for VXLAN underlay.

Each PE node can terminate the VXLAN data plane encapsulation where the VXLAN network identifier (VNI) is mapped to a bridge domain or VLAN. The PE router performs data plane learning on the traffic received from the VXLAN.

Each PE node implements EVPN to distribute the client MAC addresses learned over the VXLAN tunnel into BGP. Each PE node encapsulates the VXLAN or Ethernet frames with MPLS when sending the packets over the MPLS core and with the VXLAN tunnel header when sending the packets over the VXLAN network.

Figure 45: EVPN-VXLAN Integration Overview



Firewall Filtering and Policing Support for EVPN-VXLAN

For each firewall filter that you apply to a VXLAN, specify `family ethernet-switching` to filter Layer 2 (Ethernet) packets, or `family inet` to filter on IRB interfaces. The IRB interface acts as a Layer 3 routing interface to connect the VXLANs in one-layer or two-layer IP fabric topologies. The following limitations apply:

- Filtering and policing are not supported for VXLAN transit traffic.
- Firewall filtering on VNI at the egress VTEP device is not supported.
- Policing on VNI at the egress VTEP device is not supported.
- Match conditions against VXLAN header fields are not supported.



NOTE: EVPN-VXLAN firewall filters are configured on the interface after the VXLAN header is stripped by the VXLAN tunnel endpoint (VTEP).

For more information on configuring firewall filters, match conditions, and actions, see:

- [Configuring Firewall Filters](#)
- [Firewall Filter Match Conditions and Actions \(QFX and EX Series Switches\)](#)
- [Firewall Filter Match Conditions and Actions \(QFX10000 Switches\)](#)
- [Firewall Filters for EX Series Switches Overview](#)

Understanding Contrail Virtual Networks Use with EVPN-VXLAN

Juniper Networks Contrail virtualization software is a software-defined networking (SDN) solution that automates and orchestrates the creation of highly scalable virtual networks. These virtual networks enable you to harness the power of the cloud—for new services, increased business agility, and revenue growth. MX Series routers can use EVPN-VXLAN to provide both Layer 2 and Layer 3 connectivity for end stations within a Contrail virtual network (VN).

The Contrail software for virtual networks provides both Layer 2 and Layer 3 connectivity. With Contrail, Layer 3 routing is preferred over Layer 2 bridging whenever possible. Layer 3 routing is used through virtual routing and forwarding (VRF) tables between Contrail vRouters and physical MX Series routers. MX Series routers provide Layer 3 gateway functionality between virtual networks.

Contrail enables you to use EVPN-VXLAN when your network includes both virtual and bare-metal devices.

Two types of encapsulation methods are used in virtual networks.

- MPLS-over-GRE (generic routing encapsulation) is used for Layer 3 routing between Contrail and MX Series routers.
- EVPN-VXLAN is used for Layer 2 connectivity between virtual machines and top-of-rack (TOR) switches, for example, QFX5100 switches, within a Layer 2 domain. For Layer 2 connectivity, traffic load balancing in the core is achieved by using the multihoming all-active feature provided by EVPN. Starting with Junos OS Release 17.3R1, EX9200 switches also support EVPN-VXLAN with Contrail.



NOTE: MPLS core is not supported on switches—only MX Series routers support this feature.

You cannot simultaneously mix EVPN-VXLAN with Open vSwitch Database (OVSDB)-VXLAN on QFX Series switches. After a switch is set to OVSDB-managed, the controller treats all ports as managed by OVSDB.

EVPN-VXLAN Support for VXLAN Underlays on MX Series Routers and the EX9200 Line of Switches

MX Series routers and the EX92xx line of switches support Virtual Extensible LAN (VXLAN) gateways. Each VXLAN gateway supports the following functionalities:

- Switching functionality with traditional Layer 2 networks and VPLS networks
- Inter-VXLAN routing and VXLAN-only bridging domain with IRB
- Virtual switches
- VXLAN with VRF functionality
- Configurable load balancing
- Statistics for remote VTEP

Starting in Junos OS Release 17.3R1, EVPN-VXLAN support on MX Series routers is extended to VXLAN gateway implementation using an IPv6 underlay. We support EVPN Type 1, Type 2, Type 3, and Type 4 routes with an IPv6 underlay on MX Series routers.

We support the following service types with the IPv6 underlay support:

- VLAN-based service
- VLAN-bundle service
- Port-based service
- VLAN-aware service

Both IPv4 and IPv6 EVPN-VXLAN underlays support EVPN Type 2 MAC addresses with IP address advertisement and proxy MAC addresses with IP address advertisement.

EVPN-VXLAN Support for VXLAN Underlays on QFX Series Switches

QFX Series switches support VXLAN gateways in an EVPN-VXLAN network. All devices that support EVPN-VXLAN can use an IPv4 underlay for the VXLAN overlay.

We also support configuring an IPv6 underlay for the VXLAN overlay in EVPN-VXLAN networks on QFX Series switches. You can only configure an IPv6 underlay using MAC-VRF EVPN instances. With an IPv6 underlay, the outer IP header in the VXLAN packet is an IPv6 header, and you configure the VTEP source address as an IPv6 address. See ["EVPN-VXLAN with an IPv6 Underlay" on page 897](#) for more about IPv6 underlay support and how to configure a VXLAN gateway device to use an IPv6 underlay.

EVPN-VXLAN Support for VXLAN Underlays on ACX Series Devices

ACX7100-32C, ACX7100-48L, and ACX7024 devices can use either an IPv4 or IPv6 underlay for the VXLAN overlay. You can create an IPv6 underlay only with MAC-VRF routing instances (all service types). You must configure either an IPv4 or an IPv6 underlay across the EVPN instances in the fabric; you can't mix IPv4 and IPv6 underlays in the same fabric.

To create an IPv6 underlay, you need to enable the `vlan-extended` statement at the `[edit system packet-forwarding-options system-profile]` hierarchy.

With an IPv6 underlay, the outer IP header in the VXLAN packet is an IPv6 header, and you configure the VTEP source address as an IPv6 address. See ["EVPN-VXLAN with an IPv6 Underlay" on page 897](#) for more about IPv6 underlay support and how to configure a VXLAN gateway device to use an IPv6 underlay.

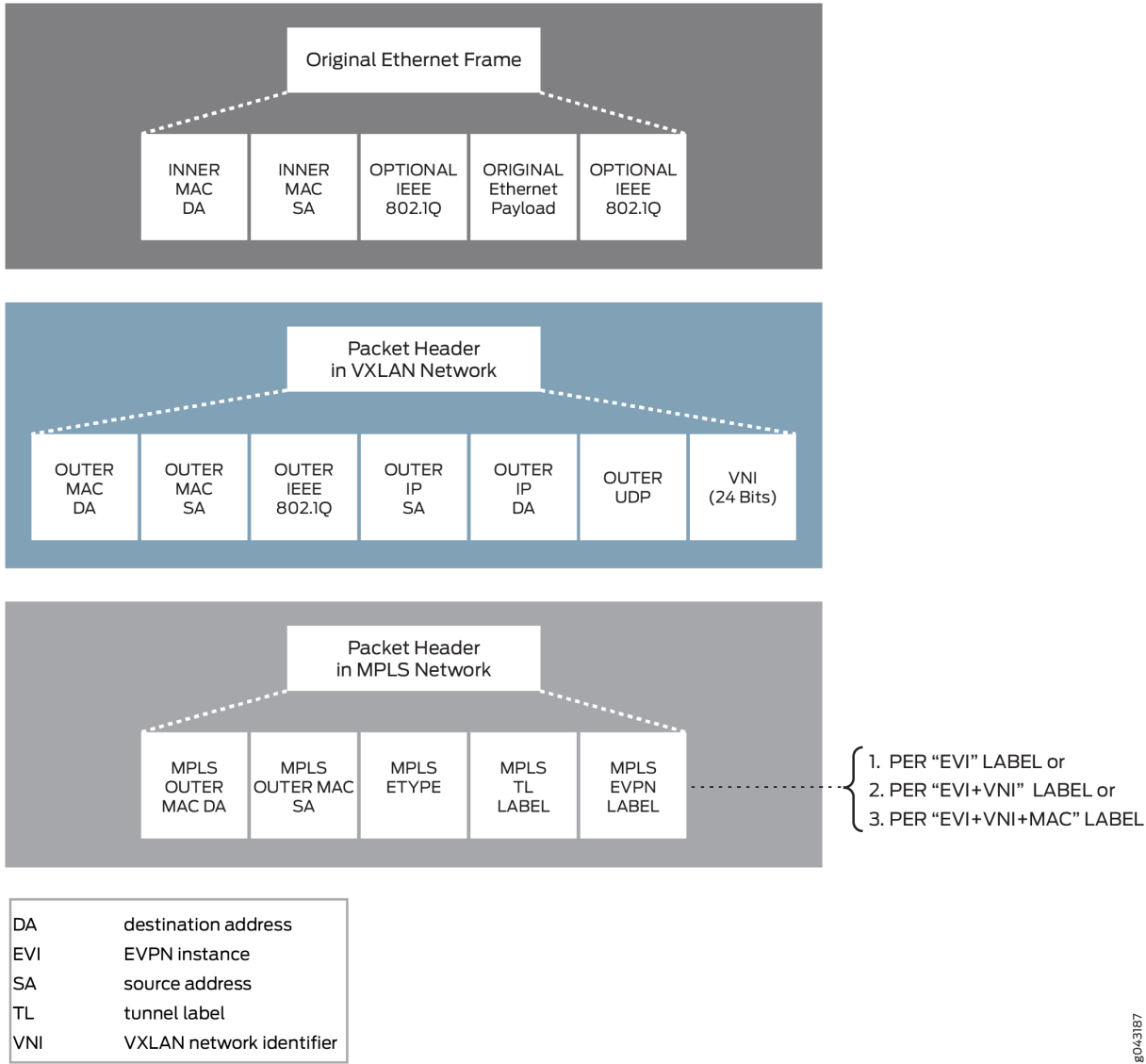
After you enable this profile, the Packet Forwarding Engine restarts. Traffic might drop if any traffic is running.

To go back to using the default system profile, issue the `delete system packet-forwarding-options system-profile vlan-extended` command. The PFE restarts after you revert to the default system profile. During this process, any traffic that's running might drop.

EVPN-VXLAN Packet Format

The EVPN-VXLAN packet format is shown in [Figure 46 on page 598](#).

Figure 46: EVPN-VXLAN Packet Format



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
22.3R1	Starting in Junos OS Evolved Release 22.3R1, QFX5130-32CD and QFX5700 switches support configuring an IPv6 underlay for the VXLAN overlay in an EVPN-VXLAN network.

21.4R1	Starting in Junos OS Releases 21.2R2 and 21.4R1, the QFX10000 line of switches and QFX5120 switches support configuring an IPv6 underlay for the VXLAN overlay in an EVPN-VXLAN network.
17.3R1	Starting with Junos OS Release 17.3R1, EX9200 switches also support EVPN-VXLAN with Contrail.
17.3R1	Starting in Junos OS Release 17.3R1, EVPN-VXLAN support on MX Series is extended to VXLAN gateway implementation using an IPv6 underlay.

RELATED DOCUMENTATION

[EVPN Multihoming Overview | 162](#)

[Understanding EVPN Pure Type 5 Routes | 50](#)

[Understanding VXLANs | 606](#)

[EVPN-over-VXLAN Supported Functionality | 599](#)

[EVPN Over VXLAN Encapsulation Configuration Overview for QFX Series and EX4600 Switches | 628](#)

EVPN-over-VXLAN Supported Functionality

IN THIS SECTION

- [VXLAN Encapsulation | 600](#)
- [EVPN BGP Routes and Attributes | 600](#)
- [EVPN Multihoming Procedure | 601](#)
- [Next-Hop Forwarding | 603](#)
- [Control Plane MAC Learning Method | 603](#)
- [Contrail vRouters and the L3-VRF Table | 604](#)
- [Designated Forwarder Election | 604](#)

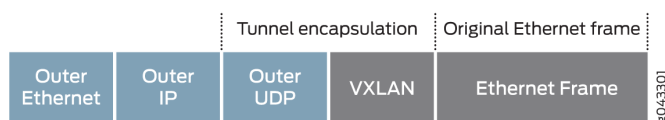
The following functionality is supported for EVPN-over-VXLAN data plane encapsulation:

VXLAN Encapsulation

EVPN supports the VXLAN data plane encapsulation type within the same EVPN instance. To support the VXLAN encapsulation type, all EVPN PE devices specify the tunnel type as VXLAN in the BGP encapsulation extended community. The ingress PE determines which encapsulation types to use based on its own encapsulation capability and the one its EVPN remote PE device advertises.

When EVPN is used as overlay solution for the underlay IP network with VXLAN encapsulation, the data packet is encapsulated with a VXLAN header at the ingress PE device, and the data packet is de-encapsulated from the VXLAN header at the egress PE device. [Figure 47 on page 600](#) shows the packet format when a packet is forwarded in the core through the underlay IP network with VXLAN encapsulation:

Figure 47: Underlay IP Network with VXLAN Encapsulation



If the underlay network uses the IPv4 protocol and IPv4 addressing, the outer IP header in the VXLAN packet is an IPv4 header. All platforms that support EVPN-VXLAN work with an IPv4 underlay network.

On some platforms, you can configure the underlay to use the IPv6 protocol and IPv6 addressing. In those cases, the outer IP header in the VXLAN packet is an IPv6 header, and you configure the VTEP source address as an IPv6 address. See ["EVPN-VXLAN with an IPv6 Underlay" on page 897](#) for details on using an IPv6 underlay.

EVPN BGP Routes and Attributes

EVPN BGP routes and attributes support EVPN-over-VXLAN data plane encapsulation and are affected as follows:

- By attaching the BGP encapsulation extended community attribute with tunnel encapsulation type VXLAN in the EVPN MAC routes, the egress PE device advertises the EVPN inclusive multicast route and EVPN per EVI route autodiscovery.
- The Ethernet tag fields for all EVPN BGP routes are set to zero to support VXLAN network identifier (VNI)-based mode only.
- The VNI, also referred to as the VNI field in the EVPN overlay, is placed in the MPLS fields for the EVPN MAC route, the EVPN inclusive multicast route, and per EVPN instance autodiscovery route.

- The Ethernet segment identifier label field is set to zero for the EVPN per Ethernet segment autodiscovery route because no Ethernet segment identifier label exists for the supported VXLAN encapsulation type.
- All corresponding EVPN routes from the remote PE device are resolved over the inet.0 or :vxlan.inet.0 table instead of the inet.3 table for IPv4. When you use an IPv6 underlay for EVPN-VXLAN tunneling, the same is true for the inet6.0 and :vxlan.inet6.0 tables.

EVPN Multihoming Procedure

You can multihome a bare-metal server (BMS) to a pair of Juniper Networks top-of-rack (TOR) switches, for example, QFX5100 switches, through a Layer 2 link aggregation group (LAG) interface. Because of the LAG interface, only one copy of BUM traffic is forwarded from a BMS to the core router. To prevent a Layer 2 loop and flooding of BUM traffic back to the same Ethernet segment to which the multihomed BMS is attached, the BUM traffic applies the multihomed active-active split-horizon filtering rule. To fully support the EVPN multihomed active-active mode feature, the Juniper Networks TOR switch also advertises the EVPN aliasing route to other EVPN PE devices.

The lack of MPLS label support for EVPN over IP with VXLAN data plane encapsulation impacts the following EVPN multihomed active-active mode functions:



NOTE: EVPN over VXLAN does not support active-standby mode of operation. You can configure only active-active mode using the `all-active` option in the ESI interface configuration. Single-homing Ethernet segments with EVPN using VXLAN encapsulation with the `single-active` option is not supported.

Starting in Junos OS Release 22.2R1, EVPN adds all-active redundancy, aliasing, and mass MAC withdrawal support, including integration of data plane VXLAN, to provide resilient connectivity between data centers to its established Data Center Interconnect (DCI) technologies. This new support builds an end-to-end DCI solution by integrating EVPN multicast with data plane VXLAN.

In an active-active topology, both links are used for load-balancing the traffic. Unicast traffic originating from the EVPN-MPLS domain is load-balanced to both gateways and is forwarded through the VXLAN tunnel to the remote VXLAN end router. Load-balanced packets can come from either of the gateways and cause a MAC flip-flop issue on the VXLAN end router. You can overcome this MAC flip-flop issue by configuring an anycast IP address as a secondary address on the loopback interfaces of the gateways. When you set the anycast address as the `vxlan-source-ip`, the VXLAN tunnel will be created to the anycast address and the MAC will be learned from the anycast address of the VTEP.

Use the following statements to set active-active redundancy at the ESI level and an anycast address on the lo0 interface. Add a secondary address with an anycast IP to the lo0 interface and include it as the

VTEP interface *vxlan-source-ip* in the routing-instance and the *secondary-vtep-address* under protocols pim.

```
set interfaces lo0 unit 0 esi identifier
set interfaces lo0 unit 0 esi all-active
set interfaces lo0 unit 0 family inet address anycast-ip-address
set interfaces lo0 unit 0 family inet address primary-ip-address primary

set protocols pim secondary-vtep-address anycast-ip-address

set routing-instances <name> vtep-source-interface lo0.0
set routing-instances <name> vtep-source-interface inet evpn-mpls-encap vxlan-source-ip anycast-ip-address
```

Local Bias and Split-Horizon Filtering Rule

Because of the missing MPLS label, the split-horizon filtering rule for the multihomed Ethernet segment is modified and based on the IP address of the EVPN PE device instead of the MPLS Ethernet segment label. For traffic coming from the access interface, any traffic forwarding to the multihomed Ethernet segment is based on the local bias for EVPN with VXLAN data plane encapsulation. Each EVPN PE device tracks the IP address of its peer multihomed EVPN PE device for which it shares the same Ethernet segment. This tracking provides the source VTEP IP address (in the outer IP header) for each VXLAN packet received from the other EVPN PE device. The split-horizon filtering rule is enforced on both ingress and egress PE devices for multi-destination traffic:

- Ingress PE—Responsible for forwarding multi-destination packets coming from any of its directly-attached access interfaces to its remaining associated multihomed Ethernet segments, regardless of the subject ingress PE device's designated forwarder (DF) election status.
- Egress PE—Allows no forwarding of any multi-destination packets to the same multihomed Ethernet segment to which an egress PE shares with its ingress PE device, regardless of the subject egress PE device's DF election status.

Aliasing

When you connect a BMS to a pair of Juniper Networks TOR switches through a LAG interface bundle, only one of the switches can learn the local MAC address. To support the load balancing of known unicast traffic coming from the VXLAN to the BMS between the switches, the EVPN PE device on the switch must advertise EVPN per EVPN instance autodiscovery route. This advertisement signals to the remote EVPN PE devices that the MAC learned from the multihomed Ethernet segment from its peer multihomed PE device is also reachable. For VXLAN encapsulation, there is no change to the EVPN procedure when providing the EVPN aliasing function.

Next-Hop Forwarding

The Layer 2 address learning daemon (l2ald) creates the VXLAN encapsulation composite next-hop at ingress, and the VXLAN de-encapsulation composite next-hop at the egress. The VXLAN encapsulation composite next-hop forwards Layer 2 unicast traffic to the remote PE device with the VXLAN encapsulation. If multiple paths are available to reach a remote MAC (as with the multihomed EVPN active-active case), the routing protocol daemon (rpd) informs the l2ald of all the remote VTEP IP addresses associated with the remote MAC. The l2ald is responsible for building an equal-cost multipath (ECMP) next hop for the remote MAC. The VXLAN de-encapsulation composite next-hop de-encapsulates the VXLAN tunnel header at the egress and then forwards the traffic to the BMS.

For known unicast traffic:

- At ingress, the rpd does not need to add a label route in the mpls.0 table.
- At egress, the rpd does not need to add a label route to point to the table next-hop in the mpls.0 table.

Control Plane MAC Learning Method

A unique characteristic of EVPN is that MAC address learning between PE devices occurs in the control plane. The local PE router detects a new MAC address from a CE device and then, using MP-BGP, advertises the address to all the remote PE devices. This method differs from existing Layer 2 VPN solutions such as VPLS, which learn by flooding unknown unicast in the data plane. This control plane MAC learning method is a crucial component of the features and benefits that EVPN provides. Because MAC learning is handled in the control plane, EVPN has the flexibility to support different data plane encapsulation technologies between PE devices. This flexibility is important because not every backbone network may be running MPLS, especially in enterprise networks.

For control plane remote MAC learning, because the l2ald creates the VXLAN encapsulation composite next-hop and VXLAN de-encapsulation composite next-hop, the rpd no longer creates an indirect next-hop used in the MAC forwarding table. The rpd relies on the existing mechanism to inform the l2ald of the remote MAC addresses learned from the control plane. Instead of informing the l2ald about the VLAN ID and the indirect next-hop index used by the remote MAC in the MAC FIB table, the rpd informs the l2ald about the remote VTEP IP address and VXLAN network identifier. The control plane remote MAC learned route points to VXLAN encapsulation composite next-hop, or an ECMP next-hop created by the l2ald in the MAC forwarding table.

For multihomed EVPN active-active, a pair of remote VTEP IP addresses are associated with the remote MAC address. The remote VTEP IP addresses are obtained from the MAC route received either from the remote PE device, or from the aliasing route from the remote PE device. When a remote PE device withdraws a MAC route or the aliasing route for the Ethernet segment from where the MAC learned, the rpd alerts the l2ald about the change to the pair of remote VTEP IP addresses accordingly. As a result, the l2ald updates the uni-list next-hop built for this MAC. When both remote MAC routes and its

associated aliasing route are withdrawn or become unresolved, the rpd informs the l2ald about the deletion of this remote MAC and the l2ald withdraws this MAC from the MAC forwarding table.

Contrail vRouters and the L3-VRF Table

The Contrail virtualization software creates virtual networks (VNs) associated with routes in a Layer 3 virtual routing and forwarding (L3-VRF) table.

The following are the associated routes:

Subnet Routes for an IRB Interface

For each pair of MAC-(virtual routing and forwarding) VRF and L3-VRF tables created for a virtual network on an MX Series router, a corresponding IRB interface is associated with the MAC-VRF and L3-VRF table pair. The L3-VRF table on the MX Series router has the subnet route from the IRB interface associated with its local virtual networks, as well as, all subnet routes of the IRB interfaces associated with other virtual networks within a data center provided by the Junos OS routing leaking feature. The MX Series routers advertise these subnet routes through MP-BGP to the Contrail control nodes. As a result, the L3-VRF table in the Contrail vRouter contains the same set of subnet routes for the IRB interfaces in its IP FIB, and the subnet routes point their next hops to the MX Series routers.

Virtual Machine Host Routes

The Contrail vRouters support proxy ARP and advertise the IP address with the EVPN MAC route for its virtual machine (VM). For both Contrail vRouters and MX Series routers, the L3-VRF table for a virtual network contains all of the VM host routes for those VMs that reside in the same virtual network, as well as, routes in all other virtual networks. Intra- virtual network and inter- virtual network traffic between the VMs is forwarded directly at Layer 3 between the Contrail vRouters.

Bare-Metal Server Host Routes

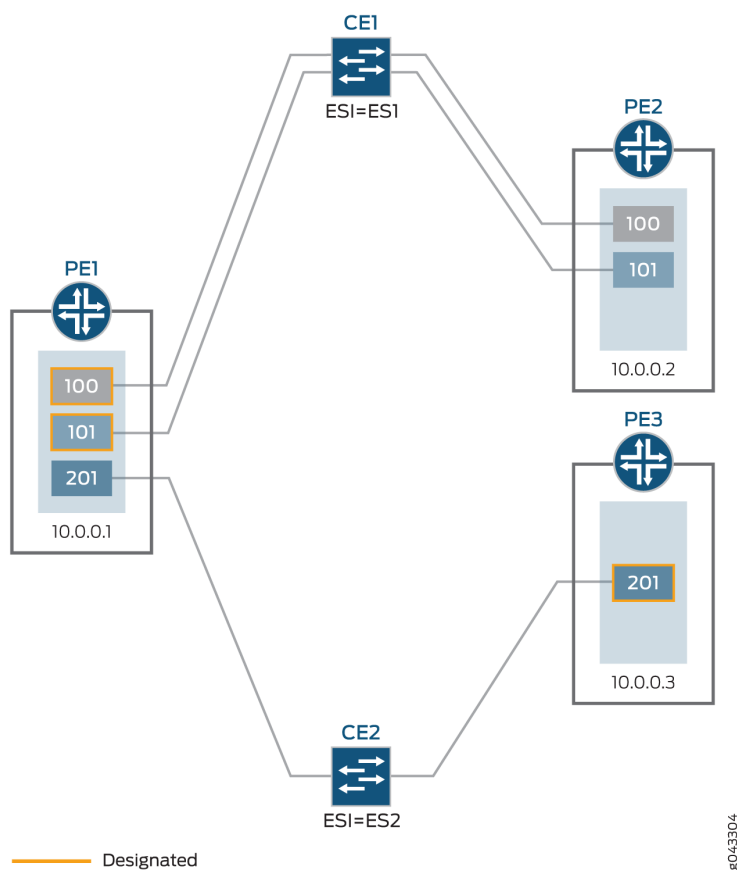
A Juniper Networks TOR switch, for example, a QFX5100 switch, does not advertise the IP address with the EVPN MAC route for the bare-metal server (BMS) to which it is attached. As a result of the EVPN MAC route advertisement from the switch, no BMS host route is installed in the L3-VRF table on Contrail vRouters and MX Series routers. However, ARP routes for the BMSs are installed in the kernel on the MX Series router if the MX Series router receives ARP responses from the BMSs.

Designated Forwarder Election

To provide better load balance and more flexible topology, the election of the designated forwarder is determined by selecting the minimum VLAN ID or VXLAN network ID for each Ethernet segment

instead of selecting based on the EVPN instance. [Figure 48 on page 605](#) shows a sample designated forwarder election topology.

Figure 48: Designated Forwarder Election Topology



CE device (CE1) has an Ethernet segment identifier value configured equal to ES1 and is connected to both PE1 and PE2 devices, with configured VLANs 100 and 101. Router PE1 is elected as the designated forwarder for the two VLANs.

CE device (CE2) has an Ethernet segment identifier value configured equal to ES2 and is connected to both PE1 and PE3 devices, with configured VLAN 201. Router PE3 is elected as the designated forwarder for this VLAN.

The Ethernet tag ID can be either of the following:

- VLAN ID (for EVPN-MPLS)
- VXLAN network ID (for EVPN-VXLAN)

RELATED DOCUMENTATION

[Understanding EVPN with VXLAN Data Plane Encapsulation | 588](#)

[EVPN Over VXLAN Encapsulation Configuration Overview for QFX Series and EX4600 Switches | 628](#)

[Using a Default Layer 3 Gateway to Route Traffic in an EVPN-VXLAN Overlay Network | 741](#)

Understanding VXLANs

IN THIS SECTION

- [VXLAN Benefits | 607](#)
- [How Does VXLAN Work? | 607](#)
- [VXLAN Implementation Methods | 608](#)
- [Using QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, EX4300-48MP, and EX4600 Switches with VXLANs | 609](#)
- [Changing the UDP Port on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches | 610](#)
- [Host Entry Overflow Prevention | 610](#)
- [Controlling Transit Multicast Traffic on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches | 612](#)
- [Using an MX Series Router, EX9200 Switch, or QFX10000 Switch as a VTEP | 612](#)
- [Manual VXLANs Require PIM | 613](#)
- [Load Balancing VXLAN Traffic | 614](#)
- [Enabling QFX5120 Switches to Tunnel Traffic on Core-Facing Layer 3 Tagged and IRB Interfaces | 614](#)
- [Using ping and traceroute with a VXLAN | 615](#)
- [Supported VXLAN Standards | 615](#)

Virtual Extensible LAN protocol (VXLAN) technology allows networks to support more VLANs. According to the IEEE 802.1Q standard, traditional VLAN identifiers are 12 bits long—this naming limits networks to 4094 VLANs. The VXLAN protocol overcomes this limitation by using a longer logical network identifier that allows more VLANs and, therefore, more logical network isolation for large networks such as clouds that typically include many virtual machines.

VXLAN Benefits

VXLAN technology allows you to segment your networks (as VLANs do), but it provides benefits that VLANs cannot. Here are the most important benefits of using VXLANs:

- You can theoretically create as many as 16 million VXLANs in an administrative domain (as opposed to 4094 VLANs on a Juniper Networks device).
- MX Series routers and EX9200 switches support as many as 32,000 VXLANs, 32,000 multicast groups, and 8000 virtual tunnel endpoints (VTEPs). This means that VXLANs based on MX Series routers provide network segmentation at the scale required by cloud builders to support very large numbers of tenants.
- QFX10000 Series switches support 4000 VXLANs and 2000 remote VTEPs.
- QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 switches support 4000 VXLANs, 4000 multicast groups, and 2000 remote VTEPs.
- EX4300-48MP switches support 4000 VXLANs.
- You can enable migration of virtual machines between servers that exist in separate Layer 2 domains by tunneling the traffic over Layer 3 networks. This functionality allows you to dynamically allocate resources within or between data centers without being constrained by Layer 2 boundaries or being forced to create large or geographically stretched Layer 2 domains.

Using VXLANs to create smaller Layer 2 domains that are connected over a Layer 3 network means that you do not need to use Spanning Tree Protocol (STP) to converge the topology but can use more robust routing protocols in the Layer 3 network instead. In the absence of STP, none of your links are blocked, which means you can get full value from all the ports that you purchase. Using routing protocols to connect your Layer 2 domains also allows you to load-balance the traffic to ensure that you get the best use of your available bandwidth. Given the amount of east-west traffic that often flows within or between data centers, maximizing your network performance for that traffic is very important.

The video *Why Use an Overlay Network in a Data Center?* presents a brief overview of the advantages of using VXLANs.



Video: [Why Use an Overlay Network in a Data Center?](#)

How Does VXLAN Work?

VXLAN is often described as an overlay technology because it allows you to stretch Layer 2 connections over an intervening Layer 3 network by encapsulating (tunneling) Ethernet frames in a VXLAN packet that includes IP addresses. Devices that support VXLANs are called *virtual tunnel endpoints (VTEPs)*—they can be end hosts or network switches or routers. VTEPs encapsulate VXLAN traffic and de-encapsulate that traffic when it leaves the VXLAN tunnel. To encapsulate an Ethernet frame, VTEPs add a number of fields, including the following fields:

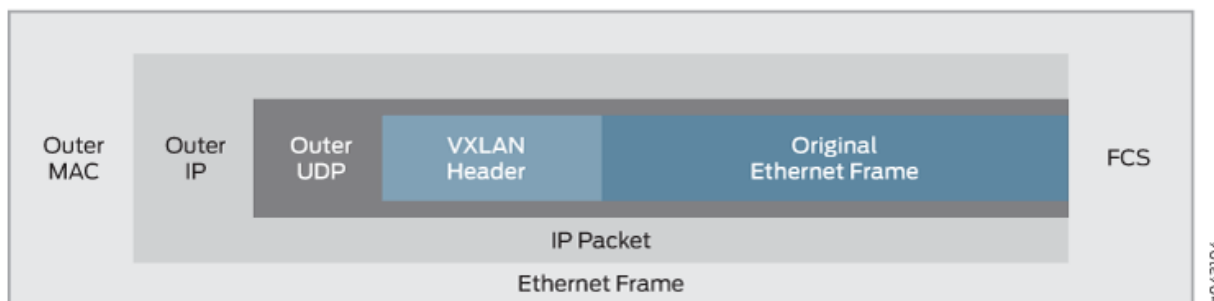
- Outer media access control (MAC) destination address (MAC address of the tunnel endpoint VTEP)
- Outer MAC source address (MAC address of the tunnel source VTEP)
- Outer IP destination address (IP address of the tunnel endpoint VTEP)
- Outer IP source address (IP address of the tunnel source VTEP)
- Outer UDP header
- A VXLAN header that includes a 24-bit field—called the *VXLAN network identifier (VNI)*—that is used to uniquely identify the VXLAN. The VNI is similar to a VLAN ID, but having 24 bits allows you to create many more VXLANs than VLANs.



NOTE: Because VXLAN adds 50 to 54 bytes of additional header information to the original Ethernet frame, you might want to increase the MTU of the underlying network. In this case, configure the MTU of the physical interfaces that participate in the VXLAN network, not the MTU of the logical VTEP source interface, which is ignored.

Figure 49 on page 608 shows the VXLAN packet format.

Figure 49: VXLAN Packet Format



VXLAN Implementation Methods

Junos OS supports implementing VXLANs in the following environments:

- **Manual VXLAN**—In this environment, a Juniper Networks device acts as a transit device for downstream devices acting as VTEPs, or a gateway that provides connectivity for downstream servers that host virtual machines (VMs), which communicate over a Layer 3 network. In this environment, software-defined networking (SDN) controllers are not deployed.



NOTE: QFX10000 switches do not support manual VXLANs.

- **OVSDB-VXLAN**—In this environment, SDN controllers use the Open vSwitch Database (OVSDB) management protocol to provide a means through which controllers (such as a VMware NSX or Juniper Networks Contrail controller) and Juniper Networks devices that support OVSDB can communicate.
- **EVPN-VXLAN**—In this environment, Ethernet VPN (EVPN) is a control plane technology that enables hosts (physical servers and VMs) to be placed anywhere in a network and remain connected to the same logical Layer 2 overlay network, and VXLAN creates the data plane for the Layer 2 overlay network.

Using QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, EX4300-48MP, and EX4600 Switches with VXLANs

You can configure the switches to perform all of the following roles:

- (All switches except EX4300-48MP) In an environment without an SDN controller, act as a transit Layer 3 switch for downstream hosts acting as VTEPs. In this configuration, you do not need to configure any VXLAN functionality on the switch. You do need to configure IGMP and PIM so that the switch can form the multicast trees for the VXLAN multicast groups. (See "[Manual VXLANs Require PIM](#)" on page 613 for more information.)
- (All switches except EX4300-48MP) In an environment with or without an SDN controller, act as a Layer 2 gateway between virtualized and nonvirtualized networks in the same data center or between data centers. For example, you can use the switch to connect a network that uses VXLANs to one that uses VLANs.
- (EX4300-48MP switches) Act as a Layer 2 gateway between virtualized and nonvirtualized networks in a campus network. For example, you can use the switch to connect a network that uses VXLANs to one that uses VLANs.
- (All switches except EX4300-48MP) Act as a Layer 2 gateway between virtualized networks in the same or different data centers and allow virtual machines to move (VMotion) between those networks and data centers. For example, if you want to allow VMotion between devices in two different networks, you can create the same VLAN in both networks and put both devices on that VLAN. The switches connected to these devices, acting as VTEPs, can map that VLAN to the same VXLAN, and the VXLAN traffic can then be routed between the two networks.
- (QFX5110 and QFX5120 switches with EVPN-VXLAN) Act as a Layer 3 gateway to route traffic between different VXLANs in the same data center.

- (QFX5110 and QFX5120 switches with EVPN-VXLAN) Act as a Layer 3 gateway to route traffic between different VXLANs in different data centers over a WAN or the Internet using standard routing protocols or virtual private LAN service (VPLS) tunnels.



NOTE: If you want a QFX5110 or QFX5120 switch to be a Layer 3 VXLAN gateway in an EVPN-VXLAN environment, you must configure integrated routing and bridging (IRB) interfaces to connect the VXLANs, just as you do if you want to route traffic between VLANs.

Because the additional headers add 50 to 54 bytes, you might need to increase the MTU on a VTEP to accommodate larger packets. For example, if the switch is using the default MTU value of 1514 bytes and you want to forward 1500-byte packets over the VXLAN, you need to increase the MTU to allow for the increased packet size caused by the additional headers.

Changing the UDP Port on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches

Starting with Junos OS Release 14.1X53-D25 on QFX5100 switches, Junos OS Release 15.1X53-D210 on QFX5110 and QFX5200 switches, Junos OS Release 18.1R1 on QFX5210 switches, and Junos OS Release 18.2R1 on EX4600 switches, you can configure the UDP port used as the destination port for VXLAN traffic. To configure the VXLAN destination port to be something other than the default UDP port of 4789, enter the following statement:

```
set protocols l2-learning destination-udp-port port-number
```

The port you configure will be used for all VXLANs configured on the switch.



NOTE: If you make this change on one switch in a VXLAN, you must make the same change on all the devices that terminate the VXLANs configured on your switch. If you do not do so, traffic will be disrupted for all the VXLANs configured on your switch. When you change the UDP port, the previously learned remote VTEPs and remote MACs are lost and VXLAN traffic is disrupted until the switch relearns the remote VTEPs and remote MACs.

Host Entry Overflow Prevention

IN THIS SECTION

- [CLI Commands | 611](#)

By default, when the host table reaches its capacity, additional host entries overflow into the Longest Prefix Match (LPM) table. This can cause a degradation of routing performance that results from smaller host entries consuming valuable space in the LPM table.

Starting with Junos OS Release 25.2R1, you can prevent host entries from spilling over into the LPM table by configuring the `set forwarding-options no-host-as-lpm` statement. When you enable this option, host entries are blocked from entering the LPM table, and an error message is logged to notify you of the full host table condition. This restriction maintains the LPM table's integrity for larger subnet routes, which are typically more critical for efficient network operation. This configuration requires a PFE restart to clear any existing host entries from the LPM table, ensuring that the table is dedicated solely to subnet routes moving forward. The restart is crucial for maintaining a clean state, as it prevents any residual host entries from affecting table space utilization.



NOTE: Configuring the `set forwarding-options no-host-as-lpm` statement restarts the PFE in standalone devices. Virtual Chassis (VC) devices require a manual restart. Restarting the PFE removes any previously learned host entries from the LPM table.

Host entry overflow prevention is particularly beneficial for environments with high host entry demands. The feature supports both IPv4 and IPv6 routes and applies to VXLAN and non-VXLAN environments, making it versatile for various deployment scenarios. Segregating host and subnet routes avoids the degradation of routing performance that results from host entries spilling over to the LPM table. Additionally, the system generates a system log message each time you toggle the CLI option, providing an audit trail for change management and troubleshooting. By leveraging this feature, you can enhance routing table management, improve routing efficiency, and ensure systematic control over routing table entries.

Please refer to [Feature Explorer](#) for a complete list of the products that support the host entry overflow prevention feature.

CLI Commands

To configure the host entry overflow prevention feature, use the following CLI command:

```
set forwarding-options no-host-as-lpm
```

This command enables the feature that blocks host entries from being added to the LPM table when the host table is full, preserving LPM space for larger subnet routes.

To verify the status of the host entry overflow prevention feature, use the command:

```
show forwarding-options no-host-as-lpm
```

This command displays the current configuration status, indicating whether the feature is enabled or disabled.

To display the a summary of the routes in the Packet Forwarding Engine Host and LPM forwarding tables, use the command:

```
show pfe route summary hw
```

By using these commands, you can effectively manage and monitor the routing table entries, ensuring optimal performance and scalability in your network environment.

Controlling Transit Multicast Traffic on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches

When the switch acting as a VTEP receives a broadcast, unknown unicast, or multicast packet, it performs the following actions on the packet:

1. It de-encapsulates the packet and delivers it to the locally attached hosts.
2. It then adds the VXLAN encapsulation again and sends the packet to the other VTEPs in the VXLAN.

These actions are performed by the loopback interface used as the VXLAN tunnel address and can, therefore, negatively impact the bandwidth available to the VTEP. Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, Junos OS Release 15.1X53-D210 for QFX5110 and QFX5200 switches, Junos OS Release 18.1R1 for QFX5210 switches, and Junos OS Release 18.2R1 for EX4600 switches, if you know that there are no multicast receivers attached to other VTEPs in the VXLAN that want traffic for a specific multicast group, you can reduce the processing load on the loopback interface by entering the following statement:

```
set protocols l2-learning disable-vxlan-multicast-transit vxlan-multicast-group multicast-group
```

In this case, no traffic will be forwarded for the specified group but all other multicast traffic will be forwarded. If you do not want to forward any multicast traffic to other VTEPs in the VXLAN, enter the following statement:

```
set protocols l2-learning disable-vxlan-multicast-transit vxlan-multicast-group all
```

Using an MX Series Router, EX9200 Switch, or QFX10000 Switch as a VTEP

You can configure an MX Series router, EX9200 switch, or QFX10000 switch to act as a VTEP and perform all of the following roles:

- Act as a Layer 2 gateway between virtualized and nonvirtualized networks in the same data center or between data centers. For example, you can use an MX Series router to connect a network that uses VXLANs to one that uses VLANs.

- Act as a Layer 2 gateway between virtualized networks in the same or different data centers and allow virtual machines to move (VMotion) between those networks and data centers.
- Act as a Layer 3 gateway to route traffic between different VXLANs in the same data center.
- Act as a Layer 3 gateway to route traffic between different VXLANs in different data centers over a WAN or the Internet using standard routing protocols or virtual private LAN service (VPLS) tunnels.



NOTE: If you want one of the devices described in this section to be a VXLAN Layer 3 gateway, you must configure integrated routing and bridging (IRB) interfaces to connect the VXLANs, just as you do if you want to route traffic between VLANs.

Manual VXLANs Require PIM

In an environment with a controller (such as a VMware NSX or Juniper Networks Contrail controller), you can provision VXLANs on a Juniper Networks device. A controller also provides a control plane that VTEPs use to advertise their reachability and learn about the reachability of other VTEPs. You can also manually create VXLANs on Juniper Networks devices instead of using a controller. If you use this approach, you must also configure Protocol Independent Multicast (PIM) on the VTEPs so that they can create VXLAN tunnels between themselves.

You must also configure each VTEP in a given VXLAN to be a member of the same multicast group. (If possible, you should assign a different multicast group address to each VXLAN, although this is not required. Multiple VXLANs can share the same multicast group.) The VTEPs can then forward ARP requests they receive from their connected hosts to the multicast group. The other VTEPs in the group de-encapsulate the VXLAN information, and (assuming they are members of the same VXLAN) they forward the ARP request to their connected hosts. When the target host receives the ARP request, it responds with its MAC address, and its VTEP forwards this ARP reply back to the source VTEP. Through this process, the VTEPs learn the IP addresses of the other VTEPs in the VXLAN and the MAC addresses of the hosts connected to the other VTEPs.

The multicast groups and trees are also used to forward broadcast, unknown unicast, and multicast (BUM) traffic between VTEPs. This prevents BUM traffic from being unnecessarily flooded outside the VXLAN.



NOTE: Multicast traffic that is forwarded through a VXLAN tunnel is sent only to the remote VTEPs in the VXLAN. That is, the encapsulating VTEP does not copy and send copies of the packets according to the multicast tree—it only forwards the received multicast packets to the remote VTEPs. The remote VTEPs de-encapsulate the encapsulated multicast packets and forward them to the appropriate Layer 2 interfaces.

Load Balancing VXLAN Traffic

The Layer 3 routes that form VXLAN tunnels use per-packet load balancing by default, which means that load balancing is implemented if there are ECMP paths to the remote VTEP. This is different from normal routing behavior in which per-packet load balancing is not used by default. (Normal routing uses per-prefix load balancing by default.)

The source port field in the UDP header is used to enable ECMP load balancing of the VXLAN traffic in the Layer 3 network. This field is set to a hash of the inner packet fields, which results in a variable that ECMP can use to distinguish between tunnels (flows).

None of the other fields that flow-based ECMP normally uses are suitable for use with VXLANs. All tunnels between the same two VTEPs have the same outer source and destination IP addresses, and the UDP destination port is set to port 4789 by definition. Therefore, none of these fields provide a sufficient way for ECMP to differentiate flows.

Enabling QFX5120 Switches to Tunnel Traffic on Core-Facing Layer 3 Tagged and IRB Interfaces



NOTE: This section applies only to QFX5120 switches running Junos OS Releases 18.4R1, 18.4R2, 18.4R2-S1 through 18.4R2-S3, 19.1R1, 19.1R2, 19.2Rx, and 19.3Rx.

When a QFX5120 switch attempts to tunnel traffic on core-facing Layer 3 tagged interfaces or IRB interfaces, the switch drops the packets. To avoid this issue, you can configure a simple two-term filter-based firewall on the Layer 3 tagged or IRB interface.



NOTE: QFX5120 switches support a maximum of 256 two-term filter-based firewalls.

For example:

```
set interfaces et-0/0/3 unit 0 family inet filter input vxlan100
set firewall family inet filter vxlan100 term 1 from destination-address 192.168.0.1/24 then
accept
set firewall family inet filter vxlan100 term 2 then routing-instance route1
```

Term 1 matches and accepts traffic that is destined for the QFX5210 switch, which is identified by the source VTEP IP address (192.168.0.1/24) assigned to the switch's loopback interface. For term 1, note that when specifying an action, you can alternatively count traffic instead of accepting it.

Term 2 matches and forwards all other data traffic to a routing instance (route 1), which is configured interface et-0/0/3.

In this example, note that interface `et-0/0/3` is referenced by routing instance `route1`. As a result, you must include the `set firewall family inet filter vxlan100 term 2 then routing-instance route1` command. Without this command, the firewall filter will not work properly.

Using ping and traceroute with a VXLAN

On QFX5100 and QFX5110 switches, you can use the `ping` and `traceroute` commands to troubleshoot traffic flow through a VXLAN tunnel by including the `overlay` parameter and various options. You use these options to force the ping or traceroute packets to follow the same path as data packets through the VXLAN tunnel. In other words, you make the underlay packets (ping and traceroute) take the same route as the overlay packets (data traffic). See *ping overlay* and *traceroute overlay* for more information.

Supported VXLAN Standards

RFCs and Internet drafts that define standards for VXLAN:

- RFC 7348, *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*
- Internet draft draft-ietf-nvo3-vxlan-gpe, *Generic Protocol Extension for VXLAN*

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
25.2R1	Starting with Junos OS Release, you can prevent host routes from spilling over into the LPM table.
14.1X53-D30	Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, Junos OS Release 15.1X53-D210 for QFX5110 and QFX5200 switches, Junos OS Release 18.1R1 for QFX5210 switches, and Junos OS Release 18.2R1 for EX4600 switches, if you know that there are no multicast receivers attached to other VTEPs in the VXLAN that want traffic for a specific multicast group, you can reduce the processing load on the loopback interface
14.1X53-D25	Starting with Junos OS Release 14.1X53-D25 on QFX5100 switches, Junos OS Release 15.1X53-D210 on QFX5110 and QFX5200 switches, Junos OS Release 18.1R1 on QFX5210 switches, and Junos OS Release 18.2R1 on EX4600 switches, you can configure the UDP port used as the destination port for VXLAN traffic.

RELATED DOCUMENTATION

Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches

[Understanding EVPN with VXLAN Data Plane Encapsulation](#) | 588

VXLAN Constraints on EX Series, QFX Series, PTX Series, and ACX Series Devices

IN THIS SECTION

- [VXLAN Constraints on QFX5xxx, EX4100, EX4100-F, EX4300-48MP, EX4400, and EX4600 Switches | 616](#)
- [VXLAN Constraints on QFX10000 Series Switches | 625](#)
- [VXLAN Constraints on PTX10000 Series Routers | 626](#)
- [VXLAN Constraints on ACX Series Routers | 626](#)
- [VXLAN Constraints on All Devices | 627](#)

When configuring Virtual Extensible LANs (VXLANs) on QFX Series and EX Series switches, be aware of the constraints described in the following sections. In these sections, “Layer 3 side” refers to a network-facing interface that performs VXLAN encapsulation and de-encapsulation, and “Layer 2 side” refers to a server-facing interface that is a member of a VLAN that is mapped to a VXLAN.

VXLAN Constraints on QFX5xxx, EX4100, EX4100-F, EX4300-48MP, EX4400, and EX4600 Switches

- (QFX5130-32CD and QFX5700 switches) We don't support a centrally routed bridging (CRB) architecture using a collapsed spine model when:
 - An interface uses both enterprise and service provider style configurations at the `edit interfaces interface-name` hierarchy.
 - Multiple access ports on the same VXLAN bridge domain use a mix of enterprise and service provider styles.

Enterprise Style Configuration

```
set interfaces interface-name unit 0 family ethernet-switching interface-mode trunk
set interfaces interface-name unit 0 family ethernet-switching vlan members vlan-name
```

Service Provider Style Configuration

```
set interfaces interface-name vlan-tagging
set interfaces interface-name encapsulation extended-vlan-bridge
set interfaces interface-name unit unit vlan-id vlan-id
```

Flexible Ethernet Services Configuration

```
set interfaces interface-name flexible-vlan-tagging
set interfaces interface-name encapsulation flexible-ethernet-services
set interfaces interface-name unit unit encapsulation vlan-bridge
set interfaces interface-name unit unit vlan-id vlan-id
set interfaces interface-name unit unit family ethernet-switching interface-mode trunk
set interfaces interface-name unit unit family ethernet-switching vlan members vlan-name
```

Supported scenarios:

- A non-collapsed spine and no VLAN is configured with a local interface.
- A collapsed spine and some VLAN's on the device are configured without a local interface, while some VLAN's on the device are configured using enterprise style on CE-facing interfaces.
- A collapsed spine and all VLAN's on the device are configured with flexible-ethernet-services encapsulation on a physical interface, while multiple logical interfaces are configured with either enterprise or service provider style.

Unsupported scenarios:

- A collapsed spine and any local interface on the device includes all configured VLAN's. The workaround for this scenario is to use flexible-ethernet-services encapsulation on the physical interface.
- A collapsed spine where some VLAN's on the device are not configured on any local interface, and some VLAN's are configured using service provider style on CE-facing interfaces. The workaround for this scenario is to configure a `vlan-id`.

- A collapsed spine where some VLAN's on the device are not part of any interface, and some VLAN's are configured using `flexible-ethernet-services` encapsulation on multiple, logical interfaces, using enterprise style. The workaround for this scenario is to configure a `vlan-id`.
- VXLAN support on a Virtual Chassis or Virtual Chassis Fabric (VCF) has the following constraints and recommendations:
 - We support EVPN-VXLAN on an EX4300-48MP Virtual Chassis only in campus networks.
 - Standalone EX4400 switches and EX4400 Virtual Chassis support EVPN-VXLAN. For multihoming use cases, hosts can be multihomed to standalone EX4400 switches, but we don't support multihoming a host with ESI-LAG interfaces to EX4400 Virtual Chassis.
 - Standalone EX4100 (EX4100 and EX4100-F) switches and EX4100 Virtual Chassis (EX4100 and EX4100-F) support EVPN-VXLAN. For multihoming use cases, hosts can be multihomed to standalone EX4100 switches, but we don't support multihoming a host with ESI-LAG interfaces to EX4100 Virtual Chassis.
 - In data center networks, we support EVPN-VXLAN only on a Virtual Chassis or VCF consisting of all QFX5100 switches, and not on any other mixed or non-mixed Virtual Chassis or VCF. We support VCF in EVPN-VXLAN data center environments running Junos OS releases starting in 14.1X53-D40 and prior to 17.1R1. However, we don't recommend using EVPN-VXLAN on a QFX5100 Virtual Chassis or VCF because the feature support is at parity only with support on standalone QFX5100 switches running Junos OS Release 14.1X53-D40.

We have deprecated VCF support in general from Junos OS Release 21.4R1 onward.

- When a QFX5100 Virtual Chassis learns a MAC address on a VXLAN interface, the MAC table entry can possibly take up to 10 to 15 minutes to age out (two to three times the 5-minute default aging interval). This happens when the Virtual Chassis learns a MAC address from an incoming packet on one Virtual Chassis member switch, and then must forward that packet over the Virtual Chassis port (VCP) links to another member switch in the Virtual Chassis on the way to its destination. The Virtual Chassis marks the MAC address as seen again at the second member switch, so the MAC address might not age out for one or two additional aging intervals beyond the first one. MAC learning can't be turned off on VXLAN interfaces only at the second Virtual Chassis member switch, so you can't avoid the extra delay in this case.
- (QFX5120 switches only) Traffic that is tunneled via a core-facing Layer 3 tagged interface or IRB interface is dropped. To avoid this limitation, you can configure flexible VLAN tagging. For more information, see ["Understanding VXLANs" on page 606](#).
- (QFX5110 and QFX5120) If you configure an interface in the enterprise style with flexible Ethernet services encapsulation, the device drops transit Layer 2 VXLAN-encapsulated packets on that interface. To work around this issue, configure the interface in the service provider style instead of using the enterprise style. For more information on enterprise style and service provider style configurations, see *Flexible Ethernet Services Encapsulation*. For an overview of configuring flexible

Ethernet services in an EVPN-VXLAN fabric, see ["Understanding Flexible Ethernet Services Support With EVPN-VXLAN" on page 654](#).

- (QFX5110 and QFX5120 switches) In EVPN-VXLAN fabrics, we don't support enterprise style, service provider style, and native VLAN configurations on the same physical interface if the native VLAN is the same as one of the VLANs in the service provider style configuration. The native VLAN can be one of the VLANs in the enterprise style configuration. For more information on enterprise style and service provider style configurations, see *Flexible Ethernet Services Encapsulation*.
- (QFX5xxx switches) In an EVPN-VXLAN fabric, you can't configure the *native-vlan-id* statement on the same interface where you enable VLAN translation with the *vlan-rewrite* statement.
- (QFX5100, QFX5110, QFX5200, and QFX5210 switches) We support VXLAN configuration in the default-switch routing instance and in MAC VRF routing instances (*instance-type mac-vrf*).

(EX4300-48MP and EX4600 switches) We support VXLAN configuration only in the default-switch routing instance.

- (QFX5100, QFX5200, QFX5210, EX4300-48MP, and EX4600 switches) Routing traffic between different VXLANs is not supported.



NOTE: The following switches support VXLAN routing as of the indicated releases, so this limitation no longer applies:

- EX4300-48MP switches: Starting in Junos OS Release 19.4R1.
- QFX5210 switches: Starting in Junos OS Release 21.3R1.

- (QFX5100, QFX5110, QFX5120, EX4600, and EX4650 switches) These switches support only one VTEP next hop on VXLAN underlay IRB interfaces. If you don't want to use multiple egress ports but need more than one VTEP next hop, as a workaround you can do one of the following:
 - Place a router between the switch and the remote VTEPs so only one next hop is between them.
 - Use physical Layer 3 interfaces instead of IRB interfaces for remote VTEP reachability.
- (QFX5110 switches) By default, routing traffic between a VXLAN and a Layer 3 logical interface—for example, an interface configured with the `set interfaces interface-name unit logical-unit-number family inet address ip-address/prefix-length` command—is not enabled. If this routing functionality is required in your EVPN-VXLAN network, you can perform some additional configuration to make it work. For more information, see ["Understanding How to Configure VXLANs and Layer 3 Logical Interfaces to Interoperate" on page 651](#).
- Integrated routing and bridging (IRB) interfaces used in EVPN-VXLAN overlay networks do not support the IS-IS routing protocol.

- (QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, EX4300-48MP, and EX4600 switches) A physical interface cannot be a member of a VLAN and a VXLAN. That is, an interface that performs VXLAN encapsulation and de-encapsulation cannot also be a member of a VLAN. For example, if a VLAN that is mapped to a VXLAN is a member of trunk port xe-0/0/0, any other VLAN that is a member of xe-0/0/0 must also be assigned to a VXLAN.



NOTE: Starting in Junos OS Releases 18.1R3 and 18.4R1 for QFX5100, QFX5110, QFX5200, and QFX5210 switches and Junos OS Release 19.4R1 for QFX5120 and EX4650 switches, this limitation no longer applies because you can configure flexible Ethernet services on the physical interface. (The same is true for aggregated Ethernet (AE) bundle interfaces.)

Also, starting in Junos OS Release 20.3R1 on QFX5110 and QFX5120 switches, we support Layer 2 VLAN and VXLAN logical interfaces on the same physical interface using service provider style interface configurations only.

For more information, see ["Understanding Flexible Ethernet Services Support With EVPN-VXLAN" on page 654.](#)

- (QFX5110, QFX5120, EX4100, and EX4400 switches) We don't support VXLAN and non-VXLAN logical interfaces on the same physical interface using enterprise style interface configurations.
- (QFX5120, EX4100, EX4300-48MP, EX4400, and EX4650 switches) With 802.1X authentication for VXLAN traffic on an access port, upon authentication, the RADIUS server dynamically switches the traffic from the original VLAN to a dynamic VLAN you configure as a VXLAN-enabled VLAN. A VXLAN-enabled VLAN is a VLAN for which you configure a VXLAN network identifier (VNI) mapping using the `vlan vni vni` statement at the `[edit vlans vlan-name]` hierarchy.

When you configure the 802.1X dynamic VLAN and its corresponding VNI mapping, you must also configure the original VLAN as a VXLAN-enabled VLAN with a VNI mapping. If you don't explicitly configure the port as a member of a VLAN, the port uses the default VLAN. In that case, you must configure the default VLAN as a VXLAN-enabled VLAN with a VNI mapping.

Also, in releases before Junos OS Release 22.2R3-S3, when any dynamic VLAN is assigned to a port, that VLAN must already have been statically configured on another port on the device. Starting in Junos OS Release 22.2R3-S3, we no longer have this constraint.

See *802.1X Authentication* and *RADIUS Server Configuration for Authentication* for more on dynamic VLAN assignment with a RADIUS server.

- Multichassis link aggregation groups (MC-LAGs) are not supported with VXLAN.



NOTE: In an EVPN-VXLAN environment, EVPN multihoming active-active mode is used instead of MC-LAG for redundant connectivity between hosts and leaf devices.

- IP fragmentation and defragmentation are not supported on the Layer 3 side.
- The following features are not supported on the Layer 2 side:
 - (QFX5100, QFX5200, QFX5210, EX4300-48MP, and EX4600 switches) IGMP snooping with EVPN-VXLAN.
 - Redundant trunk groups (RTGs).
 - The ability to shut down a Layer 2 interface or temporarily bring down the interface when a storm control level is exceeded is not supported.
 - We don't support full STP, MSTP, RSTP, or VSTP (xSTP) features with VXLAN. However, you can configure xSTP on edge (access port) for BPDU block-on-edge support. See *BPDU Protection for Spanning-Tree Protocols* for details.
- Access port security features such as the following are not supported with VXLAN:
 - DHCP snooping.
 - Dynamic ARP inspection.
 - MAC limiting and MAC move limiting.

Some exceptions include:

- MAC limiting is supported on OVSDB-managed interfaces in an OVSDB-VXLAN environment with Contrail controllers. For more information, see *Features Supported on OVSDB-Managed Interfaces*.
- On these devices serving as L2 VXLAN gateways in EVPN-VXLAN centrally-routed bridging overlay networks:
 - EX4300 multigigabit switches starting in Junos OS Release 19.4R1
 - EX4400 switches starting in Junos OS Release 21.1R1
 - EX4400 multigigabit switches starting in Junos OS Release 21.2R1
 - EX4100 and EX4100-F switches starting in Junos OS Release 22.3R1
 - EX4100 multigigabit switches starting in Junos OS Release 22.3R1

we support these access security features on Layer 2 access-side interfaces that are associated with VXLAN-mapped VLANs:

- DHCPv4 and DHCPv6 snooping
- Dynamic ARP inspection (DAI)
- Neighbor discovery inspection (NDI)
- IPv4 and IPv6 source guard
- Router advertisement (RA) guard

We support these features only on single-homed access interface connections.

- Ingress node replication is not supported in the following cases:
 - When PIM is used for the control plane (manual VXLAN).
 - When an SDN controller is used for the control plane (OVSDB-VXLAN).

Ingress node replication is supported with EVPN-VXLAN.

- PIM-BIDIR and PIM-SSM are not supported with VXLANs.
- If you configure a port-mirroring instance to mirror traffic exiting from an interface that performs VXLAN encapsulation, the source and destination MAC addresses of the mirrored packets are invalid. The original VXLAN traffic is not affected.
- (QFX5110 switches only) VLAN firewall filters are not supported on IRB interfaces on which EVPN-VXLAN is enabled.
- (EX4650 and QFX5000 Series switches) Firewall filter and policer support:
 - (QFX5100 switches) Firewall filters and policers are not supported on transit traffic on which EVPN-VXLAN is enabled. They are supported only in the ingress direction on CE-facing interfaces.
 - (QFX5100 switches) For IRB interfaces in an EVPN-VXLAN one-layer IP fabric, firewall filtering and policing is supported only at the ingress point of non-encapsulated frames routed through the IRB interface.
 - (EX4650, QFX5110, and QFX5120 switches) We support ingress filtering and policing for routed VXLAN interfaces (IRB interfaces) as ingress route Access Control Lists [IRACLs].
 - (QFX5110, QFX5120, and QFX5210 switches) We support Ingress filtering and policing on non-routed VXLAN interfaces as ingress port ACLs [IPACLs]).
 - (QFX5110 and QFX5120 switches) Filtering and policing support on non-routed VXLAN interfaces extends to the egress direction as egress Port ACLs ([EPACLs]).
- (EX4300-48MP switches only) The following styles of interface configuration are not supported:

- Service provider style, where a physical interface is divided into multiple logical interfaces, each of which is dedicated to a particular customer VLAN. The `extended-vlan-bridge` encapsulation type is configured on the physical interface.
- Flexible Ethernet services, which is an encapsulation type that enables a physical interface to support both service provider and enterprise styles of interface configuration.

For more information about these styles of interface configuration, see [Flexible Ethernet Services Encapsulation](#).

- (QFX5100 switches only) Using the `no-arp-suppression` configuration statement, you can turn off the suppression of ARP requests on one or more specified VLANs. However, starting in Junos OS Release 18.4R3, you must turn off this feature on all VLANs. To do so, you can use one of these configuration options:
 - Use a batch command to turn off the feature on all VLANs—`set groups group-name vlans * no-arp-suppression`. With this command, we use the asterisk (*) as a wildcard that specifies all VLANs.
 - Use a command to turn off the feature on each individual VLAN—`set vlans vlan-name no-arp-suppression`.



NOTE: The `no-arp-suppression` statement is hidden in the Junos OS CLI, but can still be configured when required.

- QFX5120-48Y, QFX5120-32C, and QFX5200 switches support hierarchical equal-cost multipath (ECMP), which allows these switches to perform a two-level route resolution. However, all other QFX5xxx switches do not support hierarchical ECMP. As a result, when an EVPN Type-5 packet is encapsulated with a VXLAN header, then de-encapsulated by a QFX5xxx switch that does not support hierarchical ECMP, the switch is unable to resolve the two-levels of routes that were in the inner packet. The switch then drops the packet.
- (QFX5100, QFX5110, QFX5120, QFX5200, and QFX5210 switches) When you configure the IRB interfaces on a device that is concurrently supporting configured VLANs on both a centrally-routed bridging overlay and an edge-routed bridging overlay, the IRB MAC address and virtual gateway MAC address on the IRB interface for the centrally-routed bridging overlay must be different from the IRB MAC address and virtual gateway MAC address on the IRB interface for the edge-routed bridging overlay.
- (QFX5110 and QFX5120 switches only) In an EVPN-VXLAN overlay, we do not support running a routing protocol on an IRB interface that is in a default routing instance (`default.inet.0`). Instead, we recommend including the IRB interface in a routing instance of type `vrf`.
- The following constraints apply to both VXLANs and VLANs.

- (QFX5xxx switches only) When configuring route leaking between virtual routing and forwarding (VRF) instances, you must specify each prefix with a subnet mask that is equal to or longer than /16 for IPv4 prefixes or equal to or longer than /64 for IPv6 prefixes. If a subnet mask that you specify does not meet these parameters, the specified routes will not be leaked.
- (QFX5120 switches only) By default, QFX5120 switches allocate 5 MB of a shared buffer to absorb bursts of multicast traffic. If a burst of multicast traffic exceeds 5 MB, the switch will drop the packets that the switch receives after the buffer space is exceeded. To prevent the switch from dropping multicast packets when this situation occurs, you can do one of the following:
 - Use the [shared-buffer](#) configuration statement in the [edit class-of-service] hierarchy level to reallocate a higher percentage of the shared buffer for multicast traffic. For more information about fine-tuning a shared buffer, see [Understanding CoS Buffer Configuration](#).
 - On the Juniper Networks switch from which the bursts of multicast traffic are coming, apply a shaper on the egress link.
- (QFX5xxx switches only) If you have enabled storm control on an interface, you might notice a significant difference between the configured and actual traffic rates for the interface. This difference is the result of an internal storm control meter that quantifies the actual traffic rate for the interface in increments of 64 kbps, for example, 64 kbps, 128 kbps, 192 kbps, and so on.
- (QFX5xxx and EX46xx switches) You can't use hardware-assisted inline Bidirectional Forwarding Detection (BFD) with VXLAN encapsulation of BFD packets. Also, if you configure EVPN overlay BGP peerings, use distributed BFD instead of hardware-assisted inline BFD. See *Understanding How BFD Detects Network Failures* for details on the different types of BFD sessions that you can configure.
- (QFX5130-32CD, QFX5130E-32CD, QFX5130-48C, QFX5700, and QFX5700E switches) Starting in Junos OS Evolved Releases 25.2X100-D10 and 25.4R1, we support hardware-assisted inline BFD with 100 x 3 millisecond timers over VXLAN tunnels on the listed platforms only. This support applies to:
 - Type 2 IPv4 or IPv6 L2 and L3 multihop BFD with ECMP or multihomed VTEPs with these requirements:
 - 100 x 3 ms timers
 - Overlay BGP peering between loopbacks
 - BFD configured on the overlay BGP sessions
 - Type 5 IPv4 or IPv6 multihop BFD with ECMP
 - Pure Type 5 routing instances

- (QFX5xxx and EX46xx switches) We don't support configuring and using MPLS and EVPN-VXLAN at the same time on the same device. We have this constraint because Broadcom-based platforms use the same hardware tables to store tunnel and virtual port information used by both the MPLS and VXLAN feature sets.

Also, you can't use an MPLS underlay with EVPN and a VXLAN overlay; this is a Broadcom hardware limitation.

- (QFX5xxx and EX46xx switches) We don't support tagged L3 interfaces and L2 bridge domains as subunits of the same interface with an IRB in service provider style configurations.

VXLAN Constraints on QFX10000 Series Switches

- MC-LAGs are not supported with VXLAN.



NOTE: In an EVPN-VXLAN environment, EVPN multihoming active-active mode is used instead of MC-LAG for redundant connectivity between hosts and leaf devices.

- IP fragmentation is not supported on the Layer 3 side.
- The following features are not supported on the Layer 2 side:
 - IGMP snooping with EVPN-VXLAN in Junos OS Releases before Junos OS Release 17.2R1.
 - STP (any variant).
- Access port security features are not supported with VXLAN. For example, the following features are not supported:
 - DHCP snooping.
 - Dynamic ARP inspection.
 - MAC limiting and MAC move limiting.
- Ingress node replication is not supported when an SDN controller is used for the control plane (OVSDB-VXLAN). Ingress node replication is supported for EVPN-VXLAN.
- QFX10000 switches that are deployed in an EVPN-VXLAN environment do not support an IPv6 physical underlay network.
- When the next-hop database on a QFX10000 switch includes next hops for both the underlay network and the EVPN-VXLAN overlay network, the next hop to a VXLAN peer cannot be an Ethernet segment identifier (ESI) or a virtual tunnel endpoint (VTEP) interface.
- IRB interfaces used in EVPN-VXLAN overlay networks do not support the IS-IS routing protocol.

- VLAN firewall filters applied to IRB interfaces on which EVPN-VXLAN is enabled.
- Filter-based forwarding (FBF) is not supported on IRB interfaces used in an EVPN-VXLAN environment.
- QFX10002, QFX10008, and QFX10016 switches do not support port mirroring and analyzing when EVPN-VXLAN is also configured.
- When you configure the IRB interfaces on a device that is concurrently supporting configured VLANs on both a centrally-routed bridging overlay and an edge-routed bridging overlay, the IRB MAC address and virtual gateway MAC address on the IRB interface for the centrally-routed bridging overlay must be different from the IRB MAC address and virtual gateway MAC address on the IRB interface for the edge-routed bridging overlay.
- In an EVPN-VXLAN overlay, we do not support running a routing protocol on an IRB interface that is in a default routing instance (default.inet.0). Instead, we recommend including the IRB interface in a routing instance of type `vrf`.
- In an EVPN-VXLAN environment, we don't support configuring anycast gateways with the `default-gateway` statement and the `advertise` statement on links participating in the same Ethernet segment (ES).
- You must configure class of service (CoS) rewrite rules to have the differentiated services code point (DSCP) bits copied from the inner VXLAN header to the outer VXLAN header.

VXLAN Constraints on PTX10000 Series Routers

- You must enable tunnel termination globally on PTX10K series devices in EVPN-VXLAN deployments, as follows:

```
set forwarding-options tunnel-termination
```

This option is disabled by default.

- You can't use a firewall filter on these devices to block VXLAN tunnel termination on specific ports, but you can use the following command to block VXLAN tunnel termination for a port:

```
set interfaces logical-interface-name unit n family inet/inet6 no-tunnel-termination
```

Adding the `no-tunnel-termination` option disables tunnel termination for all traffic on the specific port where it is configured.

VXLAN Constraints on ACX Series Routers

- Multihoming peers within a DC should be from a similar product family. We do not recommend mixing ACX series with QFX series, PTX series, or MX series for multihoming.

- (ACX 7xxx routers) Networks with both MAC and IP scale should configure `set system packet-forwarding-options hw-db-profile cloud-metro`. The default is `lean-edge`, which is intended for IP scale.
- (ACX 7xxx routers) Load balancing is not enabled by default. Shown here is one sample configuration. Configure only those options necessary for your deployment.

```
set forwarding-options hash-key family inet layer-3
set forwarding-options hash-key family inet layer-4
set forwarding-options hash-key family inet6 layer-3
set forwarding-options hash-key family inet6 layer-4
set forwarding-options hash-key family multiservice source-mac
set forwarding-options hash-key family multiservice destination-mac
set policy-options policy-statement <statement name> then load-balance per-packet
```

- (ACX 7xxx routers) Configure `set system packet-forwarding-options system-profile vxlan-extended` to support an EVPN-VXLAN IPv6 underlay.
- (ACX 7xxx routers) Configure `set system packet-forwarding-options system-profile vxlan-stitching` to support EVPN-VLXAN to EVPN-VXLAN stitching, and EVPN-VXLAN to EVPN-MPLS stitching with `no-control-word` support.
- (ACX 7xxx routers) Configure `set system packet-forwarding-options system-profile vxlan-stitching control-word` to support control word functions in a stitched EVPN-VXLAN to EVPN-MPLS network. Refer to *vxlan-stitching* for additional information regarding control word support.
- (ACX 7xxx routers) One user defined Virtual Gateway Address (VGA) IPv4 MAC (`virtual-gateway-v4-mac`), and one user defined VGA IPv6 MAC (`virtual-gateway-v6-mac`) will be supported system wide.
- (ACX 7xxx routers) Configure `set system packet-forwarding-options no-ip-tos-rewrite` to propagate DSCP information from payload to VXLAN router header during VXLAN encapsulation.
- (ACX 7xxx routers) ESI configuration in EVPN multihoming mode is supported per physical interface device, or per LAG interface only.
- (ACX 7xxx routers) IRB interfaces are not supported as EVPN-VXLAN underlay networks.
- (ACX 7xxx routers) For EVPN-VXLAN to EVPN-MPLS stitching, devices which do not support forwarding based on bridge-domain labels will not be compatible with ACX series devices as DC peer GW's.

VXLAN Constraints on All Devices

- If you configure multiple sub-units on a port with an ESI, we don't support the disable operation (`set interfaces logical-interface-name unit n disable`) on those logical interfaces.

RELATED DOCUMENTATION

- [Understanding VXLANs | 606](#)
- Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches*
- Manually Configuring VXLANs on QFX Series and EX4600 Switches*

EVPN Over VXLAN Encapsulation Configuration Overview for QFX Series and EX4600 Switches

This topic provides information about configuring Ethernet VPN (EVPN) with Virtual Extensible Local Area Networks (VXLAN) data plane encapsulation on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 switches.

The configuration statements described in this topic are in the `switch-options` and `protocols evpn` hierarchy levels.

CLI Statement	Description
<code>route-distinguisher</code>	Specifies an identifier attached to a route—this enables you to distinguish to which VPN or VPLS the route belongs. Each routing instance must have a unique route distinguisher associated with it. The route distinguisher is used to place bounds around a VPN so that the same IP address prefixes can be used in different VPNs without having them overlap.
<code>vrf-target</code>	Specifies a VRF target community. In effect, this statement configures a single policy for import and a single policy for export to replace the per-VRF policies for every community. <code>import</code> , <code>export</code> , and <code>auto</code> options are available.

(Continued)

CLI Statement	Description
<code>vrf-import</code>	Specifies how routes are imported into the VRF table of the local provider edge (PE) router or switch from the remote PE.
<code>vrf-export</code>	Specifies how routes are exported from the local PE router's VRF table to the remote PE router.
<code>designated-forwarder-election-hold-time</code>	Establishes when a designated forwarder is required for customer edge (CE) devices that are multihomed to more than one provider edge (PE) device. Without a designated forwarder, multihomed hosts receive duplicate packets. Designated forwarders are chosen for an Ethernet segment identifier (ESI) based on type-4 route advertisements.
<code>encapsulation vxlan</code>	Configures a VXLAN encapsulation type.
<code>extended-vni-list</code> and <code>extended-vni-all</code>	Establishes which VXLAN virtual network identifiers are designated as part of the virtual switch instance.
<code>multicast-mode</code>	Configures the multicast server mode for delivering traffic and packets for EVPN.

(Continued)

CLI Statement	Description
<code>vni-options</code>	Configures different route targets for each VXLAN virtual network identifier instance under <code>vni-options</code> .
<code>show route table</code>	Displays both imported EVPN routes, and export/import EVPN routes for the default-switch routing instance.
<code>show configuration protocols evpn</code>	Displays results of the <code>extended-vni-list</code> and <code>vni-options</code> statements.

RELATED DOCUMENTATION

[EVPN Multihoming Overview | 162](#)

[Understanding VXLANs | 606](#)

[EVPN-over-VXLAN Supported Functionality | 599](#)

[Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric Using MX Routers as Spines | 808](#)

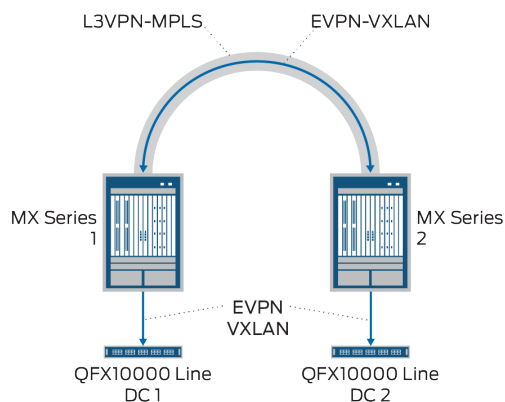
Implementing EVPN-VXLAN for Data Centers

Although there are various Data center interconnect (DCI) technologies available, EVPN has an added advantage over other MPLS technologies because of its unique features, such as active/active redundancy, aliasing, and mass MAC withdrawal. To provide a DCI solution, VXLAN is integrated with EVPN.

There are different options for using EVPN-VXLAN with DCI:

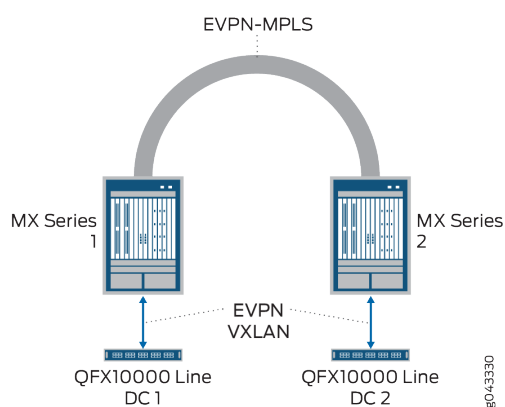
- DCI can connect multiple data centers in your WAN using MX Series edge routers with a Layer 3 VPN MPLS network between them. QFX10000 switches start and stop the VXLAN tunnel. This option requires no changes to your WAN.

Figure 50: DCI Option: Layer 3 VPN-MPLS



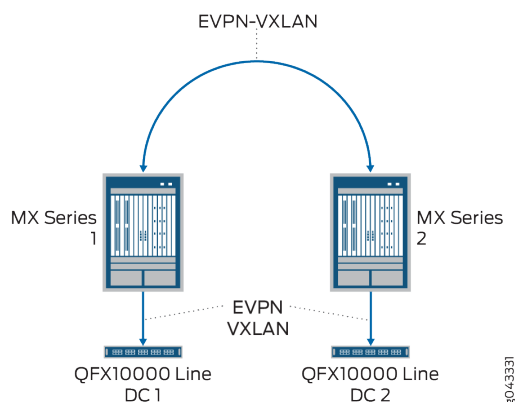
- A second option connects multiple data centers in your WAN using either MX Series edge routers or supported QFX Series switches with an EVPN-MPLS network between them. This option uses an EVPN control plane and an MPLS data plane and requires changes to your WAN. You must change your LAN architecture to natively support EVPN, and you must implement EVPN stitching between each MX router/QFX Series switch and the corresponding QFX10000 switch. For details about releases where QFX Series switches are supported, see <https://pathfinder.juniper.net/feature-explorer> and then search on EVPN.

Figure 51: DCI Option: EVPN-MPLS



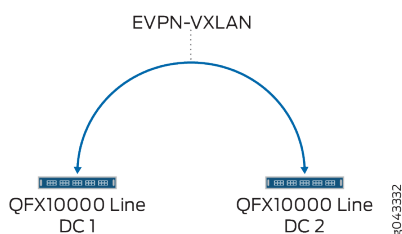
- You can also tunnel two branch locations across the Internet. In this case, implementation requires neither a traditional WAN nor MPLS. This method can use the Internet or an IP tunnel, where VXLAN rides on top of IP and EVPN is used throughout.

Figure 52: DCI Option: EVPN-VXLAN over the Internet



- If you do not have a branch router or a peering router, you can simply connect the data centers directly and EVPN is again used natively throughout. This implementation requires neither a traditional WAN nor MPLS, but you typically need a dark fiber connection.

Figure 53: DCI Option: Layer 3 VPN-MPLS Direct Connection



You can alternately create an EVPN-VXLAN fabric internally in the data center using bare-metal servers and/or virtual servers and using OpenClos for management. Here you also use VXLAN L2 gateways and L3 gateways on switches such as a QFX10000 switch. The underlying fabric is built on BGP.

EVPN-VXLAN uses both routers and switches—the configurations are the same for both devices but they are located in different areas of the Junos OS CLI. MX Series routers are configured under a routing instance with the instance type `virtual switch`. QFX Series switches are configured under `global switching-options` and `global protocol evpn`. See [Table 30 on page 633](#) for a list of CLI commands used by EVPN-VXLAN.

Table 30: CLI Commands for EVPN-VXLAN

Function	CLI Command
Specifies an identifier attached to a route. This enables you to distinguish to which VPN or VPLS the route belongs. Each routing instance must have a unique route distinguisher (RD) associated with it. The RD is used to place bounds around a VPN so that the same IP address prefixes can be used in different VPNs without having them overlap.	<i>route-distinguisher</i>
Specifies a VRF target community. In effect, this statement configures a single policy for import and a single policy for export to replace the per-VRF policies for every community. The options <code>import</code> and <code>export</code> apply to both routers and QFX Series switches. The option <code>auto</code> applies to QFX Series switches only.	<i>vrf-target</i>
Specifies how routes are imported into the VRF table of the local PE router or switch from the remote PE router.	<i>vrf-import</i>
Specifies how routes are exported from the local PE router's VRF table to the remote PE router.	<i>vrf-export</i>
A designated forwarder (DF) is required when CEs are multihomed to more than one PE. Without a designated forwarder, multihomed hosts would receive duplicate packets. Designated forwarders are chosen for an Ethernet segment identifier (ESI) based on type-4 route advertisements.	<i>designated-forwarder-election-hold-time</i>
Configures a logical link-layer encapsulation type.	<i>encapsulation</i>
Establishes which VXLAN virtual network identifiers (VNIs) will be part of the EVPN-VXLAN MP-BGP domain. There are different BUM replication options available in EVPN—using <code>extended-vni-list</code> forgoes a multicast underlay in favor of EVPN-VXLAN ingress replication.	<i>extended-vni-list</i>
You configure different route targets (RTs) for each VNI instance under <i>vni-options</i> .	<i>vni-options</i> (QFX Series switches only)
Displays both imported EVPN routes and export/import EVPN routes for the default switch routing instances.	<i>show route table</i>

Table 30: CLI Commands for EVPN-VXLAN (Continued)

Function	CLI Command
Displays results of the configuration commands <i>extended-vni-list</i> and <i>vni-options</i> .	show configuration protocols evpn

RELATED DOCUMENTATION

[Understanding EVPN with VXLAN Data Plane Encapsulation](#) | 588

PIM NSR and Unified ISSU Support for VXLAN Overview

Protocol Independent Multicast (PIM) nonstop active routing (NSR) support for Virtual Extensible LAN (VXLAN) is supported on MX Series routers.

The Layer 2 address learning daemon (l2ald) passes VXLAN parameters (VXLAN multicast group addresses and the source interface for a VXLAN tunnel [vtep-source-interface]) to the routing protocol process on the primary Routing Engine. The routing protocol process forms PIM joins with the multicast routes through the pseudo-VXLAN interface based on these configuration details.

Because the l2ald daemon does not run on the backup Routing Engine, the configured parameters are not available to the routing protocol process in the backup Routing Engine when NSR is enabled. The PIM NSR mirroring mechanism provides the VXLAN configuration details to the backup Routing Engine, which enables creation of the required states. The routing protocol process matches the multicast routes on the backup Routing Engine with PIM states, which maintains the multicast routes in the Forwarding state.

In response to Routing Engine switchover, the multicast routes remain in the Forwarding state on the new primary Routing Engine. This prevents traffic loss during Routing Engine switchover. When the l2ald process becomes active, it refreshes VXLAN configuration parameters to PIM.



NOTE: For this feature, NSR support is available for VXLAN in PIM sparse mode.


This feature does not introduce any new CLI commands. You can issue the following *show* commands on the backup Routing Engine to monitor the PIM joins and multicast routes on the backup Routing Engine:

- `show pim join extensive`

- `show multicast route extensive`

Unified ISSU Support

Starting in Junos OS Release 17.2 R1, unified in-service software upgrade is supported for VXLAN using PIM on MX Series routers. ISSU enables you to upgrade your Junos OS software on your MX Series router with no disruption on the control plane and with minimal disruption of traffic. Unified ISSU is supported only on dual Routing Engine platforms. The graceful Routing Engine switchover (GRES) and nonstop active routing (NSR) features must both be enabled. Unified ISSU allows you to eliminate network downtime, reduce operating costs, and deliver higher levels of services. See [Getting Started with Unified In-Service Software Upgrade](#).

 **NOTE:** Unified ISSU is not supported on the QFX series switches.

To enable GRES, include the `graceful-switchover` statement at the `[edit chassis redundancy]` hierarchy level.

To enable NSR, include the `nonstop-routing` statement at the `[edit routing-options]` hierarchy level and the `commit synchronize` statement at the `[edit system]` hierarchy level.

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 R1, unified in-service software upgrade is supported for VXLAN using PIM on MX Series routers.
16.2R1	Starting in Junos OS Release 16.2R1, Protocol Independent Multicast (PIM) nonstop active routing (NSR) support for Virtual Extensible LAN (VXLAN) is supported on MX Series routers.

RELATED DOCUMENTATION

`show pim join`

`show multicast route`

[Understanding Graceful Routing Engine Switchover](#)

Routing IPv6 Data Traffic through an EVPN-VXLAN Network with an IPv4 Underlay

IN THIS SECTION

- [Benefits of Routing IPv6 Data Traffic through an EVPN-VXLAN Network With an IPv4 Underlay | 637](#)
- [Centrally-Routed Bridging Overlay—How to Set Up the Routing of IPv6 Data Traffic | 637](#)
- [Edge-Routed Bridging Overlay—How to Set Up the Routing of IPv6 Data Traffic | 647](#)

On devices that can function as Layer 3 (L3) VXLAN gateways in an EVPN-VXLAN overlay network, IRB interfaces on L3 VXLAN gateways can route Layer 2 (L2) or L3 data packets from one IPv6 host to another IPv6 host through an EVPN-VXLAN network with an IPv4 underlay.

Upon receipt of an L2 or L3 data packet from an IPv6 host, an L3 VXLAN gateway encapsulates the packet with an IPv4 outer header, thereby tunneling the packet through the IPv4 underlay network. The L2 or L3 VXLAN gateway at the other end of the tunnel de-encapsulates the packet and forwards the packet towards the other IPv6 host.

The L3 VXLAN gateways in the EVPN-VXLAN overlay network learn the IPv6 routes through the exchange of EVPN Type 2 and Type 5 routes.

This feature is supported in EVPN-VXLAN overlay networks with the following architectures, which are commonly deployed within a data center:

- **EVPN-VXLAN centrally-routed bridging (CRB) overlay (also called an EVPN-VXLAN topology with a two-layer IPv4 fabric)**—A two-layer IPv4 fabric in which one layer of devices function as L3 VXLAN gateways, and another layer of devices (usually switches) function as L2 VXLAN gateways. In this architecture, the IRB interfaces configured on the L3 VXLAN gateways function in a central location in the fabric.
- **EVPN-VXLAN edge-routed bridging (ERB) overlay (also called an EVPN-VXLAN topology with a collapsed IPv4 fabric)**—A one-layer IPv4 fabric in which one layer of devices function as both L2 and L3 VXLAN gateways. In this architecture, the IRB interfaces configured on the L3 VXLAN gateways function at the edge of the fabric.

A key part of setting up this feature is configuring IRB interfaces on the L3 VXLAN gateways. In general, you configure the IRB interfaces as you would with only IPv4 hosts in the topology. However, in a CRB overlay fabric, in addition to specifying IPv4 addresses for the IRB interface and default L3 gateway (virtual gateway), you also specify IPv6 addresses. Similarly, in an ERB overlay fabric, in addition to specifying IPv4 addresses for the IRB interface, you also specify IPv6 addresses and you must configure a MAC address for the IRB interface.



NOTE: The tables in the next sections specify sample IPv6 IRB interface addresses using an IPv4-embedded IPv6 address format (for example, 2001:db8::192.168.100.1/96). The Junos CLI accepts address parameters you provide in this format to specify an IPv6 address associated with the IPv4 address for a dual-stack interface. However, in show command output that displays the IPv6 address, the output shows the address in the IPv6 address format with the colon-separated address fields as hexadecimal numbers.

Benefits of Routing IPv6 Data Traffic through an EVPN-VXLAN Network With an IPv4 Underlay

Routing IPv6 data traffic through an EVPN-VXLAN network with an IPv4 underlay provides the following benefits:

- Eliminates the need to deploy an IPv6 underlay network. The supported IPv4 fabric architectures are agile enough to support both IPv4 and IPv6 data traffic needs.
- Leverages the EVPN control plane's inherent support for exchanging IPv4 and IPv6 routes.
- Does not introduce any new configuration. To set up this feature, you configure IRB interfaces with IPv6 addresses.

Centrally-Routed Bridging Overlay—How to Set Up the Routing of IPv6 Data Traffic



NOTE: The focus of this section is the configuration of IRB interfaces on L3 VXLAN gateways. For a more comprehensive example of configuring EVPN-VXLAN, see ["Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric" on page 771](#).

[Table 31 on page 637](#) shows how to configure the addresses for IRB interfaces irb.100 and irb.200 on the three L3 VXLAN gateways. [Table 32 on page 644](#) outlines some additional required global configuration and configuration for irb.100 and irb.200.

Table 31: Centrally-Routed Bridging Overlay—Sample IRB Interface Addresses

Addresses	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3	Description
IRB interface irb.100				

Table 31: Centrally-Routed Bridging Overlay—Sample IRB Interface Addresses (Continued)

Addresses	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3	Description
IRB IPv4 address	192.168.100.1/24	192.168.100.2/24	192.168.100.3/24	Specify a different IPv4 address for irb.100 on each device.
IRB global IPv6 address	2001:db8::192.168.100.1/96	2001:db8::192.168.100.2/96	2001:db8::192.168.100.3/96	Specify a different IPv6 address for irb.100 on each device.
IRB link-local IPv6 address	fe80::100:00:01/64	fe80::100:00:01/64	fe80::100:00:01/64	Specify the same link-local IPv6 address for irb.100 on each device. Any packet destined to this IPv6 address is intercepted for Network Discovery Protocol (NDP) processing.
IRB link-local IPv6 address (DHCPv6 or SLAAC)	fe80::100:00:11/64	fe80::100:00:12/64	fe80::100:00:13/64	Specify a different link-local address on each device when using DHCPv6 or SLAAC in the EVPN Fabric. You must also configure the router advertisement for this IRB.
Virtual gateway IPv4 address	192.168.100.254/24	192.168.100.254/24	192.168.100.254/24	Specify the same IPv4 anycast address for the virtual gateway on each device.

Table 31: Centrally-Routed Bridging Overlay—Sample IRB Interface Addresses *(Continued)*

Addresses	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3	Description
Virtual gateway IPv6 address	2001:db8::192.168. 100.254/96	2001:db8::192.168. 100.254/96	2001:db8::192.168. 100.254/96	Specify the same IPv6 anycast address for the virtual gateway on each device.
Virtual gateway IPv6 address (DHCPv6 or SLAAC)	fe80::100:00:100	fe80::100:00:100	fe80::100:00:100	Specify the same virtual-gateway- address for the link- local address on all the devices when using DHCPV6 or SLAAC in the EVPN Fabric. You must also configure the router advertisement for this IRB.

Table 31: Centrally-Routed Bridging Overlay—Sample IRB Interface Addresses (*Continued*)

Addresses	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3	Description
Virtual gateway IPv4 and IPv6 MAC addresses	Method 1	Method 1	Method 1	<p>For the default L3 gateway, the IPv4 and IPv6 MAC addresses can be the same or different, as long as they are consistent across the three devices. Here are the supported options:</p> <ul style="list-style-type: none"> Method 1—you explicitly configure the same IPv4 and IPv6 MAC address on each device. Method 2—you explicitly configure different IPv4 and IPv6 MAC addresses on each device. Method 3—you do not explicitly configure IPv4 and IPv6 MAC addresses, and the system automatically generates 00:00:5e:00:01:01 as the IPv4 MAC address
	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:fe 	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:fe 	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:fe 	
	Method 2	Method 2	Method 2	
	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:ff 	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:ff 	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:ff 	
	Method 3	Method 3	Method 3	
	<ul style="list-style-type: none"> IPv4 MAC address: 00:00:5e:00:01:01 IPv6 MAC address: 00:00:5e:00:02:01 	<ul style="list-style-type: none"> IPv4 MAC address: 00:00:5e:00:01:01 IPv6 MAC address: 00:00:5e:00:02:01 	<ul style="list-style-type: none"> IPv4 MAC address: 00:00:5e:00:01:01 IPv6 MAC address: 00:00:5e:00:02:01 	

Table 31: Centrally-Routed Bridging Overlay—Sample IRB Interface Addresses (Continued)

Addresses	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3	Description
				and 00:00:5e:00:02: 01 as the IPv6 MAC address.

IRB interface irb.200

IRB IPv4 address	192.168.200.1/24	192.168.200.2/24	192.168.200.3/24	Specify a different IPv4 address for irb.200 on each device.
IRB global IPv6 address	2001:db8::192.168.200.1/96	2001:db8::192.168.200.2/96	2001:db8::192.168.200.3/96	Specify a different IPv6 address for irb.200 on each device.
IRB link-local IPv6 address	fe80::200:00:01/64	fe80::200:00:01/64	fe80::200:00:01/64	Specify the same link-local IPv6 address for irb.200 on each device. Any packet destined to this IPv6 address is intercepted for NDP processing.
IRB link-local IPv6 address (DHCPv6 or SLAAC)	fe80::200:00:11/64	fe80::200:00:12/64	fe80::200:00:13/64	Specify a different link-local address on each device when using DHCPV6 or SLAAC in the EVPN Fabric. You must also configure the router advertisement for this IRB.

Table 31: Centrally-Routed Bridging Overlay—Sample IRB Interface Addresses *(Continued)*

Addresses	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3	Description
Virtual gateway IPv4 address	192.168.200.254/24	192.168.200.254/24	192.168.200.254/24	Specify the same IPv4 anycast address for the virtual gateway on each device.
Virtual gateway IPv6 address	2001:db8::192.168.200.254/96	2001:db8::192.168.200.254/96	2001:db8::192.168.200.254/96	Specify the same IPv6 anycast address for the virtual gateway on each device.
Virtual gateway IPv6 address (DHCPv6 or SLAAC)	fe80::100:00:200	fe80::100:00:200	fe80::100:00:200	Specify the same virtual-gateway-address for the link-local address on all the devices when using DHCPV6 or SLAAC in the EVPN Fabric. You must also configure the router advertisement for this IRB.

Table 31: Centrally-Routed Bridging Overlay—Sample IRB Interface Addresses (*Continued*)

Addresses	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3	Description
Virtual gateway IPv4 and IPv6 MAC addresses	Method 1	Method 1	Method 1	<p>For the default L3 gateway, the IPv4 and IPv6 MAC addresses can be the same or different, as long as they are consistent across the three devices. Here are the supported options:</p> <ul style="list-style-type: none"> Method 1—you explicitly configure the same IPv4 and IPv6 MAC address on each device. Method 2—you explicitly configure different IPv4 and IPv6 MAC addresses on each device. Method 3—you do not explicitly configure IPv4 and IPv6 MAC addresses, and the system automatically generates 00:00:5e:00:01:01 as the IPv4 MAC address
	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:fe 	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:fe 	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:fe 	
	Method 2	Method 2	Method 2	
	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:ff 	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:ff 	<ul style="list-style-type: none"> IPv4 MAC address: 10:00:00:00:00:fe IPv6 MAC address: 10:00:00:00:00:ff 	
	Method 3	Method 3	Method 3	
	<ul style="list-style-type: none"> IPv4 MAC address: 00:00:5e:00:01:01 IPv6 MAC address: 00:00:5e:00:02:01 	<ul style="list-style-type: none"> IPv4 MAC address: 00:00:5e:00:01:01 IPv6 MAC address: 00:00:5e:00:02:01 	<ul style="list-style-type: none"> IPv4 MAC address: 00:00:5e:00:01:01 IPv6 MAC address: 00:00:5e:00:02:01 	

Table 31: Centrally-Routed Bridging Overlay—Sample IRB Interface Addresses (Continued)

Addresses	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3	Description
				and 00:00:5e:00:02: 01 as the IPv6 MAC address.

Table 32: Centrally-Routed Bridging Overlay—Required IRB Interface Configuration

Description	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3
-------------	--------------------	--------------------	--------------------

Global IRB interface configuration

Specify that the IPv4 and IPv6 MAC addresses of the default L3 gateway are advertised to the L2 VXLAN gateways without the extended community option.	set protocols evpn default-gateway no-gateway-community	set protocols evpn default-gateway no-gateway-community	set protocols evpn default-gateway no-gateway-community
---	---	---	---

IRB interface irb.100 configuration

Configure the L3 VXLAN gateway to advertise the MAC and IP routes (MAC+IP type 2 routes) on behalf of the L2 VXLAN gateways.	set interfaces irb unit 100 proxy-macip-advertisement	set interfaces irb unit 100 proxy-macip-advertisement	set interfaces irb unit 100 proxy-macip-advertisement
--	---	---	---

Table 32: Centrally-Routed Bridging Overlay—Required IRB Interface Configuration (Continued)

Description	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3
Enable the default L3 gateway to be pinged by either IPv4 or IPv6 addresses.	<pre>set interfaces irb unit 100 virtual-gateway-accept-data set interfaces irb unit 100 family inet address 192.168.100.1/24 preferred set interfaces irb unit 100 family inet6 address 2001:db8::192.168.100.1/96 preferred</pre>	<pre>set interfaces irb unit 100 virtual-gateway-accept-data set interfaces irb unit 100 family inet address 192.168.100.2/24 preferred set interfaces irb unit 100 family inet6 address 2001:db8::192.168.100.2/96 preferred</pre>	<pre>set interfaces irb unit 100 virtual-gateway-accept-data set interfaces irb unit 100 family inet address 192.168.100.3/24 preferred set interfaces irb unit 100 family inet6 address 2001:db8::192.168.100.3/96 preferred</pre>

IRB interface irb.200 configuration

Configure the L3 VXLAN gateway to advertise the MAC and IP routes (MAC+IP type 2 routes) on behalf of the L2 VXLAN gateways.	<pre>set interfaces irb unit 200 proxy-macip-advertisement</pre>	<pre>set interfaces irb unit 200 proxy-macip-advertisement</pre>	<pre>set interfaces irb unit 200 proxy-macip-advertisement</pre>
Enable the default L3 gateway to be pinged by either IPv4 or IPv6 addresses.	<pre>set interfaces irb unit 200 virtual-gateway-accept-data set interfaces irb unit 200 family inet address 192.168.200.1/24 preferred set interfaces irb unit 200 family inet6 address 2001:db8::192.168.200.1/96 preferred</pre>	<pre>set interfaces irb unit 200 virtual-gateway-accept-data set interfaces irb unit 200 family inet address 192.168.200.2/24 preferred set interfaces irb unit 200 family inet6 address 2001:db8::192.168.200.2/96 preferred</pre>	<pre>set interfaces irb unit 200 virtual-gateway-accept-data set interfaces irb unit 200 family inet address 192.168.200.3/24 preferred set interfaces irb unit 200 family inet6 address 2001:db8::192.168.200.3/96 preferred</pre>

DHCPv6 or SLAAC configuration

Table 32: Centrally-Routed Bridging Overlay—Required IRB Interface Configuration (Continued)

Description	L3 VXLAN Gateway 1	L3 VXLAN Gateway 2	L3 VXLAN Gateway 3
Configure the link-local address.	<pre>set interfaces irb.100 family inet6 address fe80::100:00:11/64 virtual- gateway-address fe80::100:00:100 set interfaces irb.200 family inet6 address fe80::200:00:11/64 fe80::200:00:100</pre>	<pre>set interfaces irb.100 family inet6 address fe80::100:00:12/64 virtual- gateway-address fe80::100:00:100 set interfaces irb.200 family inet6 address fe80::200:00:12/64 fe80::200:00:100</pre>	<pre>set interfaces irb.100 family inet6 address fe80::100:00:13/64 virtual- gateway-address fe80::100:00:100 set interfaces irb.200 family inet6 address fe80::200:00:13/64 fe80::200:00:100</pre>
Configure the global address.	<pre>set interfaces irb.100 family inet6 address 2001:db8::192.168.100.1/96 virtual-gateway-address 2001:db8::192.168.100.254 set interfaces irb.200 family inet6 address 2001:db8::192.168.200.1/96 virtual-gateway-address 2001:db8::192.168.200.254</pre>	<pre>set interfaces irb.100 family inet6 address 2001:db8::192.168.100.2/96 virtual-gateway-address 2001:db8::192.168.100.254 set interfaces irb.200 family inet6 address 2001:db8::192.168.200.2/96 virtual-gateway-address 2001:db8::192.168.200.254</pre>	<pre>set interfaces irb.100 family inet6 address 2001:db8::192.168.100.3/96 virtual-gateway-address 2001:db8::192.168.100.254 set interfaces irb.200 family inet6 address 2001:db8::192.168.200.3/96 virtual-gateway-address 2001:db8::192.168.200.254</pre>
Configure the gateway to send router advertisement packets only for the link-local virtual-gateway-address.	<pre>set protocols router- advertisement interface irb.100 virtual-router-only set protocols router- advertisement interface irb.200 virtual-router-only</pre>	<pre>set protocols router- advertisement interface irb.100 virtual-router-only set protocols router- advertisement interface irb.200 virtual-router-only</pre>	<pre>set protocols router- advertisement interface irb.100 virtual-router-only set protocols router- advertisement interface irb.200 virtual-router-only</pre>

Edge-Routed Bridging Overlay—How to Set Up the Routing of IPv6 Data Traffic



NOTE: The focus of this section is the configuration of IRB interfaces on L3 VXLAN gateways. For a more comprehensive example of configuring EVPN-VXLAN, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway"](#) on page 845.

[Table 33 on page 647](#) shows how to configure the addresses for IRB interfaces irb.100 and irb.200 on the three L3 VXLAN gateways. [Table 34 on page 649](#) outlines some additional required global IRB interface configuration.

Table 33: Edge-Routed Bridging Overlay—Sample IRB Interface Addresses

Addresses	L2 and 3 VXLAN Gateway 1	L2 and 3 VXLAN Gateway 2	L2 and 3 VXLAN Gateway 3	Description
IRB interface irb.100				
IPv4 address	192.168.100.1/24	192.168.100.1/24	192.168.100.1/24	Specify the same IPv4 address for irb.100 on each device.
Global IPv6 address	2001:db8::192.168.100.1/96	2001:db8::192.168.100.1/96	2001:db8::192.168.100.1/96	Specify the same IPv6 address for irb.100 on each device.
Link-local IPv6 address	fe80::100:00:01/64	fe80::100:00:01/64	fe80::100:00:01/64	Specify the same link-local IPv6 address for irb.100 on each device. Any packet destined to this IPv6 address is intercepted for NDP processing.

Table 33: Edge-Routed Bridging Overlay—Sample IRB Interface Addresses (Continued)

Addresses	L2 and 3 VXLAN Gateway 1	L2 and 3 VXLAN Gateway 2	L2 and 3 VXLAN Gateway 3	Description
IRB link-local IPv6 address (DHCPv6 or SLAAC)	fe80::100:00:11/64	fe80::100:00:12/64	fe80::100:00:13/64	Specify a different link-local address on each device when using DHCPv6 or SLAAC in the EVPN Fabric. You must also configure the router advertisement for this IRB.
IRB MAC address	10:00:00:00:00:fe	10:00:00:00:00:fe	10:00:00:00:00:fe	Specify the same MAC address for irb.100 on each device.

IRB interface irb.200

IPv4 address	192.168.200.1/24	192.168.200.1/24	192.168.200.1/24	Specify the same IPv4 address for irb.200 on each device.
Global IPv6 address	2001:db8::192.168. 200.1/96	2001:db8::192.168. 200.1/96	2001:db8::192.168. 200.1/96	Specify the same IPv6 address for irb.200 on each device.
Link-local IPv6 address	fe80::200:00:01/64	fe80::200:00:01/64	fe80::200:00:01/64	Specify the same link-local IPv6 address for irb.200 on each device. Any packet destined to this IPv6 address is intercepted for NDP processing.

Table 33: Edge-Routed Bridging Overlay—Sample IRB Interface Addresses (Continued)

Addresses	L2 and 3 VXLAN Gateway 1	L2 and 3 VXLAN Gateway 2	L2 and 3 VXLAN Gateway 3	Description
IRB link-local IPv6 address (DHCPv6 or SLAAC)	fe80::200:00:11/64	fe80::200:00:12/64	fe80::200:00:13/64	Specify a different link-local address on each device when using DHCPV6 or SLAAC in the EVPN Fabric. You must also configure the router advertisement for this IRB..
IRB MAC address	10:00:00:00:00:ff	10:00:00:00:00:ff	10:00:00:00:00:ff	Specify the same MAC address for irb.200 on each device.

Table 34: Edge-Routed Bridging Overlay—Required IRB Interface Configuration

Description	L2 and 3 VXLAN Gateway 1	L2 and 3 VXLAN Gateway 2	L2 and 3 VXLAN Gateway 3
-------------	--------------------------	--------------------------	--------------------------

Global IRB interface configuration

For IPv4 fabric 2, a default L3 gateway is not configured. Therefore, you must disable the advertisement of a default L3 gateway.	set protocols evpn default-gateway do-not-advertise	set protocols evpn default-gateway do-not-advertise	set protocols evpn default-gateway do-not-advertise
---	---	---	---

DHCPv6 or SLAAC configuration

Table 34: Edge-Routed Bridging Overlay—Required IRB Interface Configuration (Continued)

Description	L2 and 3 VXLAN Gateway 1	L2 and 3 VXLAN Gateway 2	L2 and 3 VXLAN Gateway 3
Configure the link-local address.	set interfaces irb.100 family inet6 address fe80::100:00:11/64 virtual- gateway-address fe80::100:00:254 set interfaces irb.200 family inet6 address fe80::200:00:11/64 virtual- gateway-address fe80::200:00:254	set interfaces irb.100 family inet6 address fe80::100:00:12/64 virtual- gateway-address fe80::100:00:254 set interfaces irb.200 family inet6 address fe80::200:00:12/64 virtual- gateway-address fe80::200:00:254	set interfaces irb.100 family inet6 address fe80::100:00:13/64 virtual- gateway-address fe80::100:00:254 set interfaces irb.200 family inet6 address fe80::200:00:13/64 virtual- gateway-address fe80::200:00:254
Configure the global IPv6 address.	set interfaces irb.100 family inet6 address 2001:db8::192.168.100.1/96 set interfaces irb.200 family inet6 address 2001:db8::192.168.200.1/9	set interfaces irb.100 family inet6 address 2001:db8::192.168.100.1/96 set interfaces irb.200 family inet6 address 2001:db8::192.168.200.1/9	set interfaces irb.100 family inet6 address 2001:db8::192.168.100.1/96 set interfaces irb.200 family inet6 address 2001:db8::192.168.200.1/9
Configure the gateway to send router advertisement packets only for the link-local virtual-gateway-address.	set protocols router- advertisement interface irb.100 virtual-router-only set protocols router- advertisement interface irb.200 virtual-router-only	set protocols router- advertisement interface irb.100 virtual-router-only set protocols router- advertisement interface irb.200 virtual-router-only	set protocols router- advertisement interface irb.100 virtual-router-only set protocols router- advertisement interface irb.200 virtual-router-only

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
15.1X53-D30	Starting with Junos OS Release 15.1X53-D30 for QFX10000 switches and Junos OS Release 18.4R1 for QFX5110 switches, the IRB interfaces on these Layer 3 VXLAN gateways can route Layer 2 or Layer 3 data packets from one IPv6 host to another IPv6 host through an EVPN-VXLAN network with an IPv4 underlay.

RELATED DOCUMENTATION

[Using a Default Layer 3 Gateway to Route Traffic in an EVPN-VXLAN Overlay Network | 741](#)

Understanding How to Configure VXLANs and Layer 3 Logical Interfaces to Interoperate

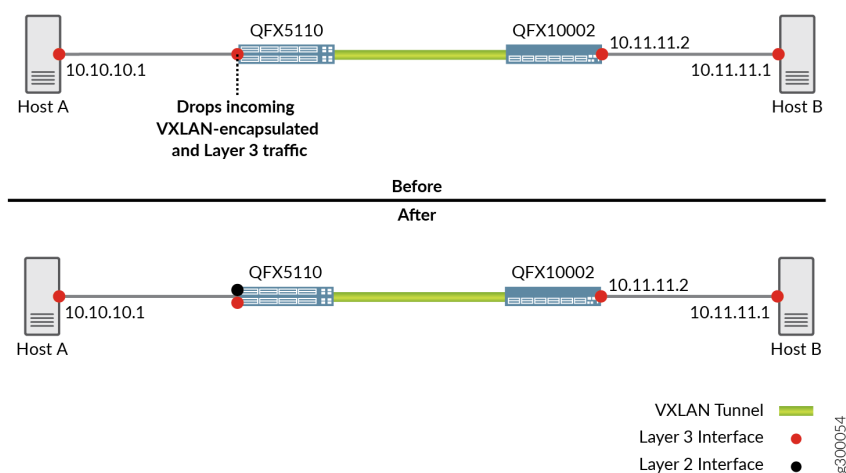


NOTE: The use case and additional configuration described in this topic apply only to QFX5110 switches.

In the sample EVPN-VXLAN network segments shown in [Figure 54 on page 652](#) (Before), hosts A and B need to exchange traffic. When host A sends a packet to host B or vice versa, the packet must traverse the following networking entities:

- On QFX5110 switch: a pure Layer 3 logical interface configured using the set `interfaces interface-name unit logical-unit-number family inet address ip-address/prefix-length` or the set `interfaces interface-name unit logical-unit-number family inet6 address ipv6-address/prefix-length` command.
- A VXLAN tunnel between the QFX5110 switch and the QFX10002 switch.
- On QFX10002 switch: a pure Layer 3 logical interface configured as described in the first bullet.

Figure 54: Results When Routing Traffic Between a VXLAN and a Layer 3 Logical Interface Is Disabled (Before) and Enabled (After)



By default, routing traffic between a VXLAN and a Layer 3 logical interface is disabled. When this functionality is disabled, the pure Layer 3 logical interface on the QFX5110 switch drops Layer 3 traffic from host A and VXLAN-encapsulated traffic from the QFX10002 switch. To prevent the pure Layer 3 logical interface on the QFX5110 switch from dropping this traffic, you can perform some additional configuration on the switch.

The additional configuration on the QFX5110 switch entails the following on a physical interface ([Figure 54 on page 652](#) (After)):

- Reconfiguring the pure Layer 3 logical interface as a Layer 2 logical interface and associating this interface with a dummy VLAN and a dummy VXLAN network identifier (VNI).
- Creating an IRB interface, which provides Layer 3 functionality within the dummy VLAN.

For example:

```
set interfaces irb unit 3500 family inet address 10.10.10.2/30
```

```
set interfaces xe-0/0/13 unit 0 family ethernet-switching interface-mode trunk
```

```
set interfaces xe-0/0/13 unit 0 family ethernet-switching vlan members VLAN-3500
```

```
set vlans VLAN-3500 vlan-id 3500
```

```
set vlans VLAN-3500 l3-interface irb.3500
```

```
set vlans VLAN-3500 vxlan vni 16770000
```

In lieu of the original pure Layer 3 logical interface, the newly created Layer 2-Layer 3 logical interfaces can now handle Layer 3 traffic from host A and VXLAN-encapsulated traffic from the QFX10002 switch.

Configuring EVPN-VXLAN Interfaces

IN THIS CHAPTER

- [Understanding Flexible Ethernet Services Support With EVPN-VXLAN | 654](#)
- [EVPN-VXLAN Lightweight Leaf to Server Loop Detection | 657](#)
- [Overlapping VLAN Support Using VLAN Translation in EVPN-VXLAN Networks | 668](#)
- [Overlapping VLAN Support Using Multiple Forwarding Instances or VLAN Normalization | 674](#)
- [Layer 2 Protocol Tunneling over VXLAN Tunnels in EVPN-VXLAN Bridged Overlay Networks | 679](#)
- [MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment | 687](#)
- [GRE over EVPN-VXLAN | 705](#)
- [DHCP Smart Relay in EVPN-VXLAN | 708](#)

Understanding Flexible Ethernet Services Support With EVPN-VXLAN

IN THIS SECTION

- [Sample Configuration 1—Layer 2 VXLANs and Layer 3 IPv4 Logical Interfaces on Same Physical Interface | 655](#)
- [Sample Configuration 2—Layer 2 VLAN and VXLAN Logical Interfaces on Same Physical Interface | 656](#)

Flexible Ethernet services are an encapsulation type that enables a physical interface to support different types of Ethernet encapsulations at the logical interface level. Juniper Networks supports flexible Ethernet services with EVPN VXLAN.

The benefits of flexible Ethernet services support with EVPN-VXLAN include the following:

- A physical interface now supports `family ethernet-switching` and Virtual Extensible LANs (VXLANs) configured on one or more logical interfaces and `family inet` configured on other logical interfaces.

- A physical interface now supports Layer 2 VLANs configured on one or more logical interfaces and VXLANs configured on other logical interfaces.
- Instead of configuring the following Layer 2 features only on logical interface unit number 0, you can now configure them on any logical interface unit number (unit 0 and any non-zero unit number):
 - Layer 2 bridging (`family ethernet-switching`)
 - Layer 2 bridging (`encapsulation vlan-bridge`)
 - VXLANs
- On a physical interface, you can now configure Layer 2 bridging with `family ethernet-switching` on one or more logical interfaces and Layer 2 bridging with `encapsulation vlan-bridge` on one or more logical interfaces.
- On a physical interface, you can now configure one or more logical interfaces with `encapsulation extended-vlan-bridge` and one or more logical interfaces with `interface-mode trunk`.



NOTE: We don't support configuring both VXLAN VLANs and non-VXLAN VLANs on the same logical interface. Instead, configure separate logical interfaces with each handling either VXLAN or non-VXLAN VLAN traffic.

This topic provides the following information about configuring this feature:

Sample Configuration 1—Layer 2 VXLANs and Layer 3 IPv4 Logical Interfaces on Same Physical Interface

This sample configuration shows how to configure a logical interface for Layer 2 VXLAN forwarding and a logical interface for Layer 3 IPv4 routing on the same physical interface.

When configuring the logical interfaces, keep the following in mind:

- For this configuration to be successfully committed and work properly, you must specify the `encapsulation flexible-ethernet-services` configuration statements at the physical interface level—for example, set `interfaces xe-0/0/5 encapsulation flexible-ethernet-services`.
- You must configure Layer 2 VXLAN and Layer 3 routing on separate logical interfaces.

The following sample configuration shows physical interface `et-0/0/16`, on which the flexible Ethernet services encapsulation type is enabled. This physical interface is divided into logical interface 100, which is a Layer 2 (`family ethernet-switching`) interface that is associated with VXLANs `v100` and `v500`, and logical interface 600, which is a Layer 3 (`family inet`) interface. Note that specifying the flexible Ethernet

services encapsulation type on physical interface et-0/0/16 also enables you to configure the Layer 2 logical interface on a non-zero unit number, in this case, 100.

```
set interfaces et-0/0/16 flexible-vlan-tagging
set interfaces et-0/0/16 encapsulation flexible-ethernet-services
set interfaces et-0/0/16 unit 100 family ethernet-switching interface-mode trunk
set interfaces et-0/0/16 unit 100 family ethernet-switching vlan members v100
set interfaces et-0/0/16 unit 100 family ethernet-switching vlan members v500
set interfaces et-0/0/16 unit 600 family inet address 10.1.1.2/24
set interfaces et-0/0/16 unit 600 vlan-id 600
set vlans v100 vlan-id 100
set vlans v100 vxlan vni 100
set vlans v500 vlan-id 500
set vlans v500 vxlan vni 500
set vlans v600 vlan-id 600
```



NOTE: This example focuses on the configuration of the physical and logical interfaces and the VXLANs with which logical interface 100 is associated. This sample configuration does not include a comprehensive configuration of EVPN and VXLAN. For a more comprehensive EVPN-VXLAN configuration, see ["Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric" on page 771](#).

Sample Configuration 2—Layer 2 VLAN and VXLAN Logical Interfaces on Same Physical Interface

This sample configuration shows how to configure a logical interface for Layer 2 VXLAN forwarding and logical interfaces for Layer 2 VLAN forwarding on the same physical interface.

When configuring the logical interfaces for Layer 2 VLAN and Layer 2 VXLAN forwarding, you must specify the encapsulation flexible-ethernet-services configuration statements at the physical interface level—for example, set interfaces xe-0/0/5 encapsulation flexible-ethernet-services.



NOTE: Starting in Junos OS Release 20.3R1 on QFX5110 and QFX5120 switches, we support Layer 2 VLAN and VXLAN logical interfaces on the same physical interface using service provider style interface configurations only.

The following sample configuration shows physical interface xe-0/0/4, on which the flexible Ethernet services encapsulation type is enabled. This physical interface is divided into the following logical interfaces:

- Logical interface 100, which is a Layer 2 (family ethernet-switching) interface that is associated with VXLAN v100.
- Logical interface 200, which is a Layer 2 (encapsulation vlan-bridge) interface that is associated with VLAN v200
- Logical interface 300, which is a Layer 2 (encapsulation vlan-bridge) interface that is associated with VLAN v300

```
set interfaces xe-0/0/4 flexible-vlan-tagging
set interfaces xe-0/0/4 encapsulation flexible-ethernet-services
set interfaces xe-0/0/4 unit 100 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/4 unit 100 family ethernet-switching vlan members v100
set interfaces xe-0/0/4 unit 200 encapsulation vlan-bridge vlan members v200
set interfaces xe-0/0/4 unit 300 encapsulation vlan-bridge vlan members v300
set vlans v100 vlan-id 100
set vlans v100 vxlan vni 100
set vlans v200 vlan-id 200
set vlans v300 vlan-id 300
```

Note that specifying the flexible Ethernet services encapsulation type on physical interface et-0/0/4 enables you to configure all Layer 2 logical interfaces on non-zero unit numbers, in this case, 100, 200, and 300. Also, you can now configure Layer 2 bridging with family ethernet-switching on one or more logical interfaces—in this case, on logical interface 100—and Layer 2 bridging with encapsulation vlan-bridge on one or more logical interfaces—in this case, logical interfaces 200 and 300.

EVPN-VXLAN Lightweight Leaf to Server Loop Detection

IN THIS SECTION

- [How Lightweight PE-CE Loop Detect Works | 658](#)
- [Options for Repairing Loops | 658](#)
- [Supported Interface Configurations | 659](#)
- [Lightweight PE-CE Loop Detect Scenarios | 659](#)
- [Lightweight PE-CE Loop Detect Use Cases using Layer 2 Heartbeats | 662](#)
- [Enable Lightweight PE-CE Loop Detect on a Logical Interface | 664](#)
- [Sample Configuration with Trunk Mode Enterprise Style Interface | 665](#)

- [Sample Configuration with Service Provider Style Interface | 666](#)
- [CLI Commands to Display or Clear Loop Detect Status | 666](#)

Configure EVPN-VXLAN lightweight provider edge (PE) to customer edge (CE) loop detection to quickly detect and break local area network (LAN) Ethernet loops downstream on the leaf-to-server port side. We call this feature lightweight leaf to server loop detect, lightweight PE-CE loop detect, or enhanced loop detect. This feature detects and breaks loops for:

- Inaccurate wiring of the fabric components.
- Inaccurate wiring or misconfiguration of third party switches to EVPN fabric devices (such as when connecting customer edge (CE) switches).

This feature helps you find and repair loops that the EVPN control plane can't detect without having to rely on the state of BGP EVPN signaling.

How Lightweight PE-CE Loop Detect Works

When you configure lightweight PE-CE loop detect, the device transmits periodic multicast protocol data units (PDUs) on PE to CE interfaces for detecting loops. The device can then block the interface upon receiving these self-generated PDUs. When the device receives a loop detect PDU, it breaks the loop by blocking (operationally shutting down) the ingress port.

The loop detect PDUs use the Connectivity Fault Management (CFM) protocol PDU format, although you don't explicitly configure CFM with this feature. The loop detect error messages logged by this feature include the CFM keyword, such as CFMD_LOOP_DETECTED and CFMD_LOOP_CLEARED.

We recommend that you enable lightweight PE-CE loop detect initially *before* you configure EVPN-VXLAN, so you can detect any loops and take corrective actions before EVPN traffic is flowing through the network. When this feature detects loops, the device raises loop detect error messages immediately. However, if you bring up a large-scale EVPN network that already contains loops, even with this feature enabled, the interface doesn't come down immediately and traffic continues to flow through the loop for some time while the network stabilizes.

If a loop is introduced into the network later in a stable running EVPN-VXLAN fabric, this feature will detect the loop and stop traffic flow through the loop immediately.

Options for Repairing Loops

If required, break and clear the loop. To bring the interface back online, you can configure a revert interval using the `revert-interval seconds` statement at the `[edit protocols loop-detect interface name]`

hierarchy level. When the revert interval expires, the device automatically brings the interface back online. The default revert interval is 0 seconds, which means the interval never expires and the interface doesn't automatically revert to its prior state.

If you don't explicitly configure a revert interval other than 0, the port never reverts to its state before the loop detect event and action. To manually bring the interface back online, you must clear the status using the `clear loop-detect enhanced interface name` command.

Supported Interface Configurations

To configure this lightweight PE-CE loop detect feature, specify a logical interface name. We don't support this feature with physical interfaces, only with logical interfaces as follows:

- **Enterprise style interface configurations without flexible Ethernet services**—Only with logical unit 0.
- **Enterprise style interface configurations with flexible Ethernet services**—On any logical interfaces you can configure on the device, including logical interfaces for a trunk interface with the native VLAN ID and other configured VLANs.
- **Service provider style interface configurations**—Only on QFX10002-60C, QFX10002, QFX10008, and QFX10016 switches, starting in Junos OS Release 22.4R1. We don't support this lightweight loop detect feature on service provider style interfaces with any other devices.
- **Aggregated Ethernet interfaces**—On a logical unit *X* of an aggregated Ethernet interface (*aeN.X*). With enterprise style aggregated Ethernet interface configurations without flexible Ethernet services configured, we only support logical unit 0. Otherwise you can use any configured logical unit *X*.

On aggregated Ethernet interfaces with Link Aggregation Control Protocol (LACP) enabled, the LACP state remains up (Collecting or Distributing) even if the loop detect action brings the logical interface down.

See *Flexible Ethernet Services Encapsulation* for more on flexible Ethernet services, enterprise style interface configurations, and service provider style interface configurations.

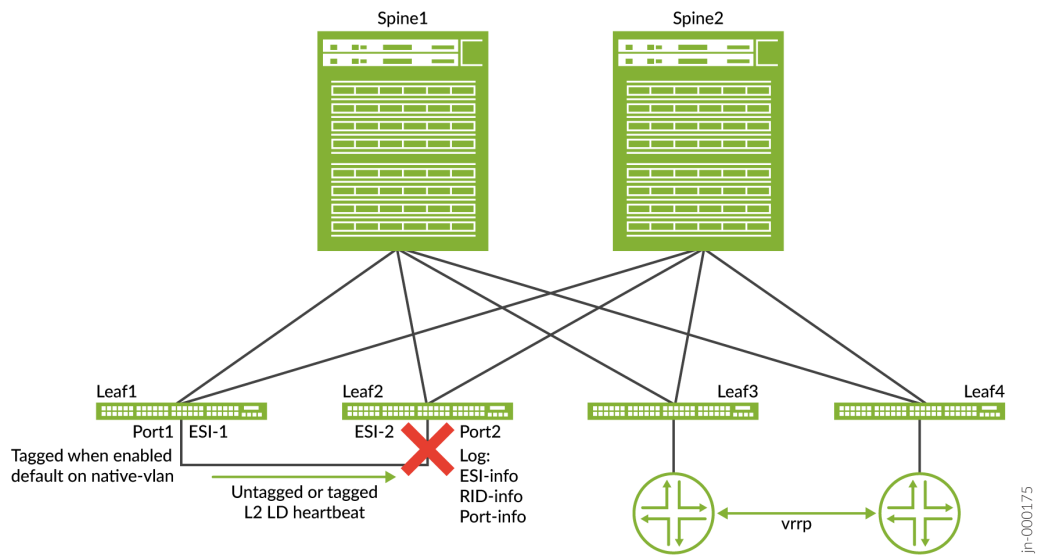
Lightweight PE-CE Loop Detect Scenarios

The following three lightweight PE-CE loop detect scenarios demonstrate that loops can form with different Ethernet segment identifiers (ESIs), with the same ESI, or with no ESI.

Different ESI Looped

When the loop occurs with different ESIs, you can enable a range of fabric router IDs on which the device triggers the loop detect feature (mandatory). Or, you can build the list automatically using router IDs based on EVPN Type 1 auto-discovery route signaling (optional).

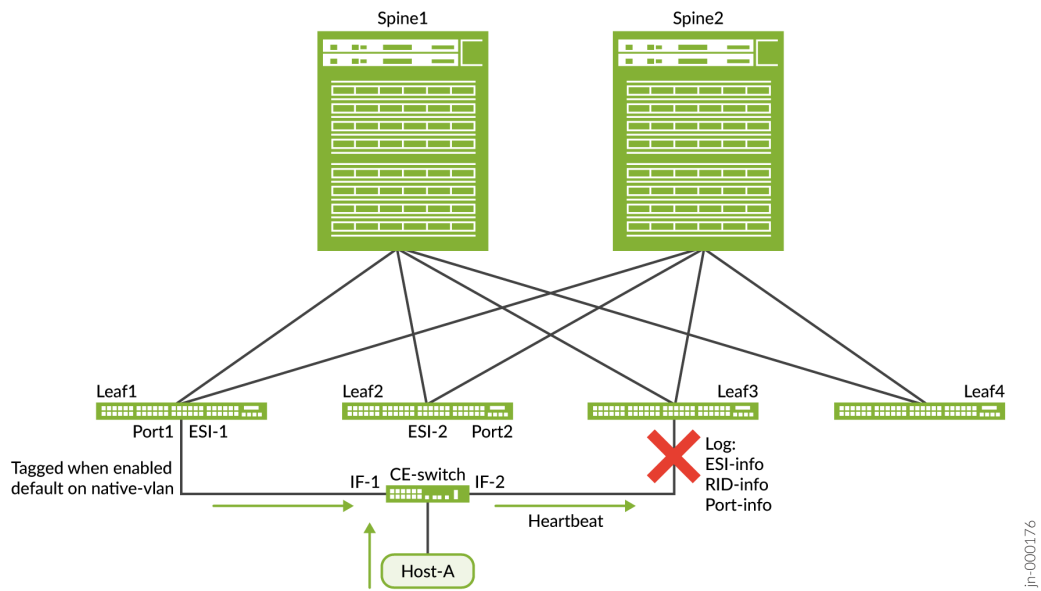
Figure 55: Different ESI Looped



Same ESI Looped

When the loop occurs with the same ESI, the CE switch is not using the same bridged interface when connecting to Leaf1 and Leaf3.

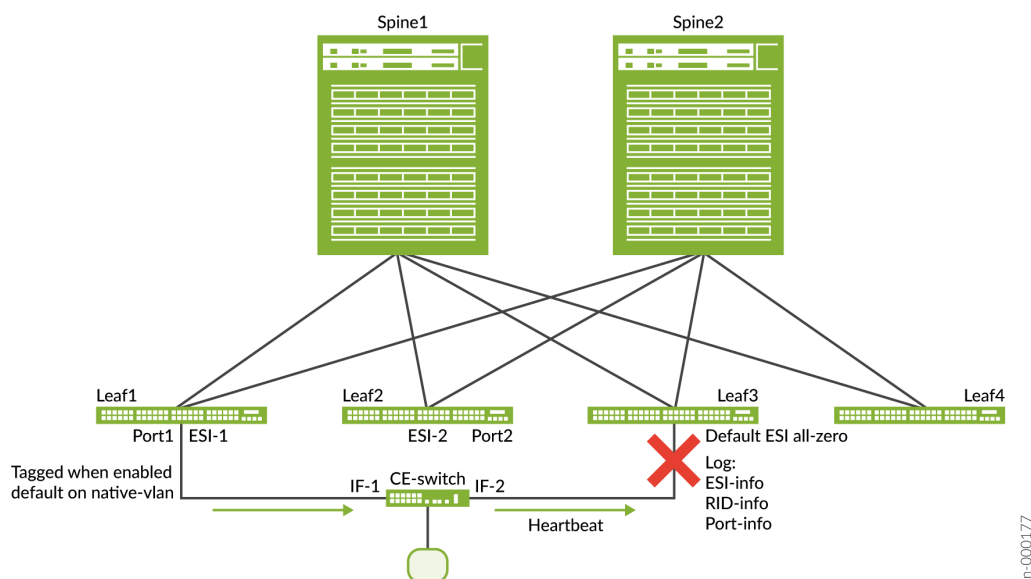
Figure 56: Same ESI Looped



No ESI on Looped Ports

When one of the looped ports doesn't have an ESI, the loop goes through the CE switch from Leaf1 to Leaf3.

Figure 57: No ESI on Looped Ports



Lightweight PE-CE Loop Detect Use Cases using Layer 2 Heartbeats

In this section we show the following use cases with lightweight PE-CE loop detect enabled:

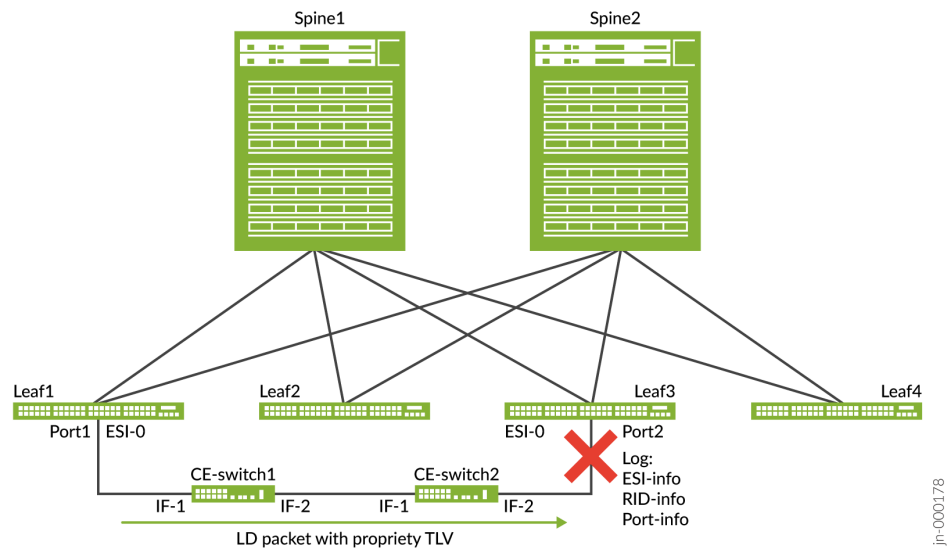
- Use Case 1: The loop is occurring through the switch due to misconfiguration of the switch.
- Use Case 2: The loop is caused by misalignment of cable connections on the switch.

In both use cases, the functionality is not dependent on the BGP speed of control-plane advertisement, and the lightweight PE-CE loop detect is independent of the configured ESI values.

EVPN-VXLAN Lightweight PE-CE Loop Detect Use Case 1

In this first case, the loop occurs at Leaf3. CE-switch1 and CE-switch2 don't have loop detect enabled. Leaf1 and Leaf3 have loop detect enabled. The Layer 2 (L2) loop detect PDU uses a proprietary type, length, value (TLV) format.

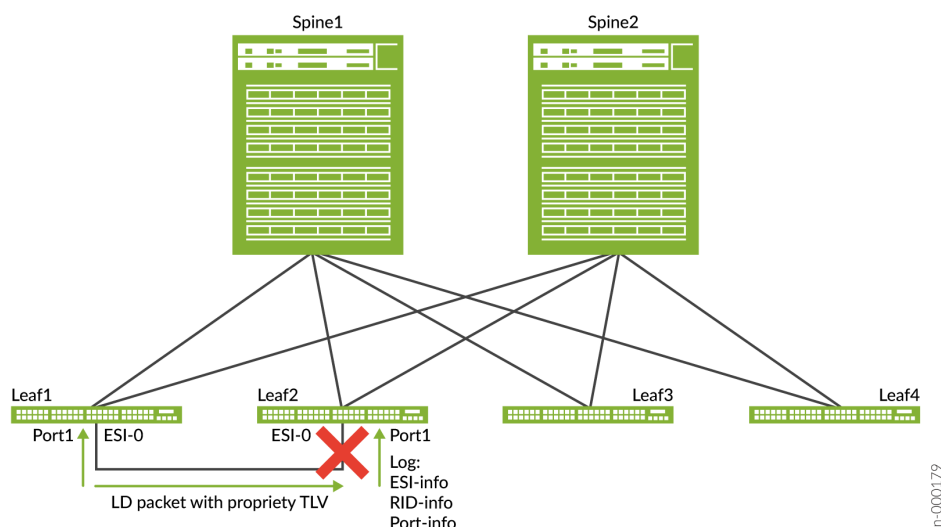
Figure 58: Scenario 1



EVPN-VXLAN Lightweight PE-CE Loop Detect Use Case 2

The L2 loop detect PDU uses a proprietary type, length, and value (TLV) and the loop occurs on Leaf1. Instead of relying on the speed of BGP, the MAC route reflections speed, and the duplicate MAC or MAC move detections in larger DC fabrics, the lightweight PE-CE loop detect is independent of the state of the BGP EVPN signaling.

Figure 59: Scenario 2



Enable Lightweight PE-CE Loop Detect on a Logical Interface

To enable lightweight loop detect for a logical interface or for all logical interfaces, use the `loop-detect` statement at the `[edit protocols]` hierarchy level. Include a supported `loop-detect-action` for the interface(s) and optionally specify a `vlan-id`, as follows:

```
[edit protocols]
set loop-detect enhanced interface (logical-interface-name | all);
set loop-detect enhanced interface (logical-interface-name | all) loop-detect-action (interface-down | laser-off);
set loop-detect enhanced interface (logical-interface-name | all) vlan-id (vlan-id | all);
```



NOTE: We require the `vlan-id` option for trunk interfaces, and enterprise style or service provider style interface configurations.

You can also optionally set the following values at the `[edit protocols loop-detect enhanced interface (logical-interface-name | all)]` hierarchy level:

- The `revert-interval` option—After you repair the loop, the device brings the interface(s) up again after this interval expires (default is 0 seconds).

- The `transmit-interval` option—Customize how often to transmit loop detect PDUs (default is 1 second, see *loop-detect* for more on the values you can set for this interval).

Starting with Junos 24.4 we've added support for monitoring all VLANs on a logical interface with the `vlan-id all` configuration statement option at the [edit protocols *loop-detect* enhanced interface] hierarchy level. This enhancement detects network loops across multiple VLANs and interfaces, improving network stability and performance in scaled environments.

When you configure the `vlan-id all` option on supported devices, the devices show the following behavioral changes for this feature:

- The `revert-interval` configuration is not effective for scaled loop-detect sessions, which makes them non-revertive. You must issue the `clear loop-detect` enhanced interface command to clear the loop condition.
- The receive statistics for loop-detect PDUs do not increment for scaled loop-detect sessions during a loop condition.
- We support only a 1-second transmit interval for scaled loop-detect sessions.

See [Feature Explorer](#) for the platforms that support lightweight PE-CE loop detect as follows:

- [Lightweight PE-CE Loop Detection on EVPN-VXLAN Fabrics](#)—Support for lightweight PE-CE loop detect
- [Lightweight Loop Detection with Scale](#)—Support for scaled lightweight PE-CE loop detect with the `vlan-id all` option

Sample Configuration with Trunk Mode Enterprise Style Interface

The following sample configuration enables lightweight PE-CE loop detect on interface `ge-0/0/1.0`, which is a trunk interface with `vlan-id 100`.

```
set interfaces ge-0/0/1.0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ge-0/0/1.0 unit 0 family ethernet-switching vlan members [ v100 v101 v102 v103
v104 ]
set protocols loop-detect enhanced interface ge-0/0/1.0 vlan-id 100
set protocols loop-detect enhanced interface ge-0/0/1.0 loop-detect-action interface-down
set protocols loop-detect enhanced interface ge-0/0/1.0 transmit-interval 1s
set protocols loop-detect enhanced interface ge-0/0/1.0 revert-interval 50s
```

Sample Configuration with Service Provider Style Interface

The following sample configuration enables loop detect for vlan-id 100 on provider edge (PE) and CE devices with service provider style interface configurations. This configuration doesn't specify a revert interval. As a result, after the device detects a loop and you correct the loop, enter the `clear loop detect enhanced interface name` command to bring the interface back online.

PE device configuration:

```
set interfaces xe-0/0/0 flexible-vlan-tagging
set interfaces xe-0/0/0 encapsulation flexible-ethernet-services
set interfaces xe-0/0/0 unit 10 encapsulation vlan-bridge
set interfaces xe-0/0/0 unit 10 vlan-id 100
set protocols loop-detect enhanced interface xe-0/0/0.10 vlan-id 100
set vlans vlan100 vlan-id 100
set vlans vlan100 interface xe-0/0/0.10
```

CE device configuration:

```
set interfaces xe-0/0/3 flexible-vlan-tagging
set interfaces xe-0/0/3 encapsulation flexible-ethernet-services
set interfaces xe-0/0/3 unit 10 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 10 vlan-id 100
set protocols loop-detect enhanced interface xe-0/0/3.10 vlan-id 100
set vlans vlan100 vlan-id 100
set vlans vlan100 interface xe-0/0/3.10
```

CLI Commands to Display or Clear Loop Detect Status

Use the `show loop-detect enhanced interface` command to display loop status on an interface or all interfaces.

Use the `clear loop-detect enhanced interface` command to restore an interface or all interfaces to their prior state after the device detects a loop and applies a configured action to break the loop.

Show command without any loop

```
user@leaf-device# run show loop-detect enhanced interface
Interface          :ge-0/0/1.0
Vlan-id            :100
```

```

ESI                               :00:00:00:00:00:00:00:00:00:00
Current status                    :Normal[Link Up]
Last loop-detect time            :-
Receive statistics                :0
Action configured                 :Interface-down
Action count                     :0
Transmit Interval                 :1s
Revert Interval                   :60s

```

Show command with loop detect status

```

user@leaf-device# run show loop-detect enhanced interface
Interface                        :ge-0/0/1.0
Vlan-id                          :100
ESI                              :00:00:00:00:00:00:00:00:00:00
Current status                   :Loop-detected
      Remote Host                 :leaf04
      Remote Chassis              :94:f7:ad:94:dd:40
      Remote Interface            :xe-0/0/2.0
      Remote ESI                  :00:00:00:00:00:00:00:00:00:00
Last loop-detect time            :Tue May 26 04:36:37 2020
Receive statistics                :4
Action configured                 :Interface-down
Action count                     :1
Transmit Interval                 :1s
Revert Interval                   :60s

```

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
22.4R1	Starting in Junos OS Release 22.4R1, QFX10002-60C, QFX10002, QFX10008, and QFX10016 support EVPN-VXLAN lightweight PE-CE loop protection on leaf device to server device links with either enterprise style or service provider style interface configurations.

RELATED DOCUMENTATION

loop-detect

```
show loop-detect enhanced interface
```

```
clear loop-detect enhanced interface
```

Overlapping VLAN Support Using VLAN Translation in EVPN-VXLAN Networks

IN THIS SECTION

- [Benefits | 669](#)
- [Overview | 669](#)
- [Supported VLAN Translation Configurations | 670](#)
- [Verify VLAN Translation Mappings on an Interface | 672](#)

You can use VLAN translation to manage overlapping VLAN IDs in an EVPN-VXLAN fabric. For this purpose, we support VLAN translation on devices operating as EVPN provider edge (PE) leaf devices in the fabric. See [Feature Explorer](#) for platform support information.

We support this feature:

- On trunk mode access-side interfaces configured in the enterprise style.
- On leaf devices in edge-routed bridging (ERB) and centrally routed bridging (CRB) overlays.
- With MAC-VRF EVPN routing instances (any supported service types).
- On access side ports that can be single-homed or multihomed.
- On Layer 2 (L2) VXLAN gateway access-side ports.
- On Layer 3 (L3) VXLAN gateway IRB interfaces for VXLAN bridge domains.

On some platforms you can configure overlapping VLANs using other methods besides the VLAN translation configuration we describe here. For example, see "[Overlapping VLAN Support Using Multiple Forwarding Instances or VLAN Normalization](#)" on page 674.

Benefits

- Simplifies re-provisioning a network after combining different business areas in the network that might use the same VLAN IDs for different functions.
- Helps service providers to maintain traffic isolation in the same network among different customers using one or more of the same VLAN IDs.

Overview

When you configure VLAN translation, you map the host VLAN ID in tagged packets coming in on an interface to a configured VLAN ID. We call that configured VLAN ID a *mapped VLAN value*. For ingress packets from the host, the device substitutes the mapped VLAN value for the host VLAN ID before the packet enters the packet processing pipeline. On egress when forwarding tagged traffic toward the host, the device replaces the mapped VLAN value with the host VLAN ID.

You use the usual VLAN configuration statements to define the VLANs you plan to use as mapped VLAN values. You also associate interfaces with those VLANs. Then to configure the VLAN translation, use the `vlan-rewrite translate from-vlan-id to-vlan-id` statement at the [edit interfaces *interface-name* unit *logical-unit-number* family ethernet-switching] for each host VLAN mapping and interface as needed. With this statement:

- The *from-vlan-id* is the host VLAN ID.
- The *to-vlan-id* is the mapped VLAN value.

You can specify host VLAN IDs and mapped VLAN values in the usual VLAN range—1 through 4094.

If you configure an interface with multiple host VLANs and map *some* (but not all) of those host VLANs to mapped VLAN values, the interface:

- Accepts and passes through packets that are tagged with host VLAN IDs that are not mapped (host VLAN IDs that have no corresponding `vlan-rewrite translate` statement).
- Accepts packets that are tagged with a host VLAN ID for which the interface has a `vlan-rewrite translate` configuration. The interface drops packets that are tagged with the corresponding mapped VLAN value.

For example, the following configuration includes VLANs 100 and 101 on interface `xe-0/0/1`. The configuration also establishes VLAN translation for host VLAN ID 200 to mapped VLAN value 100.

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan100
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan101
set vlans vlan100 vlan-id 100
```

```
set vlans vlan101 vlan-id 101

set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 200 100
```

With this configuration, the interface:

- Accepts incoming packets that are tagged with VLAN 101 because that VLAN isn't mapped.
- Accepts incoming packets that are tagged with VLAN 200 because that VLAN is a host VLAN with a vlan-rewrite mapping.
- Drops incoming packets tagged with VLAN 100 because that VLAN is a mapped VLAN value for host VLAN 200.

Supported VLAN Translation Configurations

We support VLAN translation on an interface as follows:

- You can map a host VLAN ID to a mapped VLAN value only on access interfaces with enterprise style interface configurations.
- You can map each host VLAN to one and only one mapped VLAN value.

You'll see a commit error if you try to configure VLAN translation of the same host VLAN to more than one mapped VLAN value.

For example, the following configuration includes VLAN IDs 100 and 101 on interface xe-0/0/1. The configuration also maps host VLAN ID 200 to mapped VLAN value 100.

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan100
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan101
set vlans vlan100 vlan-id 100
set vlans vlan101 vlan-id 101

set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 200 100
```

If you then try to map the same host VLAN 200 to the other configured VLAN 101 on the same interface, the CLI won't allow the commit:

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 200 101
```

- You can map only one host VLAN to the same mapped VLAN value on an interface.

In other words, after you map a host VLAN, you can't map a different host VLAN tag to the same mapped VLAN value on the same interface. The CLI doesn't block the commit operation, but only the most recent mapping will take effect.

For example, you configure VLAN 100 on an interface, and map host VLAN 200 to mapped VLAN value 100:

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan100
set vlans vlan100 vlan-id 100

set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 200 100
```

You commit that configuration, then configure a mapping from host VLAN 300 to the same mapped VLAN value 100:

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 300 100
```

The device overwrites the first mapping with the second one for that mapped VLAN value, and processes packets only according to the second mapping.

- You can map *different* host VLANs to *different* mapped VLAN values on the *same interface*.

For example, if you configure VLANs 100 and 101 on an interface, you can map host VLAN 200 to one mapped VLAN value (100) and host VLAN 300 to another mapped VLAN value (101) on that same interface:

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan100
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan101
set vlans vlan100 vlan-id 100
set vlans vlan101 vlan-id 101

set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 200 100
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 300 101
```

- You can map the *same* host VLAN to the *same* mapped VLAN value on *different interfaces*.

For example, if you configure two interfaces xe-0/0/1 and xe-0/0/2 as members of VLAN 100, you can map host VLAN 200 to the same mapped VLAN value (100) for both interfaces:

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan100
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members vlan100
set vlans vlan100 vlan-id 100

set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 200 100
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan-rewrite translate 200 100
```

- You can map *different* host VLAN tags to the *same* mapped VLAN value on *different* interfaces.

For example, your configuration includes VLAN 100 on interfaces xe-0/0/1 and xe-0/0/2. You can map host VLAN 200 on xe-0/0/1 and host VLAN 300 on xe-0/0/2 to the same mapped VLAN value 100:

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan100
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members vlan100
set vlans vlan100 vlan-id 100

set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 200 100
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan-rewrite translate 300 100
```



NOTE: (QFX5xxx switches) In an EVPN-VXLAN fabric, you can't configure the [native-vlan-id](#) statement on the same interface where you enable VLAN translation with the [vlan-rewrite](#) statement.

Verify VLAN Translation Mappings on an Interface

Run the `show ethernet-switching interface interface-name detail` CLI command to verify the VLAN translation mappings on an interface.

For example, consider again the case where you configure different VLANs (VLAN 100 and VLAN 101) on the same interface, xe-0/0/1. Then you map different host VLAN IDs (200 and 300) to each of those mapped VLAN values (100 and 101, respectively).

```
set interfaces xe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan100
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members vlan101
set vlans vlan100 vlan-id 100
set vlans vlan101 vlan-id 101

set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 200 100
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan-rewrite translate 300 101
```

In the output from the `show ethernet-switching interface xe-0/0/1 detail`, you see the host VLAN ID (200 or 300) in the `Trunk id: output` field. You also see the corresponding mapped VLAN value (100 or 101) in the `VLAN id: output` field, and its configured VLAN name (vlan100 or vlan101) in the `VLAN name: output` field.

```
user@leaf> show ethernet-switching interface xe-0/0/1 detail
```

Information for interface family:

Name: xe-0/0/1.0

Type: IFF

Handle: 0x2701c10

Index: 554

Generation: 186

Flags: UP

IFD index: 659

Routing/Vlan index: 7

IFL index: 554

Address family: 63

Sequence number: 0

MAC sequence number: 0

MAC limit: 294912

MACs learned: 0

Static MACs learned: 0

Non configured static MACs learned: 0

MAC+IP limit: 0

MAC+IPs learned: 0

Name: xe-0/0/1.0

Type: IFBD (static)

Handle: 0x1de8890

Index:

Generation: 147

Trunk id: 200

Flags: UP,

IFD index:

Routing/Vlan index: 5

IFL index:

Address family:

VLAN id: 100

VLAN name: vlan100

Sequence number: 0

MAC sequence number: 0

MAC limit: 294912

MACs learned: 0

Static MACs learned: 0

Non configured static MACs learned: 0

MAC+IP limit: 0	MAC+IPs learned: 0
VSTP index: 9	STP State: Forwarding
Tagging: tagged	Rewrite op: SWAP
Name: xe-0/0/1.0	
Type: IFBD (static)	Handle: 0x1de7c50
Index:	Generation: 148
Trunk id: 300	Flags: UP,
IFD index:	Routing/Vlan index: 6
IFL index:	Address family:
VLAN id: 101	VLAN name: vlan101
Sequence number: 0	MAC sequence number: 0
MAC limit: 294912	MACs learned: 0
Static MACs learned: 0	Non configured static MACs learned: 0
MAC+IP limit: 0	MAC+IPs learned: 0
VSTP index: 9	STP State: Forwarding
Tagging: tagged	Rewrite op: SWAP

Overlapping VLAN Support Using Multiple Forwarding Instances or VLAN Normalization

IN THIS SECTION

- [Benefits | 675](#)
- [Overlapping VLANs Using the Enterprise Style Interface Configuration | 675](#)
- [Overlapping VLANs Using the Service Provider Interface Configuration | 676](#)

In platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS) deployments, each customer uses a separate physical interface to connect to a leaf device. In this situation, you can't use the same VLAN ID for two different tenants in separate MAC-VRF instances that share the same default forwarding instance. However, you can have a VLAN name with the same VLAN-ID (for example, vlan200 with VLAN ID 200) in two different MAC-VRF instances if they each have their own forwarding instances. You can also configure explicit or implicit VLAN normalization using service provider style interface configuration.

You can configure forwarding instance identifiers on the QFX10000 line of switches and ACX7100-32C and ACX7100-48L devices.

We support overlapping VLANs on some platforms that don't support multiple forwarding instances. See ["Overlapping VLAN Support Using VLAN Translation in EVPN-VXLAN Networks" on page 668](#) for more information.



NOTE: On the QFX10000 line of switches, you can configure up to 99 forwarding instance identifiers.

On ACX7100-32C and ACX7100-48L devices, you can configure up to 6 forwarding instances.

Benefits

- Enables overlapping VLANs
- Identifies which customers are sharing VLANs

Overlapping VLANs Using the Enterprise Style Interface Configuration

Overlapping VLANs with Multiple Forwarding Instances

This configuration allows overlapping VLAN-ID 200 in different MAC-VRF instances by configuring separate forwarding instances.

```
set interfaces et-0/0/20:3 unit 0 family ethernet-switching interface-mode trunk
set interfaces et-0/0/20:3 unit 0 family ethernet-switching vlan members vlan200
set interfaces et-0/0/0:0 unit 0 family ethernet-switching interface-mode trunk
set interfaces et-0/0/0:0 unit 0 family ethernet-switching vlan members vlan200
set routing-instances MAC-VRF1 instance-type mac-vrf
set routing-instances MAC-VRF1 forwarding-instance identifier 1
set routing-instances MAC-VRF1 interface et-0/0/20:3.0
set routing-instances MAC-VRF1 vlans vlan200 vlan-id 200
set routing-instances MAC-VRF1 vlans vlan200 vxlan vni 200
set routing-instances MAC-VRF2 instance-type mac-vrf
set routing-instances MAC-VRF2 forwarding-instance identifier 2
set routing-instances MAC-VRF2 interface et-0/0/0:0.0
set routing-instances MAC-VRF2 vlans vlan200 vlan-id 200
set routing-instances MAC-VRF2 vlans vlan200 vxlan vni 2000
```

Overlapping VLANs Using the Service Provider Interface Style Configuration

Implicit Normalization With VLAN ID Sample Configuration

This configuration implicitly translates the customer VLAN ID to the VLAN ID specified in the bridge domain upon ingress. At egress, the VLAN ID used to normalize the bridge domain is removed, and the customer VLAN ID is pushed onto the frame. In this case, the overlapping VLAN IDs of 150 are mapped to VLAN ID 200 and 400, respectively. In this example, the subscriber traffic is transported over the EVPN core using VXLAN VNI assignments that match the normalized VLAN IDs.

```
set interfaces et-0/0/20:3 flexible-vlan-tagging
set interfaces et-0/0/20:3 unit 150 encapsulation vlan-bridges
set interfaces et-0/0/20:3 unit 150 vlan-id 150

set interfaces et-0/0/0 flexible-vlan-tagging
set interfaces et-0/0/0 unit 150 encapsulation vlan-bridge
set interfaces et-0/0/0 unit 150 vlan-id 150

set routing-instances VS1 vlans vlan200 vlan-id 200
set routing-instances VS1 vlans vlan200 interface et-0/0/20:3.150
set routing-instances VS1 vlans vlan200 vxlan vni 200

set routing-instances VS2 vlans vlan400 vlan-id 400
set routing-instances VS2 vlans vlan400 interface et-0/0/0:0.150
set routing-instances VS2 vlans vlan400 vxlan vni 400
```

Implicit Normalization with VLAN ID None Sample Configuration

This configuration implicitly normalizes the two overlapping customer VLANs to unique VLAN IDs within their respective bridge domains. At ingress, the customer VLAN ID is stripped and the (now) untagged traffic is transported over the EVPN VXLAN fabric. At egress on the customer interface, the VLAN tag is pushed back onto the frame. In the EVPN core, the two customer VLANs, which both use VLAN ID 150, map to VXLAN VNIs 200 and 400, respectively.

```
set interfaces et-0/0/20:3 flexible-vlan-tagging
set interfaces et-0/0/20:3 unit 150 encapsulation vlan-bridge
set interfaces et-0/0/20:3 unit 150 vlan-id 150

set interfaces et-0/0/0 flexible-vlan-tagging
set interfaces et-0/0/0 unit 150 encapsulation vlan-bridge
```

```

set interfaces et-0/0/0 unit 150 vlan-id 150

set routing-instances MAC-VRF1 vlans vlan200 vlan-id none
set routing-instances MAC-VRF1 vlans vlan200 interface et-0/0/20:3.150
set routing-instances MAC-VRF1 vlans vlan200 vxlan vni 200

set routing-instances MAC-VRF2 vlans vlan400 vlan-id none
set routing-instances MAC-VRF2 vlans vlan400 interface et-0/0/0:0.150
set routing-instances MAC-VRF2 vlans vlan400 vxlan vni 400

```

Explicit Normalization with VLAN Maps

Explicit normalization does not specify a VLAN ID in the bridge domain. Instead, VLAN map operations are used to manipulate the VLAN label stack to achieve the desired normalization. In this example, two customers use the same VLAN ID of 150. The input and output maps applied to the interface explicitly swap the customer tag with the normalized assignments used in the provider network. In the egress direction, the swap operation causes the interface's VLAN tag to be swapped onto the frame.

Explicit normalization is needed when using the `vlan-bundle` service type for the instance.



NOTE: VLAN map operations are supported only on service provider style interface configurations. IRB interfaces are not supported within the bridge domain with this method.

```

set interfaces ge-2/0/4 flexible-vlan-tagging
set interfaces ge-2/0/4 unit 150 encapsulation vlan-bridge
set interfaces ge-2/0/4 unit 150 vlan-id 150
set interfaces ge-2/0/4 unit 150 input-vlan-map swap
set interfaces ge-2/0/4 unit 150 input-vlan-map vlan-id 200
set interfaces ge-2/0/4 unit 150 output-vlan-map swap

set interfaces ge-2/0/10 flexible-vlan-tagging
set interfaces ge-2/0/10 unit 150 encapsulation vlan-bridge
set interfaces ge-2/0/10 unit 150 vlan-id 150
set interfaces ge-2/0/10 unit 150 input-vlan-map swap
set interfaces ge-2/0/10 unit 150 input-vlan-map vlan-id 400
set interfaces ge-2/0/10 unit 150 output-vlan-map swap

set routing-instances VS4 instance-type mac-vrf
set routing-instances VS4 protocols evpn encapsulation vxlan
set routing-instances VS4 protocols evpn multicast-mode ingress-replication

```

```

set routing-instances VS4 vtep-source-interface lo0.0
set routing-instances VS4 bridge-domains vlanBundle-1 interface ge-2/0/4.150
set routing-instances VS4 bridge-domains vlanBundle-1 vxlan vni 200
set routing-instances VS4 bridge-domains vlanBundle-1 vxlan ingress-node-replication
set routing-instances VS4 service-type vlan-bundle
set routing-instances VS4 route-distinguisher 10.1.0.1:4
set routing-instances VS4 vrf-target target:10:4

set routing-instances VS5 instance-type mac-vrf
set routing-instances VS5 protocols evpn encapsulation vxlan
set routing-instances VS5 protocols evpn multicast-mode ingress-replication
set routing-instances VS5 vtep-source-interface lo0.0
set routing-instances VS5 bridge-domains vlanBundle-1 interface ge-2/0/10.150
set routing-instances VS5 bridge-domains vlanBundle-1 vxlan vni 400
set routing-instances VS5 bridge-domains vlanBundle-1 vxlan ingress-node-replication
set routing-instances VS5 service-type vlan-bundle
set routing-instances VS5 route-distinguisher 10.1.0.1:5
set routing-instances VS5 vrf-target target:10:5

```

Multiple Forwarding Instances Sample Configuration

This configuration is similar to the enterprise style interface configuration. You use separate forwarding instances to allow overlapping VLAN-ID 200 in different MAC-VRF instances.

```

set interfaces et-0/0/20:3 flexible-vlan-tagging
set interfaces et-0/0/20:3 unit 200 encapsulation vlan-bridge
set interfaces et-0/0/20:3 unit 200 vlan-id 200

set interfaces et-0/0/0 flexible-vlan-tagging
set interfaces et-0/0/0 unit 200 encapsulation vlan-bridge
set interfaces et-0/0/0 unit 200 vlan-id 200

set interfaces et-0/0/1 flexible-vlan-tagging
set interfaces et-0/0/1 unit 200 encapsulation vlan-bridge
set interfaces et-0/0/1 unit 200 vlan-id-list 1000-2000
set routing-instances MAC-VRF1 instance-type mac-vrf
set routing-instances MAC-VRF1 forwarding-instance identifier 1
set routing-instances MAC-VRF1 vlans vlan200 vlan-id 200
set routing-instances MAC-VRF1 vlans vlan200 interface et-0/0/20:3.200
set routing-instances MAC-VRF1 vlans vlan200 vxlan vni 200

```

```

set routing-instances MAC-VRF2 instance-type mac-vrf
set routing-instances MAC-VRF2 forwarding-instance identifier 2
set routing-instances MAC-VRF2 vlans vlan200 vlan-id 200
set routing-instances MAC-VRF2 vlans vlan200 interface et-0/0/0:0.200
set routing-instances MAC-VRF2 vlans vlan200 interface et-0/0/1.200
set routing-instances MAC-VRF2 vlans vlan200 vxlan vni 2000

```

RELATED DOCUMENTATION

[Overlapping VLAN Support Using VLAN Translation in EVPN-VXLAN Networks | 668](#)

vlan-rewrite

Layer 2 Protocol Tunneling over VXLAN Tunnels in EVPN-VXLAN Bridged Overlay Networks

SUMMARY

Enables Layer 2 (L2) protocol tunneling (L2PT) for protocol control frames from an access interface to VXLAN tunnel destinations in an EVPN-VXLAN bridged overlay (BO) network.

IN THIS SECTION

- [L2PT over VXLAN Tunnels | 679](#)
- [Platform-Specific L2PT over VXLAN Behavior | 684](#)
- [Configure and Verify L2PT over VXLAN | 685](#)

L2PT over VXLAN Tunnels

IN THIS SECTION

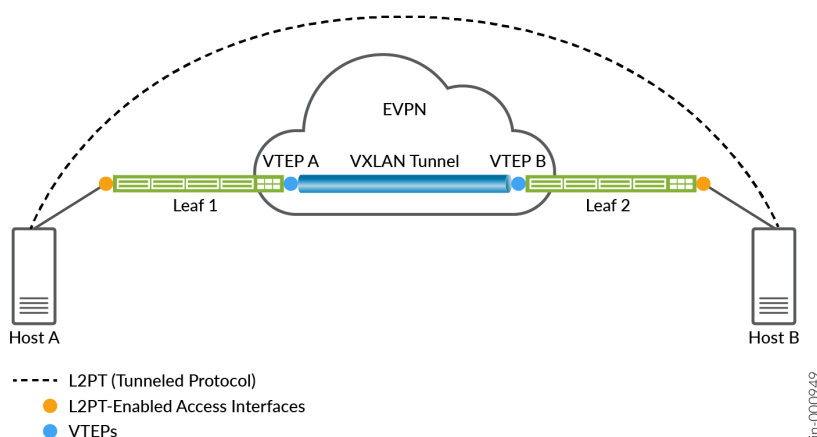
- [Benefits of L2PT over VXLAN Tunnels | 681](#)
- [L2PT over VXLAN Support Details | 681](#)
- [Supported Protocols to Tunnel | 682](#)

L2PT transmits bridge protocol data units (BPDUs) transparently across a bridged network between devices as if those devices are directly connected for those protocol control frames. The nodes in the transmission path don't trap and process the control frames for the tunneled protocols.

VXLAN tunnels in an EVPN-VXLAN bridged overlay (BO) network help isolate one customer's traffic from the traffic of other customers across the network.

L2PT over VXLAN tunnels enables a device in an EVPN-VXLAN bridged overlay network to transmit tagged or untagged protocol control frames transparently to destinations across the VXLAN tunnel endpoints (VTEPs) in the network. The VXLAN tunnels keep each customer's traffic separated, and L2PT prevents the devices in the network along the transmission path from processing the protocol frames. See [Figure 60 on page 680](#):

Figure 60: L2PT over VXLAN Tunnels



You enable L2PT on an access interface for the protocols you want to tunnel through the EVPN-VXLAN network. When control frames come into the device on that access interface, the device doesn't trap those frames to the control plane for protocol processing. Instead, the device encapsulates the frame with a VXLAN media access control (MAC) UDP header and VXLAN network identifier (VNI), and transparently floods the frames over the associated VXLAN tunnels toward the destination host. In other words, the devices in the EVPN-VXLAN network send those frames across the network the same way they send data packets.

The protocol control packets can be VLAN-tagged or untagged. The devices use the following VNI values for the VXLAN encapsulation:

- **VLAN-tagged**—The device uses the VNI corresponding to the control packet's VLAN.
- **Untagged**—The device uses the VNI corresponding to the native VLAN.

The device at the destination VTEP terminates the VXLAN tunnel, decapsulates the frame, and floods the frame toward the destination access interfaces.

You can use L2PT over VXLAN in cases such as when your network has hosts attached to leaf devices over remote connections in the EVPN-VXLAN network, and the devices use LLDP to discover EVPN-VXLAN neighbor relationships.

To configure this feature for L2 protocol control frames coming in on a specified access interface name, you must configure both of the following statements at the `[edit protocols layer2-control l2pt interface interface-name]` hierarchy level:

- `destination vxlan-tunnel`—Enable L2PT in general toward network destinations that are across the VXLAN tunnels in the network.
- `protocol (all | protocol-name)`—Specify an L2 protocol to tunnel. Use the `all` option to tunnel all supported protocols, or include individual `protocol` statements for each protocol you want to tunnel.

See ["Configure and Verify L2PT over VXLAN" on page 685](#) for more on how to configure this feature.

Benefits of L2PT over VXLAN Tunnels

- Provides a secure option for network administrators to extend tagged or untagged L2 BPDUs across the EVPN-VXLAN BO fabric.

L2PT over VXLAN Support Details

Use [Feature Explorer](#) to confirm platform and release support for this feature.

See ["Platform-Specific L2PT over VXLAN Behavior" on page 684](#) for a summary of platform-specific support differences. This section describes common supported behavior with this feature on all platforms.

We support L2PT over VXLAN tunnels in an EVPN-VXLAN BO network as follows:

- ["Supported Protocols to Tunnel" on page 682](#) lists the L2 protocols you can configure the device to transparently forward over VXLAN tunnels.
- If you enable an L2 protocol for L2PT over VXLAN on an interface, you can't configure that interface with the same protocol for data traffic on the device. The device throws a commit error in that case.
- When you enable L2PT over VXLAN for a supported protocol, the device transparently forwards the protocol control frames with that protocol destination MAC address. If you enable L2PT for a protocol that has the same destination MAC address as one or more other supported protocols, the device transparently forwards the control frames for all of those protocols.

For example, LACP and 802.3AH protocol use the same destination MAC address, 01:80:C2:00:00:02. If you configure L2PT over VXLAN for LACP, the device tunnels 802.3ah

protocol frames as well as LACP frames. (See ["Supported Protocols to Tunnel" on page 682](#) for the standard destination MAC addresses for each supported protocol.)

- The device can tunnel L2 protocol frames that are VLAN-tagged or are untagged. We support Q-in-Q for the tunneled traffic according to the use cases described in ["Examples: Tunneling Q-in-Q Traffic in an EVPN-VXLAN Overlay Network" on page 1326](#).

To send tagged control BPDUs over the VXLAN tunnels, the device uses the VNI corresponding to the VLAN in the tagged frame. To send untagged control BPDUs over the VXLAN tunnels, the device uses the native VLAN and its corresponding VNI.



NOTE: For L2PT over VXLAN to work, you must configure a VLAN as the native VLAN and a VNI mapping for the native VLAN on the interface.

- The specified access interface can be an interface that is:
 - A physical interface or an aggregated Ethernet (AE) interface.
 - Configured in the enterprise style or the service provider style. (See *Flexible Ethernet Services Encapsulation* and ["Understanding Flexible Ethernet Services Support With EVPN-VXLAN" on page 654](#).)
 - A single-homed or multihomed connection.
- You can't configure L2PT over VXLAN and standard L2PT (for the same or different protocols) on the same interface. See *Layer 2 Protocol Tunneling* for details on the standard L2PT feature (not over VXLAN encapsulation tunnels), which uses a MAC rewrite operation to perform the L2 tunneling. The MAC rewrite statements and commands described there apply only to the standard L2PT feature and not to L2PT over VXLAN.

However, you can configure L2PT over VXLAN on one interface and standard L2PT on a different interface (for the same or different protocols). You can also use the `show l2pt interface` command to see if L2PT over VXLAN or standard L2PT is enabled for an interface. In the Destination output field, the command displays VXLAN-TUNNEL for interfaces you enable with L2PT over VXLAN, or the MAC rewrite address for interfaces you enable with standard L2PT.

Supported Protocols to Tunnel

See ["Platform-Specific L2PT over VXLAN Behavior" on page 684](#) for platform-specific differences for supported protocol options.

You can configure the device to transparently tunnel BPDUs for the protocols listed in [Table 35 on page 683](#). The table lists the options you configure at the [edit protocols layer2-control l2pt interface *interface-name* protocol] hierarchy level to enable L2PT for each protocol, as well as the protocol's standard destination MAC address.

The device tunnels the protocol control frames based on the destination MAC address. As a result, the device will tunnel all protocols that share the same destination MAC address if you enable L2PT for any one of the protocols with that destination MAC address.

Table 35: Protocol Options with L2PT over VXLAN Tunnels

Protocol	Protocol Name Configuration Option	Protocol MAC Address
802.1X—IEEE 802.1X authentication	ieee8021x	01:80:C2:00:00:03
802.3ah—IEEE 802.3ah Operation, Administration, and Maintenance (OAM) link fault management (LFM)	ieee8023ah	01:80:C2:00:00:02
Cisco Discovery Protocol (CDP)	cdp	01:00:0C:CC:CC:CC
Ethernet local management interface (E-LMI)	elmi	01:80:C2:00:00:07
Generic Attribute Registration Protocol (GARP) VLAN Registration Protocol (GVRP)	gvrp	01:80:C2:00:00:21
Link Aggregation Control Protocol (LACP)	lacp	01:80:C2:00:00:02
Link Layer Discovery Protocol (LLDP)	lldp	01:80:C2:00:00:0E
Multiple MAC Registration Protocol (MMRP)	mmrp	01:80:C2:00:00:20
MVRP VLAN Registration Protocol (MVRP)	mvrp	01:80:c2:00:00:21
Per-VLAN Spanning Tree (PVST) protocol and PVST plus (PVST+) protocol	pvstp	01:00:0C:CC:CC:CD
STP, Rapid STP (RSTP), and Multiple STP (MSTP)	stp	01:80:C2:00:00:00
Unidirectional Link Detection (UDLD)	udld	01:00:0C:CC:CC:CC

Table 35: Protocol Options with L2PT over VXLAN Tunnels (Continued)

Protocol	Protocol Name Configuration Option	Protocol MAC Address
VLAN Spanning Tree Protocol (VSTP)	vstp	01:00:0C:CC:CC:CD
VLAN Trunking Protocol (VTP)	vtp	01:00:0C:CC:CC:CC

Platform-Specific L2PT over VXLAN Behavior

Use [Feature Explorer](#) to confirm platform and release support for this feature.

Use the following table to review platform-specific behaviors for your platforms that support this feature.

Table 36: Platform Specific L2PT Over VXLAN Differences

Platform	Difference
ACX Series (Junos OS Evolved)	<ul style="list-style-type: none"> These platforms do not support GVRP, UDLD, or VSTP.
EX Series and QFX Series (Junos OS)	<ul style="list-style-type: none"> On these devices, you don't need to explicitly enable tunneling BPDUs in the STP family of protocols (called xSTP here for brevity, which includes STP, MSTP, RSTP, VSTP, and PVST/PVST+). These devices flood xSTP control BPDUs into the VXLAN tunnels by default, so we don't support the protocol options in Table 35 on page 683 to tunnel xSTP protocols over VXLAN. If you don't want the device to tunnel xSTP control BPDUs over VXLAN, you can configure the BPDU protection feature on the ingress access interface. See <i>bpdu-block</i> and <i>bpdu-block-on-edge</i> for details on the BPDU protection feature.

Table 36: Platform Specific L2PT Over VXLAN Differences (*Continued*)

Platform	Difference
QFX Series (Junos OS Evolved)	<ul style="list-style-type: none"> • These devices do not tunnel xSTP protocol BPDUs over VXLAN by default. We provide the <code>pvstp</code>, <code>stp</code>, and <code>vstp</code> protocol options on these devices to enable tunneling xSTP BPDUs over VXLAN (see Table 35 on page 683). • On these devices, you must set the <code>enable-all-ifl</code> option at the <code>[edit protocols layer2-control l2pt interface <i>interface-name</i>]</code> hierarchy level on any L2PT-enabled physical interface that you configure with multiple logical interface units. See Step 3 in "Configure and Verify L2PT over VXLAN" on page 685 for details. • These devices implement L2PT over VXLAN using protocol profiles. A protocol profile is a set of protocols to tunnel on an interface. Each distinct set of protocols is a different profile, and the same profile can be shared on multiple interfaces. However, these devices support only up to 8 profiles, which limits the number of protocol combinations you can configure to tunnel on the device.

Configure and Verify L2PT over VXLAN

To enable this feature, you configure statements in the `l2pt` stanza at the `[edit protocols layer2-control]` hierarchy level.

Configure and verify L2PT over VXLAN on the device as follows:

1. Enable this feature on the access interface from which you want to use L2PT to forward any received control BPDUs transparently across the device's VXLAN tunnels. You must include the interface name and the destination `vxlan-tunnel` options:

```
set protocols layer2-control l2pt interface interface-name destination vxlan-tunnel
```

For example, you might configure a service provider style access interface (et-0/1/1:3) and enable L2PT over VXLAN for that interface as follows:

```
set protocols layer2-control l2pt interface et-0/1/1:3 destination vxlan-tunnel
```

2. Set protocol you want to tunnel when control BPDUs arrive on that interface. See the supported protocol options in ["Supported Protocols to Tunnel" on page 682](#).

```
set protocols layer2-control l2pt interface interface-name protocol protocol-name
```

Configure additional protocol *protocol-name* statements at this hierarchy for each protocol you want to tunnel from this interface.

For example, configure the following if you want to transparently tunnel LLDP and VTP control frames coming in on interface et-0/1/1:3:

```
set protocols layer2-control l2pt interface et-0/1/1:3 protocol lldp
set protocols layer2-control l2pt interface et-0/1/1:3 protocol vtp
```

To tunnel control BPDUs from *all* of the supported protocols from this interface, you can use a single statement and specify the *all* option, as follows:

```
set protocols layer2-control l2pt interface interface-name protocol all
```

3. (Required on supported Junos OS Evolved QFX Series switches) Configure the *enable-all-ifl* option on the L2PT-enabled interface.

When you commit a configuration update, the commit operation doesn't necessarily apply the statements in a predictable order. In a configuration update in which you enable L2PT on an interface, if the configuration update includes any logical interfaces on that physical interface, the commit operation might not also tag all of the logical interfaces as having L2PT enabled. The *enable-all-ifl* option ensures that all of the logical interfaces on the specified physical interface have the same L2PT settings.

```
set protocols layer2-control l2pt interface interface-name enable-all-ifl
```

4. Repeat steps 1 through 3 for each access interface from which you want to tunnel control BPDUs for one or more protocols.
5. Use the `show l2pt interface` CLI command to verify L2PT over VXLAN operation. This command displays the interfaces and protocols for which the device transparently tunnels control BPDUs across its VXLAN tunnels.

For example:

```
user@device> show l2pt interface
```

Interface	Destination	Protocols
xe-0/0/1	VXLAN-TUNNEL	8021X 8023AH ELMI LACP LLDP MMRP MVRP
ae10	VXLAN-TUNNEL	UDLD

Or to display details only for a specified interface, such as et-0/1/1:3:

```
user@device> show l2pt interface et-0/1/1:3 detail
```

```
Interface: et-0/1/1:3
Destination: VXLAN-TUNNEL
Protocols: LLDP VTP
```

RELATED DOCUMENTATION

l2pt (Destination Tunnels)

show l2pt interface

MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment

IN THIS SECTION

- Benefits of MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment | 689
- MAC Filtering | 689
- Storm Control | 693
- Port Mirroring and Analyzers | 694
- Configuring Remote Mirroring with GRE Encapsulation | 700
- Configuring Remote Mirroring with VXLAN Encapsulation | 702

We support MAC filtering, storm control, and port mirroring and analyzing in an Ethernet VPN-Virtual Extensible LAN (EVPN-VXLAN) overlay network.

We support each of these features using the enterprise style for interface configuration.

We also support these features using service provider (SP) style for interface configuration with some limitations:

- We support port mirroring with a firewall filter in the input direction using the SP style interface configuration, but not in the output direction.
- We support storm control only on a single logical interface in SP style physical interface configurations. You can't configure storm control on multiple logical interfaces with SP style configurations.
 - Due to hardware limitations, storm control applied to a logical interface also applies to the underlying physical interface.
 - The logical interface with storm control configured logs the storm, but any logical interface on the physical interface can trigger storm control.

We support these features only in an EVPN-VXLAN edge-routed bridging (ERB) overlay, which is also called an EVPN-VXLAN topology with a collapsed IP fabric. This overlay network includes the following components:

- A single layer of Juniper Networks switches—for example, QFX10002, QFX5120, or QFX5110 switches—each of which functions as both a Layer 3 spine device and a Layer 2 leaf device.
- Customer edge (CE) devices that are single-homed or multihomed in active/active mode to the spine-leaf devices.



NOTE: We support both local and remote port mirroring with EVPN-VXLAN on some platforms:

- local—The packets are mirrored to a destination on the same device.
- remote—The packets are mirrored to a destination on a remote device.

If you want to use remote port mirroring with EVPN-VXLAN, be sure to look at the Feature Explorer page for [Remote port mirroring with VXLAN encapsulation](#) to see the supported platforms and releases.

See ["Port Mirroring and Analyzers" on page 694](#) for more on local or remote port mirroring with EVPN-VXLAN.

This topic includes the following information:

Benefits of MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment

- MAC filtering enables you to filter and accept packets from ingress CE-facing interfaces, thereby reducing the volume of associated MAC addresses in the Ethernet switching table and traffic in a VXLAN.
- Storm control enables you to monitor traffic levels on EVPN-VXLAN interfaces, and if a specified traffic level is exceeded, drop broadcast, unknown unicast, and multicast (BUM) packets and on some Juniper Networks switches, disable the interface for a specified amount of time. This feature can prevent excessive traffic from degrading the network.
- With port mirroring and analyzers, you can analyze traffic down to the packet level in an EVPN-VXLAN environment. You can use this feature to enforce policies related to network usage and file sharing and to identify problem sources by locating abnormal or heavy bandwidth usage by particular stations or applications.

MAC Filtering

MAC filtering enables you to filter MAC addresses and accept traffic. We support this feature only on ingress CE-facing interfaces, which are interfaces on which VXLAN encapsulation is typically not enabled. To use this feature, you must do the following:

- Create a firewall filter in which you specify one or more of the supported match conditions in [Table 37 on page 690](#) and [Table 38 on page 691](#).
- Apply the firewall filter to a Layer 2 interface configured in the `[edit interfaces interface-name unit logical-unit-number family ethernet-switching filter]` hierarchy.

Table 37: Match Conditions Supported on QFX5100 and QFX5110 Switches

Match Conditions	Interface Input Filter Support	Interface Output Filter Support
Source MAC address	X	X
Destination MAC address	X	X
User VLAN ID	X	X
Source port	X	
Destination port	X	
Ether-type	X	
IP protocol	X	
IP precedence	X	
ICMP codes	X	
TCP flags	X	
IP address	X	



NOTE: In Junos OS Release 18.4R1, QFX5100 and QFX5110 switches support MAC filtering only on an interface. And, starting in Junos OS Release 18.4R2 and in later releases, QFX5100, QFX5110, QFX5120-48Y, and EX4650-48Y switches also support MAC filtering on a VXLAN-mapped VLAN. Junos OS Release 22.2 includes support Mac Filtering and Transit VNI Matching for pure IPv6 underlay on QFX10002, QFX10008, and QFX10016 devices.

Table 38: Match Conditions Supported on QFX10000 Switches

Match Conditions	Interface Input Filter Support	Interface Output Filter Support
Source MAC address	X	
Destination MAC address	X	
User VLAN ID		
Source port	X	X
Destination port	X	X
Ether-type	X	X
IP protocol	X	
IP precedence	X	X
ICMP codes	X	X
TCP flags	X	X
IP address	X	X



NOTE: When configuring MAC filters on QFX10000 switches, keep the following in mind:

- You can apply a filter to an interface only. You cannot apply a filter to a VXLAN-mapped VLAN.
- We do not support a mix of Layer 2 match conditions and Layer 3/Layer 4 match conditions in the same firewall filter. For example, if you include source MAC address

and source port match conditions in the same firewall filter on a QFX10002 switch, the firewall filter will not work.

- We do not support the user VLAN ID match condition. Therefore, if you need to filter logical interfaces, each of which is mapped to a particular VLAN, you must use the service provider style of configuration when configuring the physical interface and associated logical interfaces. After creating a firewall filter, you must then apply the filter to each logical interface to achieve the effect of the user VLAN ID match condition.
- With Junos OS Release 22.2, support is also implemented for VxLAN Network ID (VNI) matching on source/destination IP outer headers for transit traffic on a Layer-3 interface for QFX10002, QFX10008, and QFX10016 devices. VNI matches are made on outer headers only and on ingress traffic only. On transit devices that route tunnel packets, MAC filtering must support the matching of VNI in the outer header, along with outer header source and destination IPv6 addresses as match conditions. Use the VNI match filter under the vxlan match CLI option for the set firewall family inet6 filter term from the <vxlan [vni <vni-id>]> command. Use the show firewall filter command to display statistics.

Through the firewall filter, you specify MAC addresses associated with a VXLAN that are allowed on a particular interface.



NOTE: After you apply the firewall filter to a Layer 2 interface, the interface resides under the default-switch instance.

The following sample configuration on a QFX5110 switch creates a firewall filter named DHCP-Discover-In that accepts and counts incoming traffic that meets multiple match conditions (source MAC address, destination MAC address, destination ports, and VLAN ID) on Layer 2 logical interface xe-0/0/6.0:

```
set firewall family ethernet-switching filter DHCP-Discover-In term 1 from source-mac-address
00:00:5E:00:53:ab/48
set firewall family ethernet-switching filter DHCP-Discover-In term 1 from destination-mac-
address ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter DHCP-Discover-In term 1 from destination-port dhcp
set firewall family ethernet-switching filter DHCP-Discover-In term 1 from destination-port
bootps
set firewall family ethernet-switching filter DHCP-Discover-In term 1 from destination-port
bootpc
set firewall family ethernet-switching filter DHCP-Discover-In term 1 from user-vlan-id 803
```

```

set firewall family ethernet-switching filter DHCP-Discover-In term 1 then accept
set firewall family ethernet-switching filter DHCP-Discover-In term 1 then count DHCP-Discover-In
set firewall family ethernet-switching filter DHCP-Discover-In term 2 then accept
set interfaces xe-0/0/6 unit 0 family ethernet-switching filter input DHCP-Discover-In

```

Storm Control

By default, storm control is enabled on QFX and EX switches for Layer 2 interfaces that are associated with VXLANs. The storm control level is set to 80 percent of the combined BUM traffic streams.



NOTE: EVPN-VXLAN storm control works a bit differently on ACX series platforms. For details, see *Platform Specific Storm Control Behavior*.

In an EVPN-VXLAN environment, storm control is implemented and configured on Layer 2 interfaces that are associated with VXLANs the same as in a non-EVPN-VXLAN environment except for the following differences:

- In an EVPN-VXLAN environment, the traffic types that storm control monitors are as follows:
 - Layer 2 BUM traffic that originates in a VXLAN and is forwarded to interfaces within the same VXLAN.
 - Layer 3 multicast traffic that is received by an integrated routing and bridging (IRB) interface in a VXLAN and is forwarded to interfaces in another VXLAN.
- After creating a storm control profile, you must bind it to an ingress Layer 2 interface at the [edit interfaces *interface-name* unit *logical-unit-number* family ethernet-switching] hierarchy.



NOTE: After you bind the profile to a Layer 2 interface, the interface resides within the default-switch instance.

- If the traffic streams on an interface exceed the specified storm control level, the Juniper Networks switch drops the excess packets, which is known as *rate limiting*. In addition, QFX10000 switches in an EVPN-VXLAN environment support the disabling of the interface for a specified amount of time using the action-shutdown configuration statement at the [edit forwarding-options storm-control-profiles] hierarchy level and the recovery-timeout configuration statement at the [edit interfaces *interface-name* unit *logical-unit-number* family ethernet-switching] hierarchy level.



NOTE: QFX5100 and QFX5110 switches in an EVPN-VXLAN environment do not support the disabling of the interface for a specified amount of time.



NOTE: On QFX5110 switches, if you configure enhanced storm control and a native analyzer on an interface, and the native analyzer has the VxLAN VLAN as input, the shutdown action will not work for the VLAN on that interface. Rate limiting will work as expected.

The following configuration creates a profile named `scp`, which specifies that if the bandwidth used by the combined BUM traffic streams exceeds 5 percent on Layer 2 logical interface `et-0/0/23.0`, the interface drops the excess BUM traffic.

```
set forwarding-options storm-control-profiles scp all bandwidth-percentage 5
set interfaces et-0/0/23 unit 0 family ethernet-switching storm-control scp
```

The following configuration creates a profile named `scp`, which specifies that if the bandwidth used by the multicast traffic stream (broadcast and unknown unicast traffic streams are excluded) exceeds 5 percent on Layer 2 logical interface `et-0/0/23.0`, the interface drops the excess multicast traffic.

```
set forwarding-options storm-control-profiles scp all bandwidth-percentage 5 no-broadcast no-unknown-unicast
set interfaces et-0/0/23 unit 0 family ethernet-switching storm-control scp
```

The following configuration on a QFX10000 switch creates the same profile as in the previous configuration. However, instead of implicitly dropping multicast traffic if the traffic stream exceeds 5 percent, the following configuration explicitly disables the interface for 120 seconds and then brings the interface back up.

```
set forwarding-options storm-control-profiles scp all bandwidth-percentage 5 no-broadcast no-unknown-unicast
set forwarding-options storm-control-profiles scp all action-shutdown
set interfaces ge-0/0/0 unit 0 family ethernet-switching storm-control scp recovery-timeout 120
```

Port Mirroring and Analyzers

To analyze traffic in an EVPN-VXLAN environment, we support the following port mirroring and analyzer functionality:

- Local mirroring
 - On an interface
 - On a VXLAN
- Remote mirroring
 - On an interface
 - On a VXLAN

The following sections provide more information about the supported functionality and include sample configurations.

Local Mirroring



NOTE: Local mirroring is comparable to Switched Port Analyzer (SPAN).

Table 39: Local Mirroring Support

Entity to Which Local Mirroring Is Applied	Traffic Direction	Filter-based Support	Analyzer-based Support
CE-facing interface	Ingress	Supported. See Use Case 1: Sample Configuration.	Supported. See Use Case 2: Sample Configuration.
CE-facing interface	Egress	Not supported.	Supported; however, egress mirrored traffic might carry incorrect VLAN tags that differ from the tags in the original traffic. See Use Case 3: Sample Configuration.
IP fabric-facing interface	Ingress	Supported.	Supported. See Use Case 4: Sample Configuration.

Table 39: Local Mirroring Support (Continued)

Entity to Which Local Mirroring Is Applied	Traffic Direction	Filter-based Support	Analyzer-based Support
IP fabric-facing interface	Egress	Not supported.	Supported. However, the decision to mirror happens at ingress so the layer 2 header will not be the same as a switched or routed packet. Mirrored VXLAN-encapsulated packets will not include a VXLAN header. See Use Case 5: Sample Configuration.
VXLAN-mapped VLAN	Ingress	Supported.	Supported only for traffic entering through a CE-facing interface. See Use Case 6: Sample Configuration.

Configuring Local Mirroring

Use Case 1: Firewall filter-based

Through the use of a port mirroring instance named pm1 and a firewall filter, this configuration specifies that Layer 2 traffic that enters VXLAN100 through logical interface xe-0/0/8.0 is mirrored to an analyzer on logical interface xe-0/0/6.0 and then to port mirroring instance pm1.

```

set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members VXLAN100
set interfaces xe-0/0/8 unit 0 family ethernet-switching filter input IPACL
set interfaces xe-0/0/6 unit 0 family ethernet-switching
set forwarding-options port-mirroring instance pm1 family ethernet-switching output interface
xe-0/0/6
set firewall family ethernet-switching filter IPACL term to-analyzer then port-mirror-instance
pm1

```

Use Case 2: Analyzer-based

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that enters logical interface xe-0/0/8.0 is mirrored to an analyzer on logical interface xe-0/0/6.0.

```
set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members VXLAN100
set interfaces xe-0/0/6 unit 0 family ethernet-switching
set forwarding-options analyzer ANA1 input ingress interface xe-0/0/8.0
set forwarding-options analyzer ANA1 output interface xe-0/0/6.0
```

Use Case 3: Analyzer-based

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that exits logical interface xe-0/0/8.0 is mirrored to an analyzer on logical interface xe-0/0/6.0.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members VXLAN100
set interfaces xe-0/0/6 unit 0 family ethernet-switching
set forwarding-options analyzer test input egress interface xe-0/0/8
set forwarding-options analyzer test output interface xe-0/0/6
```

Use Case 4: Analyzer-based

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that enters logical interface xe-0/0/29.0 is mirrored to an analyzer on logical interface xe-0/0/6.0.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/29 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/29 unit 0 family ethernet-switching vlan members VXLAN100
set interfaces xe-0/0/6 unit 0 family ethernet-switching
set forwarding-options analyzer test input ingress interface xe-0/0/29
set forwarding-options analyzer test output interface xe-0/0/6
```

Use Case 5: Analyzer-based

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that exits logical interface xe-0/0/29.0 is mirrored to an analyzer on logical interface xe-0/0/6.0.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/29 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/29 unit 0 family ethernet-switching vlan members VXLAN100
set interfaces xe-0/0/6 unit 0 family ethernet-switching
set forwarding-options analyzer test input egress interface xe-0/0/29
set forwarding-options analyzer test output interface xe-0/0/6
```

Use Case 6: Analyzer-based

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that enters the VLAN named VXLAN100 and is mirrored to an analyzer on logical interface xe-0/0/6.0.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members VXLAN100
set interfaces xe-0/0/6 unit 0 family ethernet-switching
set forwarding-options analyzer test input ingress vlan VXLAN100
set forwarding-options analyzer test output interface xe-0/0/6
```

Remote Mirroring

Remote port mirroring is used when the output destination is not on the same switch as the source. Remote mirroring delivers the mirrored traffic to one or more remote destination hosts. It is often used in the data center environment for troubleshooting or monitoring.

In an EVPN-VXLAN environment, the mirrored traffic flow at the source switch is encapsulated and tunneled through the underlay IP fabric to the destination host IP address. We support the following types of encapsulation:

- Generic routing encapsulation (GRE) is used with remote mirroring to encapsulate the traffic between switches separated by a routing domain. In an EVPN-VXLAN overlay, GRE encapsulation enables mirroring between leaf devices over the IP fabric. Use remote mirroring with GRE when the mirroring destination hosts are connected to a switch that is part of the same fabric as the source switch. Remote mirroring with GRE encapsulation is comparable to encapsulated remote SPAN (ERSPAN).



NOTE: On ACX7100-32C and ACX7100-48L platforms, the comparable version of ERSPAN is ERSPAN version 2 (ERSPAN v2).

- VXLAN encapsulation supports remote mirroring for EVPN-VXLAN when the source and the output destinations are in separate VNI domains. You must configure a specific VXLAN for mirroring traffic for the output destination interfaces and mapped to a VNI. Remote mirroring with VXLAN encapsulation is comparable to remote SPAN (RSPAN).

Table 40: Remote Mirroring Support

Entity to Which Remote Mirroring Is Applied	Traffic Direction	Filter-based Support	Analyzer-based Support
CE-facing interface	Ingress	Supported. Not supported on ACX7100.	Supported. See Use Case 1: Sample Configuration. Supported on ACX7100. However, mirrored packets include a GRE header.
CE-facing interface	Egress	Not supported.	Supported. See Use Case 2: Sample Configuration. Supported on ACX7100. However, mirrored packets include a GRE header.
IP fabric-facing interface	Ingress	Supported. Not supported on ACX7100.	Supported. See Use Case 3: Sample Configuration. Supported on ACX7100. However, mirrored VXLAN-encapsulated packets include a VXLAN header and a GRE header.
IP fabric-facing interface	Egress	Not supported.	Supported. However, the decision to mirror happens at ingress so the Layer 2 header will not be the same as a switched or routed packet. See Use Case 4: Sample Configuration. NOTE: Mirrored traffic might include a bogus VLAN ID tag of 4094 on the native MAC frame. Supported on ACX7100. However, mirrored packets include a GRE header but do not include a VXLAN header.

Table 40: Remote Mirroring Support (Continued)

Entity to Which Remote Mirroring Is Applied	Traffic Direction	Filter-based Support	Analyzer-based Support
VXLAN-mapped VLAN	Ingress	Supported. Not supported on ACX7100.	Supported only for traffic entering a CE-facing interface. See Use Case 5: Sample Configuration. Not supported on ACX7100.

Configuring Remote Mirroring with GRE Encapsulation

The following sample configurations are for analyzer-based remote mirroring with GRE encapsulation.



NOTE: For an example of configuring a remote analyzer with GRE encapsulation on ACX7100, see [Example: Enable a Remote Analyzer Instance at an ESI-LAG Interface](#).

Use Case 1

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that enters logical interface xe-0/0/8.0 is mirrored to a remote logical interface with an IP address of 10.9.9.2.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members VXLAN100
set forwarding-options analyzer test input ingress interface xe-0/0/8.0
set forwarding-options analyzer test output ip-address 10.9.9.2
```

Use Case 2

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that exits logical interface xe-0/0/8.0 is mirrored to a remote logical interface with an IP address of 10.9.9.2.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members VXLAN100
```

```
set forwarding-options analyzer test input egress interface xe-0/0/8
set forwarding-options analyzer test output ip-address 10.9.9.2
```

Use Case 3

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that enters logical interface xe-0/0/29.0 is mirrored to a remote logical interface with an IP address of 10.9.9.2.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/29 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/29 unit 0 family ethernet-switching vlan members VXLAN100
set forwarding-options analyzer test input ingress interface xe-0/0/29
set forwarding-options analyzer test output ip-address 10.9.9.2
```

Use Case 4

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that exits logical interface xe-0/0/29.0 is mirrored to a remote logical interface with an IP address of 10.9.9.2.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/29 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/29 unit 0 family ethernet-switching vlan members VXLAN100
set forwarding-options analyzer test input egress interface xe-0/0/29
set forwarding-options analyzer test output ip-address 10.9.9.2
```

Use Case 5

Through the use of the analyzer configuration statement at the [set forwarding-options] hierarchy level, this configuration specifies that Layer 2 traffic that enters VXLAN100, which is mapped to logical interface xe-0/0/8.0, is mirrored to a remote logical interface with an IP address of 10.9.9.2.

```
set vlans VXLAN100 vlan-id 100
set interfaces xe-0/0/8 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/8 unit 0 family ethernet-switching vlan members VXLAN100
set forwarding-options analyzer test input ingress vlan VXLAN100
set forwarding-options analyzer test output ip-address 10.9.9.2
```

Configuring Remote Mirroring with VXLAN Encapsulation

To find out what platforms support this feature as of which releases, see the Feature Explorer page for [Remote port mirroring with VXLAN encapsulation](#).

Analyzer-based configuration

The following sample configuration is for analyzer-based remote mirroring with VXLAN encapsulation. Layer 2 traffic that enters VLAN100 is mirrored to remote output destination VLAN3555, which maps to VNI 1555.

This configuration uses a loopback interface on the destination VLAN to encapsulate the mirrored packets.

- Destination interface xe-0/0/2 is externally connected to loopback interface xe-0/0/3.
- Logical interfaces xe-0/0/2.0 and xe-0/0/3.0 are members of destination VLAN3555.
- Interface xe-0/0/2.0 is configured in enterprise style without any VNI mapping, while interface xe-0/0/3.0 is configured in service provider style with the same vlan ID and with the VNI mapping. This is to prevent flooding or looping between these ports.
- Mac-learning must be disabled on xe-0/0/2.0.



NOTE: The ingress interface must be configured in trunk mode, so the encapsulated packets are tagged. To decapsulate the tagged packets, configure the `set protocols l2-learning decapsulate-accept-inner-vlan` command at the decapsulation node.

```
set vlans VLAN3555 vlan-id 3555
set vlans VLAN3555 vxlan vni 1555
set vlans VLAN3555 interface xe-0/0/3.3555
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members VLAN3555
set interfaces xe-0/0/3 flexible-vlan-tagging
set interfaces xe-0/0/3 encapsulation extended-vlan-bridge
set interfaces xe-0/0/3 unit 3555 vlan-id 3555
set switch-options interface xe-0/0/2 no-mac-learning
set forwarding-options analyzer test input ingress vlan VLAN100
set forwarding-options analyzer test output vlan VLAN3555
```

To configure a logical loopback interface instead of using an external connection, use the following command:

set interfaces *interface-name* ether-options loopback

The sample configuration below uses a logical loopback on interface xe-0/0/2:

```
set vlans VLAN3555 vlan-id 3555
set vlans VLAN3555 vxlan vni 1555
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members VLAN3555
set interfaces xe-0/0/2 ether-options loopback
set switch-options interface xe-0/0/2 no-mac-learning
set forwarding-options analyzer test input ingress vlan VLAN100
set forwarding-options analyzer test output vlan VLAN3555
```

Firewall filter-based configuration

The following configuration applies a firewall filter, filter1, to ingress traffic at interface xe-0/0/34. Traffic ingressing on this interface is mirrored to the destination VLAN3555. The destination VLAN is defined using a port mirroring instance named pm1.

```
set interfaces xe-0/0/34 unit 0 family ethernet-switching filter input filter1
set forwarding-options port-mirroring instance pm1 family ethernet-switching output vlan vlan3555
set firewall family ethernet-switching filter filter1 term to-analyzer then port-mirror-instance pm1
```

Remote Port Mirroring Using VNI Match Conditions

For QFX10002, QFX10008, and QFX10016 Series switches, you can use VXLAN network identifier (VNI) values as a match condition when filtering traffic for remote port mirroring. This ability is often used in network planning and for analytics such as deep packet inspection (DPI).

The remote port mirroring feature acts to create a copy of targeted ingress packets, which are encapsulated in an outer IPv4 GRE header, and then forwarded to designated remote destination. Support for VNI match conditions means you can select packets to be mirrored on the basis of their VNI so that only those flows are directed to the remote mirror port. You can also configure a differentiated service code point (DSCP) value to prioritize the flow, for example, high priority or best-effort delivery. Integrated routing and bridging (IRB) interfaces are not supported as a destination mirror port.

At a high level, the procedure for remote port mirroring based on VNIs is to create a remote port mirroring instance, promote VNI in the firewall filter family to handle VNIs, create the necessary filter rules and actions, and then apply the firewall policy to an ingress interface. GRE tunneling should also be in place on the interface used to send the mirrored packets.

Remote Port Mirroring on the Basis of VNI Use Case: Sample Configuration

The code samples below highlight the key Junos CLI configurations you need to mirror packets according to VNI.

1. Enable remote port mirroring, and also configure the traffic source and the destination to which you want to send the mirrored packets.

```
set forwarding-options port-mirroring remote-port-mirroring
set port-mirroring remote-port-mirroring instance name output ip-source-address IPv4 address
set port-mirroring remote-port-mirroring instance name output ip-destination-address IPv4 address
```

2. Create a firewall filter (here, named *fbf_vni_st*), and promote VNI in this filter to be the packet forwarding module (in other words, this command sets the entire filter to optimize VNI match conditions).

```
set firewall family inet filter fbf_vni_st promote vni
```

3. Create an ingress firewall filter (*fbf_vni_st*), that specifies a VNI match condition (6030 in this sample), and an action (*count*, in this sample).

```
set firewall family inet filter fbf_vni_st term t1-vni from protocol udp
set firewall family inet filter fbf_vni_st term t1-vni from vxlan vni 6030
set firewall family inet filter fbf_vni_st term t1-vni then count t1-vni-6030
set firewall family inet filter fbf_vni_st term default-term then count default-cnt
```

4. Apply the filter to ingress traffic at interface you want to mirror.

```
set interfaces interface unit number family inet filter input fbf_vni_st
```

RELATED DOCUMENTATION

[Understanding Storm Control](#)

[Enabling and Disabling Storm Control \(ELS\)](#)

[Understanding Port Mirroring and Analyzers](#)

[Port Mirroring Constraints and Limitations](#)

GRE over EVPN-VXLAN

SUMMARY

GRE over EVPN-VXLAN enables encapsulation and de-encapsulation of GRE packets within VXLAN tunnels. This protects inner traffic from exposure to intermediate devices during traversal across network segments.

IN THIS SECTION

- [Benefits of GRE over EVPN-VXLAN Support | 705](#)
- [Overview | 706](#)
- [Configuration Example | 706](#)

The integration of Generic Routing Encapsulation (GRE) over EVPN-VXLAN Type 2 tunnels on Junos OS Evolved leverages the forwarding ASIC to encapsulate and de-encapsulate GRE packets. This functionality ensures that GRE frames from remote locations can be efficiently transported through data center networks utilizing EVPN-VXLAN. Support for GRE encapsulation and de-encapsulation, along with configurations such as the "tunnel-loopback" option under Integrated Routing and Bridging (IRB) in the VXLAN bridge domain, enhances the flexibility and performance of complex tunneling setups. Additionally, it includes specific operational parameters and limitations, such as the support for IPv4 underlay only and a maximum throughput of 400 Gbps per DLB (Dynamic Load Balancing) port due to the loopback requirements.

Benefits of GRE over EVPN-VXLAN Support

- Enhances network flexibility by supporting complex tunneling scenarios, accommodating diverse customer use cases and improving overall data traffic management.
- Enhances security by encapsulating GRE traffic within VXLAN tunnels. This provides an additional layer of security by isolating customer traffic from external devices and potential threats.
- Improves traffic handling efficiency by utilizing the forwarding ASIC for GRE encapsulation and de-encapsulation, ensuring high performance in data center network environments.
- Simplifies configuration and implementation with the use of existing EVPN-VXLAN and GRE settings, minimizing the need for new commands or extensive reconfiguration.

Overview

When you implement GRE over EVPN-VXLAN on the Junos OS Evolved platform, you leverage the forwarding ASIC to encapsulate and de-encapsulate GRE packets within VXLAN tunnels. This functionality is essential for ensuring secure and efficient traversal of customer traffic across different network segments. To configure this feature, you will primarily use the `tunnel-loopback` option under IRB interfaces. This configuration allows GRE packets to be looped back for encapsulation within VXLAN, and upon reaching the destination, VXLAN headers are de-encapsulated first, followed by the GRE headers. This ensures that the internal traffic remains hidden from intermediate devices, maintaining its integrity and preventing double-billing issues for ISP customers.

To configure GRE over EVPN-VXLAN, you will use existing Junos OS Evolved CLI commands for EVPN-VXLAN and GRE, ensuring a streamlined configuration process. Key configuration steps include setting up the IRB interfaces with the `tunnel-loopback` statement. For example, you would use commands like `set interfaces irb unit name family inet tunnel-loopback` to enable the `tunnel-loopback` functionality. Additionally, you need to configure the appropriate routing instances and protocol settings to ensure proper encapsulation and de-encapsulation of traffic. This setup is critical for maintaining the operational efficiency and security of network traffic across multiple segments.

While this feature offers significant benefits, it is crucial to understand its limitations. GRE over EVPN-VXLAN requires a loopback pass for both encapsulation and de-encapsulation, which limits the overall throughput per DLB port. Additionally, it does not support VXLAN Type 5 tunnels, `bypass-loopback` options configured under the flexible tunnel interface for GRE, or filter-based de-encapsulation for GRE. GRE over EVPN-VXLAN support is also limited to an IPv4 underlay only. Despite these limitations, this feature enhances the flexibility and scalability of network designs, ensuring secure and efficient traffic traversal.

Please refer to [Feature Explorer](#) for a complete list of the products that support this feature.

Configuration Example

This example illustrates an ISP using GRE over VXLAN.

```
set interfaces et-0/0/0 description TO_QFX5K_VXLAN;
set interfaces et-0/0/0 speed 40g;
set interfaces et-0/0/0 ether-options 802.3ad ae1;

set interfaces et-0/0/4 description TO_MX_GRE;
set interfaces et-0/0/4 flexible-vlan-tagging;
set interfaces et-0/0/4 encapsulation flexible-ethernet-services;
set interfaces et-0/0/4 unit 100 vlan-id 100;
set interfaces et-0/0/4 unit 100 family inet address 10.100.0.1/24;
```

```

set interfaces ae1 mtu 9192;
set interfaces ae1 aggregated-ether-options lacp active;
set interfaces ae1 aggregated-ether-options lacp periodic fast;
set interfaces ae1 unit 0 family inet address 10.0.0.2/24;

set interfaces fti0 unit 1 tunnel encapsulation gre source address 172.16.128.1;
set interfaces fti0 unit 1 tunnel encapsulation gre destination address 172.16.128.128;
set interfaces fti0 unit 1 family inet address 10.255.57.0/31;

set interfaces irb mtu 9000;
set interfaces irb unit 1024 family inet tunnel-loopback;
set interfaces irb unit 1024 family inet address 172.16.129.77/31;
set interfaces irb unit 1088 family inet address 172.16.129.76/31;

set interfaces lo0 unit 0 family inet address 172.16.128.1/32 primary;
set interfaces lo0 unit 0 family inet address 192.168.200.100/32;

set forwarding-options tunnel-termination;

set routing-instances isp instance-type virtual-router;
set routing-instances isp routing-options autonomous-system 65001;
set routing-instances isp protocols bgp group ext type external;
set routing-instances isp protocols bgp group ext neighbor 10.100.0.2 peer-as 65002;
set routing-instances isp protocols bgp group int type internal;
set routing-instances isp protocols bgp group int local-address 172.16.129.76;
set routing-instances isp protocols bgp group int neighbor 172.16.129.77 family inet unicast;
set routing-instances isp protocols ospf area 0.0.0.0 interface irb.1088;
set routing-instances isp interface et-0/0/4.100;
set routing-instances isp interface irb.1088;

set routing-instances l2-vrf instance-type mac-vrf;
set routing-instances l2-vrf protocols evpn encapsulation vxlan;
set routing-instances l2-vrf vtep-source-interface lo0.0;
set routing-instances l2-vrf service-type vlan-aware;
set routing-instances l2-vrf route-distinguisher 172.16.128.1:2;
set routing-instances l2-vrf vrf-target target:64657:1;
set routing-instances l2-vrf vlans vlan-1024 vlan-id 1024;
set routing-instances l2-vrf vlans vlan-1024 l3-interface irb.1024;
set routing-instances l2-vrf vlans vlan-1024 vxlan vni 101024;
set routing-instances l2-vrf vlans vlan-1088 vlan-id 1088;
set routing-instances l2-vrf vlans vlan-1088 l3-interface irb.1088;
set routing-instances l2-vrf vlans vlan-1088 vxlan vni 101088;

```

```

set routing-options router-id 172.16.128.1;
set routing-options autonomous-system 65001;

set protocols bgp group evpn type internal;
set protocols bgp group evpn local-address 172.16.128.1;
set protocols bgp group evpn family evpn signaling;
set protocols bgp group evpn neighbor 172.16.128.11;

set protocols bgp group isp1 type internal;
set protocols bgp group isp1 local-address 172.16.129.77;
set protocols bgp group isp1 tcp-mss 1400;
set protocols bgp group isp1 neighbor 172.16.129.76 family inet unicast;

set protocols bgp group BGP-GRE type external;
set protocols bgp group BGP-GRE peer-as 65534;
set protocols bgp group BGP-GRE local-as 65001;
set protocols bgp group BGP-GRE neighbor 10.255.57.1;

set protocols ospf area 0.0.0.0 interface lo0.0 passive;
set protocols ospf area 0.0.0.0 interface ae1.0 interface-type p2p;
set protocols ospf area 0.0.0.0 interface irb.1024;

```

RELATED DOCUMENTATION

[Understanding VXLANs](#) | 606

DHCP Smart Relay in EVPN-VXLAN

DHCP smart relay provides redundancy and resiliency to DHCP relay by allowing the relay agent to use multiple IP addresses as the gateway IP address. When forwarding DHCP client requests to a DHCP server, the relay agent initially uses the primary IP address as the gateway IP address. If the DHCP server does not respond with an offer message after three attempts, the relay agent repeats the process using an alternate IP address as the gateway IP address.

To use DHCP smart relay, you must have at least two IP addresses configured on the relay agent interface. If you configure more than two IP addresses on the relay agent interface, smart relay tries each IP address until an offer message is received.

DHCP smart relay is supported for EVPN over VXLAN in centrally routed bridging and edge-routed bridging modes. EVPN is usually configured with DHCP stateless relay to support multihoming, but

DHCP smart relay is supported only for stateful relay. To support multihoming in an EVPN over VXLAN deployment with DHCP stateful relay, use the following configuration:

```
set forwarding-options dhcp-relay forward-snooped-clients all-interfaces
set forwarding-options dhcp-relay overrides allow-snooped-clients
set forwarding-options dhcp-relay route-suppression access-internal
```

To enable smart relay on all interfaces that are relay agents, configure the following statement:

```
set forwarding-options dhcp-relay overrides apply-secondary-as-giaddr
```

To enable smart relay on specific interfaces that are relay agents, configure the following statement:

```
set forwarding-options dhcp-relay group group-name interface interface-name overrides apply-
secondary-as-giaddr
```

Configuring VLAN-Aware Bundle Services, VLAN-Based Services, and Virtual Switch Support

IN THIS CHAPTER

- [Understanding VLAN-Aware Bundle and VLAN-Based Service for EVPN | 710](#)
- [Configuring EVPN with VLAN-Based Service | 711](#)
- [Configuring EVPN VLAN-Aware Bundle Service | 717](#)
- [Virtual Switch Support for EVPN Overview | 718](#)
- [Configuring EVPN with Support for Virtual Switch | 720](#)

Understanding VLAN-Aware Bundle and VLAN-Based Service for EVPN

IN THIS SECTION

- [VLAN-Aware Bundle Service | 711](#)
- [VLAN-Based Service | 711](#)

A Data Center Service Provider (DCSP) hosts the data center for its multiple customers on a common physical network. To each customer (also called a tenant), the service looks like a full-fledged data center that can expand to 4094 VLANs and all private subnets. For disaster recovery, high availability, and optimization of resource utilization, it is common for the DCSP to span across the data center to more than one site. When deploying the data center services, a DCSP faces the following main challenges:

- Extending Layer 2 domains across more than one data center site. This requires optimal intra-subnet traffic forwarding.
- Supporting optimal inter-subnet traffic forwarding and optimal routing in the event of virtual machine (VM).

- Supporting multiple tenants with independent VLAN and subnet space.

The DCSP might require Ethernet VLAN services to be extended over a WAN with a single EVPN instance (EVI). On the QFX Series, each EVPN and virtual switch routing instance corresponds to an EVI. The QFX Series supports VLAN-aware bundle service and VLAN-based service, which maintain data and control plane separation.



NOTE: If you create VLANs that are not part of a routing instance, they become part of the Default Switch routing instance.

VLAN-Aware Bundle Service

VLAN-aware bundle services supports the mapping of one or more routing instances of type Virtual Switch to many VLAN IDs (VIDs) and multiple bridge tables, with each bridge table corresponding to a different VLAN. To enable VLAN-aware bundle service, configure a Virtual Switch routing instance. For service provider-related applications, where the VLAN ID is local to the Layer 2 logic interface, enable the `flexible-vlan-tagging` statement in your configuration. For enterprise-related applications, where the VLAN ID has global significance, enable the `family ethernet-switching` statement in your configuration. VLAN-aware bundle service supports up to 4000 VLANs per routing instance.

VLAN-Based Service

VLAN-based service supports the mapping of one routing instance of type EVPN to one VLAN. There is only one bridge table that corresponds to the one VLAN. If the VLAN consists of multiple VLAN IDs (VIDs)—for example, there is a different VID per Ethernet segment on a provider edge device—then VLAN translation is required for packets that are destined to the Ethernet segment. To enable VLAN-based service, configure an EVPN routing instance. Up to 100 EVPN routing instances are supported.

Configuring EVPN with VLAN-Based Service

VLAN-based service supports the mapping of one or more routing instances of type EVPN to only one VLAN. There is only one bridge table that corresponds to the one VLAN. If the VLAN consists of multiple VLAN IDs (VIDs)—for example, there is a different VID per Ethernet segment on a provider edge device—then VLAN translation is required for packets that are destined to the Ethernet segment.

To configure VLAN-based service and Layer 3 routing with two EVPN routing instances on a provider edge device.

1. Configure the first routing instance of type `evpn` named `evpn1`.

For example:

```
[edit]
user@switch# set routing-instances evpn1 instance-type
evpn
```

2. Configure the access interface for handling EVPN traffic.

For example:

```
[edit]
user@switch# set routing-instances evpn1 interface
xe-0/0/8.100
```

3. Configure a Layer 3 integrated and routing (IRB) interface for the evpn1 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn1 l3-interface
irb.100
```

4. Configure the Virtual Tunnel Endpoint interface for the evpn1 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn1 vtep-source-interface
lo0.0
```

5. Configure a VLAN identifier for the evpn1 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn1 vlan-id
none
```

6. Configure a route distinguisher for the evpn1 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn1 route-distinguisher
1.2.3.11:1
```

7. Configure the VPN routing and forwarding (VRF) target community for the evpn1 routing instance.
For example:

```
[edit]
user@switch# set routing-instances evpn1 vrf-target
target:1234:11
```

8. Configure the encapsulation type for the evpn1 routing instance.
For example:

```
[edit]
user@switch# set routing-instances evpn1 protocols
evpn encapsulation vxlan
```

9. Configure the VXLAN Network Identifier (VNI) for the evpn1 routing instance.
For example:

```
[edit]
user@switch# set routing-instances evpn1 vxlan
vni 100
```

10. Configure the second routing instance of type evpn named evpn2:
For example:

```
[edit]
user@switch# set routing-instances evpn2 instance-type
evpn
```

11. Configure the access interface on the provider edge device (PE) for handling EVPN traffic.

For example:

```
[edit]
user@switch# set routing-instances evpn2 interface
xe-0/0/8.200
```

12. Configure a Layer 3 integrated and routing (IRB) interface for the evpn2 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn2 l3-interface
irb.200
```

13. Configure the loopback address as the virtual tunnel endpoint source interface for the evpn2 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn2 vtep-source-interface
lo0.0
```

14. Configure the VLAN identifier for the evpn2 routing instance to be none.

For example:

```
[edit]
user@switch# set routing-instances evpn2 vlan-id
none
```

15. Configure a route distinguisher for the evpn2 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn2 route-distinguisher
1.2.3.11:2
```

16. Configure the VPN routing and forwarding (VRF) target community for the evpn2 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn2 vrf-target
target:1234:24
```

17. Configure the encapsulation type for the evpn2 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn2 protocols
evpn encapsulation vxlan
```

18. Configure the VXLAN Network Identifier (VNI) for the evpn2 routing instance.

For example:

```
[edit]
user@switch# set routing-instances evpn2 vxlan
vni 200
```

19. Configure a VPN routing and forwarding (VRF) routing instance.

For example:

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

20. Configure the first of two integrated routing and bridging (IRB) interface for the vrf instance.

For example:

```
[edit]
user@switch# set routing-instances vrf interface
irb.100
```

21. Configure the second of two integrated routing and bridging (IRB) interface for the VPN routing and forwarding (VRF) instance.

For example:

```
[edit]
user@switch# set routing-instances vrf interface
  irb.200
```

22. Configure the loopback interface for the vrf instance.

For example:

```
[edit]
user@switch# set routing-instances vrf interface
  lo0.1000
```

23. Configure a unique route distinguisher for the VPN routing and forwarding (VRF) instance to identify from which EVPN the route belongs.

For example:

```
[edit]
user@switch# set routing-instances vrf route-distinguisher
  1.2.3.1:2
```

24. Configure the VPN routing and forwarding (VRF) target community for the VPN routing and forwarding (VRF) routing instance.

For example:

```
[edit]
user@switch# set routing-instances vrf vrf-target
  target:2222:22
```

25. Configure the auto-export option to automatically derive the route target.

For example:

```
[edit]
user@switch# set routing-instances vrf routing-options
  auto-export
```

Configuring EVPN VLAN-Aware Bundle Service

An Ethernet VPN (EVPN) VLAN-aware bundle service configured within an EVPN instance (EVI) provides a many-to-one mapping of VLAN IDs. The MAC-VRF instance corresponding to the EVI maintains a unique bridge table for each VLAN ID, and unique labels for each bridge domain.

Configure a Layer 2 interface with VLAN-aware bundle service support.

1. Configure the interface with flexible-vlan-tagging.

```
set interfaces interface flexible-vlan-tagging
```

2. Configure the interface with flexible-ethernet-services encapsulation.

```
set interfaces interface encapsulation flexible-ethernet-services
```

3. Configure the interface to support the first VLAN.

```
set interfaces interface unit unit 1 encapsulation vlan-bridge
set interfaces interface unit unit 1 vlan-id vlan-id 1
```

4. Configure the interface to support an additional VLAN.

```
set interfaces interface unit unit 2 encapsulation vlan-bridge
set interfaces interface unit unit 2 vlan-id vlan-id 2
```

5. Configure a MAC-VRF routing-instance.

```
set routing-instances instance-name instance-type mac-vrf
```

6. Configure the EVPN protocol with VXLAN encapsulation.

```
set routing-instances instance-name protocols evpn encapsulation vxlan
```

7. Configure the vlan-aware service-type.

```
set routing-instances instance-name service-type vlan-aware
```

8. Configure a route-distinguisher.

```
set routing-instances instance-name route-distinguisher x:x
```

9. Configure a vrf-target.

```
set routing-instances instance-name vrf-target target:x:x
```

10. Configure VLANs within the routing-instance.

```
set routing-instances instance-name vlans vlan-1 vlan-id vlan-id-1
set routing-instances instance-name vlans vlan-1 interface interface.vlan-id-1
set routing-instances instance-name vlans vlan-2 vlan-id vlan-id-2
set routing-instances instance-name vlans vlan-2 interface interface.vlan-id-2
```



NOTE: You must configure set protocols evpn esi-resolution-per-bridge-domain on a Junos OS based device to interoperate with an ACX device running Junos OS Evolved, when both are configured with VLAN-aware bundle service. This statement is the default behavior on ACX devices.



NOTE: Junos OS based devices support either the label-allocation per-instance model, or the label-allocation per-bridge-domain model. ACX devices only support label-allocation per-bridge-domain. See *label-allocation* for more details on each model. label-allocation-per-bridge-domain is the default behavior on ACX devices and does not need explicit configuration.

RELATED DOCUMENTATION

| [*esi-resolution-per-bridge-domain*](#)

Virtual Switch Support for EVPN Overview

Starting with Junos OS Release 14.1, VLAN-aware bundle service is introduced on MX Series routers. Starting with Junos OS Release 14.2, VLAN-aware bundle service is introduced on EX9200 switches.

Starting with Junos OS Release 17.3, VLAN-aware bundle service is introduced on QFX Series switches. This feature allows Ethernet VLANs over a WAN to share a single EVPN instance while maintaining data-plane separation between the different VLANs.

Junos OS has a highly flexible and scalable virtual switch interface. With a virtual switch, a single router or switch can be divided into multiple logical switches. Layer 2 domains (also called *bridge-domains* or *vlan*s) can be defined independently in each virtual switch. To configure VLAN-aware bundle service, an EVPN must run in a virtual-switch routing instance.

On the EX Series and MX Series, a single EVPN instance can stretch up to 4094 bridge domains or VLANs defined in a virtual switch to remote sites. A virtual switch can have more than 4094 bridge domains or VLANs with a combination of none, single, and dual VLANs. However, because EVPN signaling deals only with single VLAN tags, a maximum of 4094 bridge domains or VLANs can be stretched. The EVPN virtual switch also provides support for trunk and access interfaces.

Starting with Junos OS Release 17.3R1, on the QFX10000 line of switches, you can configure both EVPN and virtual-switch routing instances. The EVPN routing instance supports VLAN-based service. With VLAN-based service, the EVPN instance includes only a single broadcast domain, and there is a one-to-one mapping between a VNI and MAC-VRF. Up to 100 EVPN routing instances are supported. The virtual-switch routing instance supports VLAN-aware service, and up to 10 virtual-switch routing instances with 2000 VLANs are supported.

If you create VLANs that are not part of a routing instance, they become part of the default switch routing instance.



NOTE:

- The `none` VLAN option is supported with bridge domains or VLANs under the virtual switch instance type for EVPNs.
- Access interfaces configured with single or dual VLAN tags are supported in EVPN. By default, only Ethernet frames with single or no VLAN tags are transported across the EVPN core. As a result, dual-tagged Ethernet frames received on the access interfaces must be normalized to Ethernet frames with single or no VLAN tags for proper transmission over the EVPN core.

You can enable transporting of dual-tagged frames across the EVPN core network by including both the `vlan-id none` and `no-normalization` configuration statements together.

- Dual tagged trunk interfaces are not supported in an EVPN virtual switch instance type for EVPN-MPLS.

There are two types of VLAN-aware bundle service:

- VLAN-aware bundle without translation

The service interface provides bundling of customer VLANs into a single Layer 2 VPN service instance with a guarantee end-to-end customer VLAN transparency. The data-plane separation between the customer VLANs is maintained by creating a dedicated bridge-domain for each VLAN.

- VLAN-aware bundle with translation

The service interface provides bundling of customer VLANs into a single Layer 2 VPN service instance. The data-plane separation between the customer VLANs is maintained by creating a dedicated bridge-domain for each VLAN. The service interface supports customer VLAN translation to handle the scenario where different VLAN Identifiers (VIDs) are used on different interfaces to designate the same customer VLAN.

EVPN with virtual switch provides support for VLAN-aware bundle with translation only.

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
17.3R1	Starting with Junos OS Release 17.3R1, on the QFX10000 line of switches, you can configure both EVPN and virtual-switch routing instances.
17.3	Starting with Junos OS Release 17.3, VLAN-aware bundle service is introduced on QFX Series switches.
14.2	Starting with Junos OS Release 14.2, VLAN-aware bundle service is introduced on EX9200 switches.
14.1	Starting with Junos OS Release 14.1, VLAN-aware bundle service is introduced on MX Series routers.

RELATED DOCUMENTATION

| [Example: Configuring EVPN with Support for Virtual Switch](#) | 1482

Configuring EVPN with Support for Virtual Switch

You can configure an Ethernet VPN (EVPN) with virtual switch support to enable multiple tenants with independent VLAN and subnet space within an EVPN instance. Virtual switch provides the ability to extend Ethernet VLANs over a WAN using a single EVPN instance while maintaining data-plane separation between the various VLANs associated with that instance. A single EVPN instance can stretch up to 4094 bridge domains defined in a virtual switch to remote sites.

When configuring virtual switch for EVPN, be aware of the following considerations:

- Due to default ARP policing, some of the ARP packets not destined for the device can be missed. This can lead to delayed ARP learning and synchronization.
- Clearing ARP for an EVPN can lead to inconsistency between the ARP table and the EVPN ARP table. To avoid this situation, clear both ARP and EVPN ARP tables.
- The `vlan-tag` can be configured for local switching. However, vlan-tagged VLANs should not be extended over the EVPN cloud.

This task explains how to configure one Virtual Switch instance that includes one VLAN.

1. Configure the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance instance-type virtual-switch
```

2. Configure the interface names for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance interface interface-name
```

3. Configure the route distinguisher for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance route-distinguisher route-distinguisher-value
```

4. Configure the VPN routing and forwarding (VRF) target community for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vrf-target vrf-target
```

5. List the VLAN identifiers that are to be EVPN extended.

```
[edit routing-instances]
user@PE1# set evpn-instance protocols evpn extended-vni-list [vlan-id-range]
```

6. Configure the VLAN and VLAN ID for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vlans name of VLAN vlan-id VLAN ID number
```

7. Configure VXLAN encapsulation and Virtual Network Identifier for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vxlan vni VNI number
```

8. Configure the virtual tunnel endpoint source interface for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vtep-source-interface interface-name
```

9. Verify and commit the configuration.

```
[edit]
user@PE1# commit
commit complete
```

RELATED DOCUMENTATION

| [Example: Configuring EVPN with Support for Virtual Switch](#) | 1482

Load Balancing with EVPN-VXLAN Multihoming

IN THIS CHAPTER

- [Dynamic Load Balancing in an EVPN-VXLAN Network | 723](#)
- [Fast Reroute for Egress Link Protection with EVPN-VXLAN Multihoming | 727](#)

Dynamic Load Balancing in an EVPN-VXLAN Network

IN THIS SECTION

- [Benefits of Dynamic Load Balancing in an EVPN-VXLAN Network | 724](#)
- [How Dynamic Load Balancing Works | 724](#)
- [How Traffic Is Balanced | 725](#)
- [How to Verify That Dynamic Load Balancing Is Enabled | 726](#)

When your EVPN-VXLAN network includes a multihomed device that can be reached through multiple VTEPs that share a common Ethernet segment identifier (ESI), dynamic load balancing works as follows:

- The EVPN control plane (overlay) identifies the common ESI as the next hop for a destination device with a particular MAC address.
- Based on the parameters in a packet, the forwarding plane in a Juniper Networks switch (hardware) dynamically chooses one of the VTEPs associated with the ESI. The VTEP then forwards the packet along the selected underlay path to the destination device.

By default, Juniper Networks switches have dynamic load balancing enabled. So, you don't need to configure the feature to get it up and running in an EVPN-VXLAN network.

Instead of statically assigning one virtual tunnel endpoint (VTEP) to forward traffic to a destination device in an EVPN-VXLAN network, Juniper Networks switches now support dynamic load balancing.

Benefits of Dynamic Load Balancing in an EVPN-VXLAN Network

- More efficient use of aggregated Ethernet links that share a common ESI.
- Better bandwidth utilization throughout an EVPN-VXLAN network.

How Dynamic Load Balancing Works

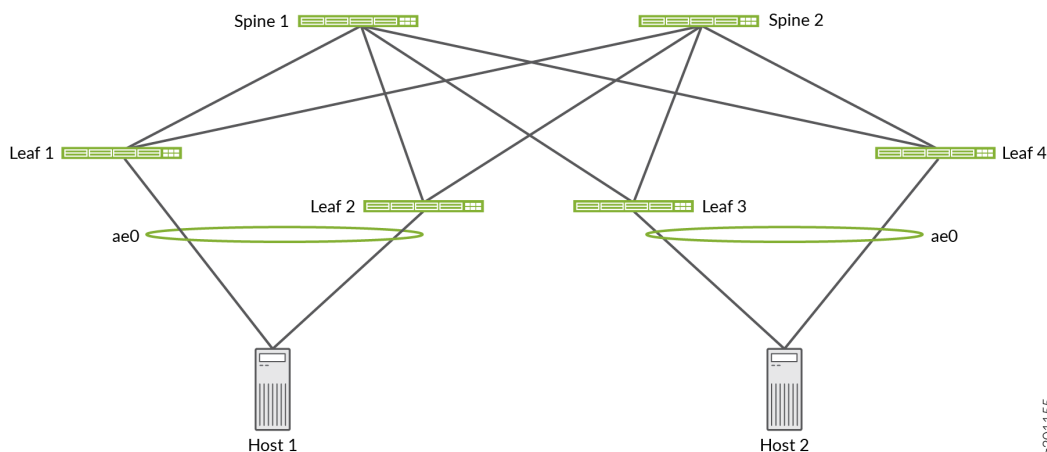
Figure 61 on page 724 shows a sample EVPN-VXLAN network in which we support dynamic load balancing. This network includes the following elements:

- Multihomed Hosts 1 and 2. Each of the hosts is connected to two leaf devices through an aggregated Ethernet LAG that is assigned a common ESI.
- Multihomed Leafs 1 through 4. Each of the leaf devices is connected to Spines 1 and 2.



NOTE: To keep things simple, the sample EVPN-VXLAN network in Figure 61 on page 724 shows that the leaf devices are multihomed to two spine devices. However, we support dynamic load balancing among more than two spine devices.

Figure 61: EVPN-VXLAN Network with Dynamic Load Balancing

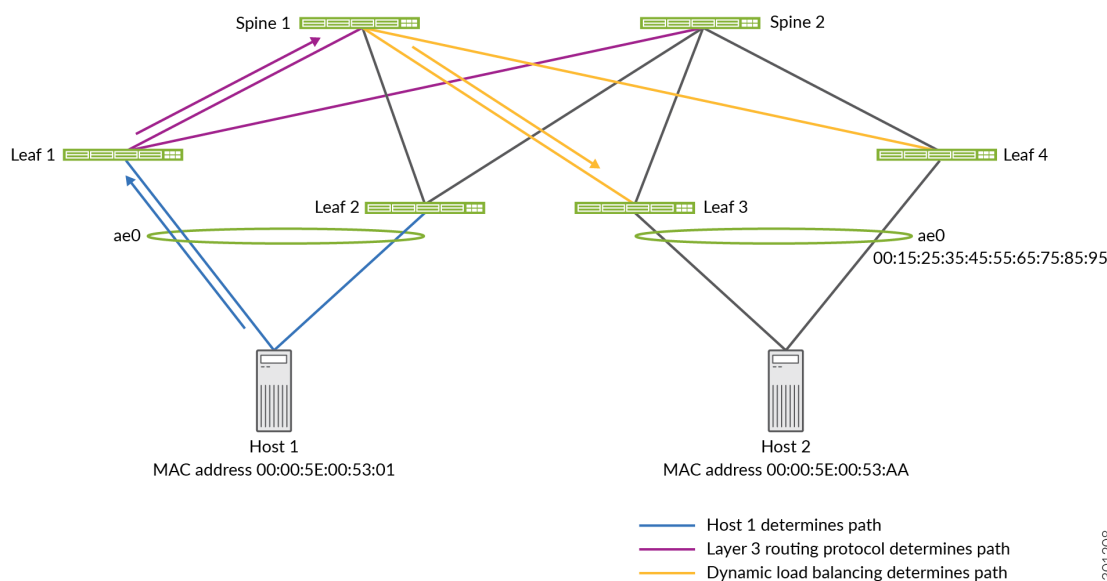


In this EVPN-VXLAN network, the leaf devices perform dynamic load balancing. To understand how dynamic load balancing works, here's what happens when Host 1 sends a packet to Host 2. In addition

to the following dynamic load balancing explanation, [Figure 62 on page 725](#) provides a graphical summary of the path options and the choices made.

- Host 1 must choose one of the aggregated Ethernet interfaces through which to forward the packet. In this case, Host 1 chooses the interface to Leaf 1.
- Upon receipt of the packet, Leaf 1 identifies Host 2's destination MAC address 00:00:5E:00:53:AA as a member of remote ESI 00:15:25:35:45:55:65:75:85:95. This ESI is assigned to aggregated Ethernet interface ae0, to which Leafs 3 and 4 are connected.
- Leaf 1 can choose either Leaf 3 or 4 as the intermediate Layer 2 EVPN-VXLAN next hop to which to forward the packet. Using packet parameters established by the dynamic load balancing feature, Leaf 1 dynamically chooses Leaf 3.
- Leaf 1 can choose either Spine 1 or 2 as the next hop to reach Leaf 3. Using Layer 3 routing tables and routes programmed into the switch hardware, Leaf 1 chooses Spine 1.

Figure 62: Summary of Path Options and Choices



How Traffic Is Balanced

Juniper Networks switches use a hash of the following packet parameters to dynamically select the next-hop VTEP:

- Packets with an IP header:

- IP header fields:
 - Source IP address
 - Destination IP address
 - Protocol
- VLAN ID
- Layer 4 (TCP and UDP) source and destination ports
- Packets with an MPLS/IP header:
 - Up to three top labels
 - IP header fields:
 - Source IP address
 - Destination IP address
 - Layer 4 (TCP and UDP) source and destination ports
- Packets with a Layer 2 header only:
 - Source MAC address
 - Destination MAC address
 - VLAN ID

The hashing takes place before a packet undergoes VXLAN encapsulation.

To refine the hashing input used by dynamic load balancing, you can include the [enhanced-hash-key hash-parameters](#) `ecmp` configuration statements at the `[edit forwarding-options]` hierarchy level.

How to Verify That Dynamic Load Balancing Is Enabled

You can verify that dynamic load balancing is enabled by entering the following command:

```
user@switch> show ethernet-switching global-information
Global Configuration:

MAC aging interval      : 300
...
LE  VLAN aging time     : 1200
RE  state                : Master
```

```
VXLAN Overlay load bal: Enabled
VXLAN ECMP           : Enabled
```

In the output that appears, check the VXLAN Overlay load bal field to make sure that it is set to Enabled.

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
20.3R1	Starting with Junos OS Release 20.3R1, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, and QFX5220 switches support dynamic load balancing in an EVPN-VXLAN network.

RELATED DOCUMENTATION

show ethernet-switching table

show ethernet-switching vxlan-tunnel-end-point svlbnh

Fast Reroute for Egress Link Protection with EVPN-VXLAN Multihoming

SUMMARY

You can enable the fast reroute egress link protection (ELP) feature on multihoming peer provider edge (PE) devices in an EVPN-VXLAN network. This feature improves route convergence times and avoids load-balanced traffic loss toward a multihomed customer edge (CE) device if a PE device link to the CE device fails.

IN THIS SECTION

- [Benefits | 727](#)
- [How Fast Reroute ELP Works with Multihoming in an Ethernet Segment | 728](#)
- [Configure Fast Reroute ELP on a PE Device | 732](#)
- [Verify Fast Reroute ELP Tunnel Creation and Operation | 734](#)

Benefits

- Avoids loss of load-balanced traffic toward a multihomed CE device when a link to the CE device fails.

- Improves route convergence time for multihoming Ethernet segment routes in higher scaled EVPN-VXLAN fabrics.

How Fast Reroute ELP Works with Multihoming in an Ethernet Segment

IN THIS SECTION

- [ELP with Fast Reroute in an Ethernet Segment | 729](#)
- [ELP Tunnels with Multiple Peer PE Devices | 731](#)
- [Fast Reroute ELP Requirements and Limitations | 732](#)

In EVPN fabrics, to improve reliability and performance, customers often multihome CE devices to two or more PE devices in an Ethernet segment (ES). Multihoming helps to:

- Safeguard against traffic loss if a PE device link to the multihomed CE device fails.
- Enable load balancing among multiple paths to the destination.

The connections for a multihomed CE to different PE devices share an ES identifier (ESI). EVPN devices use ES routes (EVPN Type 4 routes) to advertise their local multihoming peer PE connections. The PE devices also use the ES routes to elect designated forwarders (DFs) among the peer devices to avoid sending duplicate traffic on the ESI.

With EVPN-VXLAN, a PE device connected to a multihomed CE device advertises ES tunnel routes (VXLAN tunnel routes) for each locally attached ESI. Remote PE devices receive these advertisements and create load-balancing next-hop tables for the ES. The remote PE devices use the ES next hops to load-balance traffic that is destined for a multihomed CE device among the peer PE devices associated with the ESI.

When one of the ESI links fails while a remote PE is sending traffic to the multihomed CE device:

1. The PE device with the failed link withdraws the corresponding ES route.
2. A remote PE device continues load balancing traffic toward the ESI using routes that correspond to the failed link until the device receives the route withdrawal and adjusts its load-balancing next hops.
3. The PE device with the failed link drops the excess (residual) load-balanced traffic destined for the failed link.
4. The remote PE devices receive and process the route withdrawal. The remote PE devices update their MAC address routing information and next-hop tables. At that point, the remote PE devices stop sending load-balanced traffic to the failed link.

Until the remote PE routing tables converge (which might take seconds in scaled environments), the destination CE device loses some of the traffic.

To help avoid this load-balanced traffic loss on the ESI when a member link goes down, you can enable the fast reroute ELP feature on the multihoming peer PE devices. This feature also helps improve ESI route convergence time, especially in highly-scaled networks.

ELP with Fast Reroute in an Ethernet Segment

When you enable this feature on a PE device, the device creates a backup VXLAN tunnel called an ELP tunnel to each of its peer PE devices for an ES. Upon ES link failure, the PE device uses the ELP tunnel to reroute traffic to one or more of its peer PE devices.



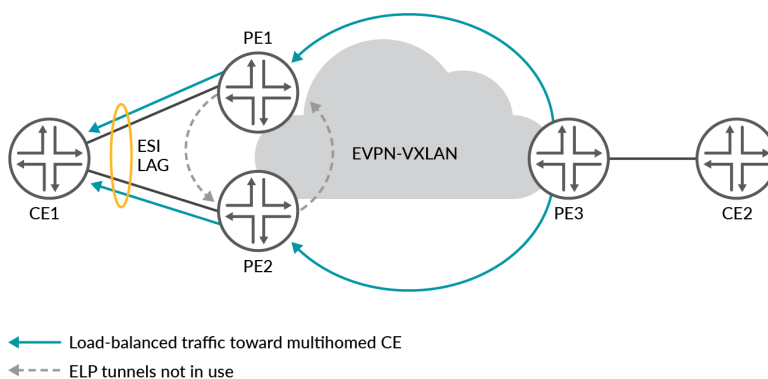
NOTE: For this feature to work, you enable it uniformly on all peer PE devices associated with an ESI. Each PE device on the ESI has an ELP tunnel to all of the peer PE devices.

The following figures illustrate how the ELP tunnel reroute process works in a simple EVPN-VXLAN fabric where:

- CE1 is multihomed to peer PE devices PE1 and PE2.
- PE1 and PE2 have ELP tunnels to each other.
- PE3 is a remote source that load-balances traffic toward CE1 using VXLAN tunnel endpoints (VTEPs) on PE1 and PE2.

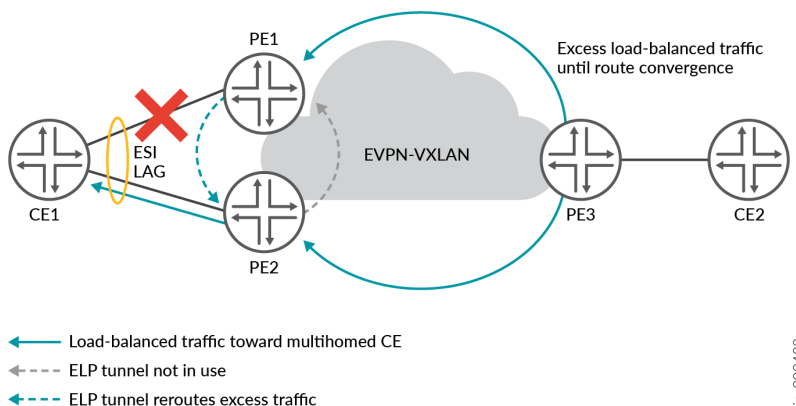
In [Figure 63 on page 729](#), all the ES links toward CE1 are up, so PE1 and PE2 each forward their portions of the load-balanced traffic to CE1. PE1 and PE2 don't need to use their ELP tunnels.

Figure 63: Load-Balanced Traffic to Multihomed CE with All ESI Links Up



In [Figure 64 on page 730](#), the link from PE1 toward CE1 fails. PE1 withdraws the ES route for the failed link. However, until PE3 achieves route convergence, PE3 still sends some excess load-balanced traffic to PE1. Instead of dropping its portion of the load-balanced traffic, PE 1 uses its ELP tunnel to reroute that traffic to PE2. PE2 forwards both portions of the load-balanced traffic to CE1.

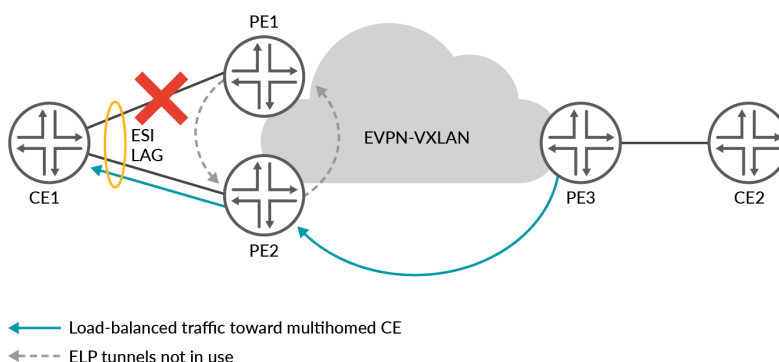
Figure 64: ES Link Down with Fast Reroute to Peer PE



jnr-000428

In [Figure 65 on page 730](#), PE3 achieves route convergence and stops sending any load-balanced traffic for the ES to PE1. PE3 sends all traffic for the ES only to PE2, and PE2 sends the traffic to CE1. PE1 doesn't need to use its ELP tunnel to PE2 anymore.

Figure 65: ELP Tunnel Not Used After ES Route Convergence



jnr-000429

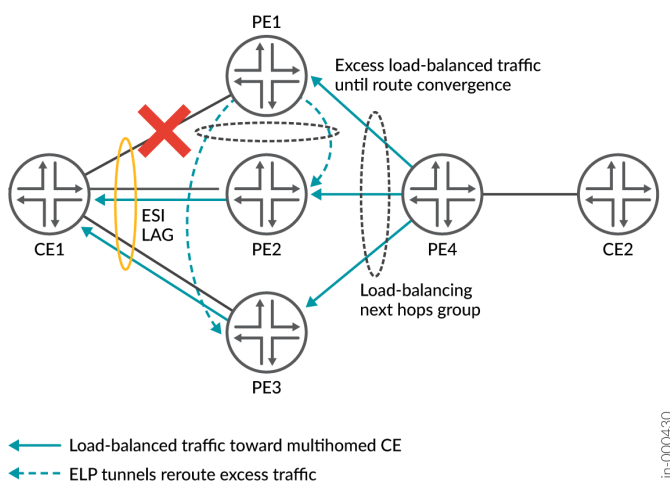
ELP Tunnels with Multiple Peer PE Devices

When you enable this feature on a PE device with two or more peer PE devices on the ES, the PE device sets up a backup VXLAN ELP tunnel to each peer PE device. See [Figure 66 on page 731](#). Each PE device maintains a next hops group for load balancing traffic destined for the ES. The remote source PE device distributes the traffic toward the ES among the next hops in the group. Each peer PE device on the ES also maintain a load-balancing next hops group with all of the ELP tunnels for the ES, so the device can also load balance rerouted traffic.

[Figure 66 on page 731](#) shows what happens if one of the links on the ES fails. In this case, the link from PE1 to CE1 fails. As a result:

- Until PE4 achieves route convergence, PE4 still sends some excess load-balanced traffic to PE1.
- Instead of dropping its portion of the load-balanced traffic, PE1 reroutes its excess load-balanced traffic on both of its ELP tunnels toward peer PE devices PE2 and PE3.
- PE2 and PE3 send all of the traffic to CE1, including the load-balanced traffic they receive from PE4 and the traffic from PE1's ELP tunnels.
- (Not shown) PE4's routing tables converge. PE4 stops sending any traffic toward PE1 and load-balances the traffic only to PE2 and PE3.

Figure 66: ELP Tunnels for Two or More Peer PE Devices



Fast Reroute ELP Requirements and Limitations

To enable an ELP tunnel on a PE device, configure the *reroute-address* statement at the [edit forwarding-options evpn-vxlan] hierarchy level. Specify the IP address the device uses for the ELP tunnel using the *inet ip-address* option to the *reroute-address* statement. ["Configure Fast Reroute ELP on a PE Device" on page 732](#) describes more about this statement and the other steps required to configure this feature.

With this feature:

- You must configure this statement on all of the peer PE devices for a particular ES for the ELP tunnels on any peer PE device to work.
- Multihoming peer PE devices do not flush MAC address table entries for aggregated Ethernet (AE) interface bundle links in active/active state when an AE ESI interface goes down. This behavior helps implement the fast reroute actions and improves route convergence times.

Fast reroute ELP has the following limitations:

- This feature applies only with multihoming configured on AE logical interfaces.
- We support this feature only for IPv4 client traffic.
- This feature works for unicast traffic and VLAN flooding only, not multicast traffic.
- We support this feature only with enterprise style interface configurations. (See *Flexible Ethernet Services Encapsulation* for more on this type of interface configuration.)
- This feature handles rerouting traffic upon ES link failure on only one multihoming peer at a time that is associated with a particular ESI.



NOTE: Fast reroute ELP is not supported for planned administrative link-down events such as when you administratively disable a PE-CE interface locally on the PE device by running the `set interface interface-name disable` command. For a planned interface-down maintenance event on the PE device, if needed, take other administrative actions to prevent traffic loss during the maintenance. However, if a PE-CE link goes down due to an administrative link-down command on the CE device, the PE device detects the link failure and invokes the fast reroute operation in that case.

Configure Fast Reroute ELP on a PE Device

You configure fast reroute ELP feature elements on supported multihoming peer PE devices for an ES. For this feature to work, you must enable it on all of the multihoming peer PE devices for the ES. You configure the steps that are listed here uniformly on each peer PE device.

You don't need to configure any corresponding elements on the multihomed CE device in the ES or on the remote source PE devices that send traffic to the ES. The CE devices and remote source PE devices can be devices that don't support this feature.

To enable the fast reroute ELP feature on a multihoming peer PE device for an ES:

1. Configure the IP address you want to use as the ELP tunnel source VTEP address for the ELP ES. You configure this address as a secondary address on the device loopback interface (lo0). The ELP tunnel address must be different from the device router ID address (which is the primary, preferred address). For example, on a PE device PE1 with router ID 192.168.0.1, you might use 192.168.102.1 as the IP address for the ELP tunnel to peer PE device PE2:

```
[edit]
user@PE1# show interfaces lo0
unit 0 {
    family inet {
        address 192.168.0.1/32 { # router ID
            primary;
            preferred;
        }
    }
}

[edit]
user@PE1# set interfaces lo0 unit 0 family inet address 192.168.102.1

[edit]
user@PE1# show interfaces lo0
unit 0 {
    family inet {
        address 192.168.0.1/32 { # router ID
            primary;
            preferred;
        }
        address 192.168.102.1/32 { # reroute ELP VTEP (different from router ID)
        }
    }
}

[edit]
```

2. Enable fast reroute ELP using the secondary lo0 address you configured in step 1.

When you commit this configuration, the PE device creates a corresponding ELP tunnel.

For example, using the secondary lo0 interface IP address from step 1:

```
[edit]
user@PE1# set forwarding-options evpn-vxlan reroute-address inet 192.168.102.1
```



NOTE: The ELP tunnel is idle until a member link on the PE device to a multihomed CE device fails. When the link fails, the remote source device continues to send some load-balanced traffic toward the failed link until route convergence. The PE device uses the ELP tunnel only to reroute any excess load-balanced traffic to its peer PE devices until the remote source stops sending traffic to it. After that point, the ELP tunnel is idle again.

Verify Fast Reroute ELP Tunnel Creation and Operation

IN THIS SECTION

- [show ethernet-switching table | 736](#)
- [show ethernet-switching vxlan-tunnel-end-point esi | 736](#)
- [show ethernet-switching vxlan-tunnel-end-point remote | 737](#)
- [show ethernet-switching vxlan-tunnel-end-point source | 738](#)
- [show route table __default_evpn__evpn.0 extensive | 738](#)
- [show route table bgp.evpn.0 extensive | 739](#)
- [show interfaces \(with ELP Tunnel VTEP Interfaces\) | 739](#)

When you enable the fast reroute ELP tunnel feature, you can use the CLI show commands in this section to see ELP tunnel IP addresses, next hop IDs, and source and remote VTEP information.

We explain some sample show command outputs here based on a topology similar to [Figure 64 on page 730](#), which we include again below for reference:

Figure 67: Sample Topology for Show Command Output

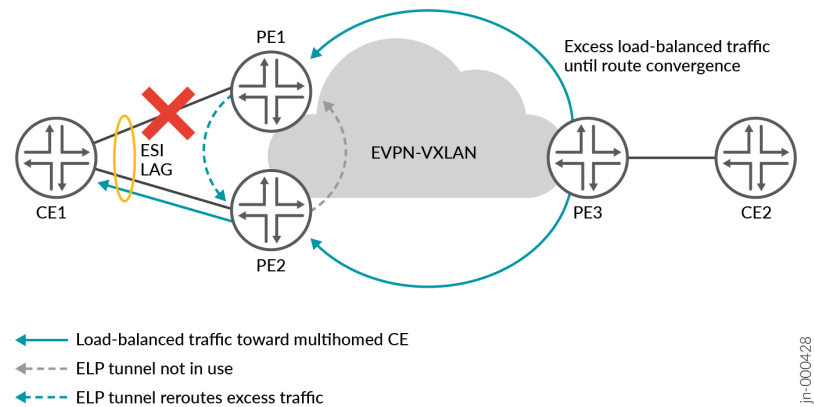


Table 41 on page 735 shows the sample topology primary lo0 IP addresses for the PE devices and secondary lo0 IP addresses for their ELP tunnels:

Table 41: Sample Topology IP Addresses for Multihoming Peer PE Devices

Device	Device IP Address	ELP Tunnel IP Address
PE1	192.168.0.1/32	192.168.102.1/32 ELP tunnel from PE1 to PE2
PE2	192.168.0.2/32	192.168.201.1/32 ELP tunnel from PE2 to PE1

The sample topology also has these hosts (not shown) behind the CE devices in Figure 67 on page 735:

Table 42: Sample Topology Host Information at Layer 2

Host Description	VLAN name (VLAN ID)	MAC Address
Receiver behind CE1	Vlan2 (VLAN ID 2)	00:00:94:00:00:01
Remote source behind CE2	Vlan3 (VLAN ID 3)	00:01:95:00:00:01

Finally, in this sample topology, the ESI assigned to the multihomed links from CE1 to PE1 and PE2 is 00:00:00:00:20:00:00:00:00:00.

show ethernet-switching table

Use the `show ethernet-switching table mac-address extensive` command to see ELP tunnel next hops for a particular MAC route.

For example, this output shows that PE1 will use the ELP tunnel next hop (ELP-NH field) to send any traffic it receives to the host behind CE1 on Vlan2 (MAC address 00:00:94:00:00:01):

```
user@PE1> show ethernet-switching table 00:00:94:00:00:01 extensive

MAC address: 00:00:94:00:00:01
Routing instance: L3VRF_MVS1
VLAN name: Vlan2, VLAN ID: 2
Learning interface: ae0.0
ELP-NH: 11777
Layer 2 flags:
in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd,adv_t_to_remote,rcvd_from_remote
Epoch: 0                      Sequence number: 3
Learning mask: 0x4000000000000000100000000000000001 BGP sequence number: 0
```

show ethernet-switching vxlan-tunnel-end-point esi

Use the `show ethernet-switching vxlan-tunnel-end-point esi esi-identifier esi-value` command to see the ELP tunnel VTEP IP address, next hops and VTEP interface name.

For example, this output shows PE1 has an ELP tunnel and ELP next hop (ELP-NH field) to remote VTEP PE2 (RVTEP-IP field) for ESI 00:00:00:00:20:00:00:00:00:00:

```
user@PE1> show ethernet-switching vxlan-tunnel-end-point esi esi-identifier
00:00:00:00:20:00:00:00:00:00

ESI                      RTT                      VLNBH INH      ESI-IFL  LOC-IFL  #RVTEPs
00:00:00:00:20:00:00:00:00:00 L3VRF_MVS1          28425 135034  esi.28425 ae0.0,    1
Aliasing
LOCAL-IFL  ELP-NH  RVTEP-NH-LIST
ae0.0      11777    11822,
RVTEP-IP   RVTEP-IFL  VENH  MASK-ID  FLAGS  MAC-COUNT
192.168.0.2      vtep-8.32830  11821  0        2      52
```

show ethernet-switching vxlan-tunnel-end-point remote

Use the `show ethernet-switching vxlan-tunnel-end-point remote` command to see ELP tunnel source VTEP and remote VTEP information about a device, including:

- VTEP IP addresses and interface names
- Corresponding next hop ID and ELP tunnel IP address

If the ELP-IP output field contains an IP address, that means the remote VTEP in the RVTEP-IP column has a corresponding ELP tunnel. Also, the ELP-IP column in the RVTEP-IP row contains “ELP” if that RVTEP-IP address is an ELP tunnel address.

See the sample output below and [Table 43 on page 737](#) for details on what this output means:

```
user@PE1> show ethernet-switching vxlan-tunnel-end-point remote
Logical System Name      Id  SVTEP-IP      IFL  L3-Idx  SVTEP-Mode  ELP-SVTEP-IP
<default>                0   192.168.0.1   lo0.0  0                192.168.102.1
RVTEP-IP      IFL-Idx  Interface  NH-Id  RVTEP-Mode  ELP-IP      Flags
192.168.0.2    2756    vtep.32830  11821  RNVE        192.168.201.1
192.168.201.1  2757    vtep.32831  11822  RNVE        ELP
VNID           MC-Group-IP
530            0.0.0.0
531            0.0.0.0
```

Table 43: Sample show ethernet-switching vxlan-tunnel-end-point remote Output Fields

Output Fields	Meaning
SVTEP-IP—192.168.0.1 ELP-SVTEP-IP—192.168.102.1	PE1 (192.168.0.1) is the source VTEP and has an ELP tunnel with IP address 192.168.102.1 to PE2.
First row under RVTEP-IP heading with: RVTEP-IP—192.168.0.2 ELP-IP—192.168.201.1	PE1 has PE2 (192.168.0.2) as a remote VTEP. PE2 has an ELP tunnel with IP address 192.168.201.1 to PE1.

Table 43: Sample show ethernet-switching vxlan-tunnel-end-point remote Output Fields (Continued)

Output Fields	Meaning
Third row under RVTEP-IP heading with: RVTEP-IP—192.168.201.1 ELP-IP—ELP	The remote VTEP with IP address 192.168.201.1 is an ELP tunnel remote VTEP. From the other output fields in this table, we know this remote VTEP is the ELP tunnel from PE2 to PE1.

show ethernet-switching vxlan-tunnel-end-point source

Use the `show ethernet-switching vxlan-tunnel-end-point source` command to see the device ELP tunnel source VTEP IP address in the ELP-SVTEP-IP output field.

For example, the following sample output on PE1 shows that PE1 (SVTEP -IP 192.168.0.1) has an ELP tunnel with IP address 192.168.102.1 (ELP-SVTEP-IP) on the lo0 interface:

```
user@PE1> show ethernet-switching vxlan-tunnel-end-point source
```

Logical System Name	Id	SVTEP-IP	IFL	L3-Idx	SVTEP-Mode	ELP-SVTEP-IP
<default>	0	192.168.0.1	lo0.0	0		192.168.102.1
L2-RTT		Bridge Domain		VNID	Translation-VNID	MC-Group-IP
L3VRF_MVS1		Vlan2+2		2		0.0.0.0
L3VRF_MVS1		Vlan3+3		3		0.0.0.0

show route table __default_evpn__.evpn.0 extensive

The `show route table __default_evpn__.evpn.0 extensive` routing table command includes the `Reroute address` field, which shows the local reroute ELP tunnel address.

For example, the `Reroute address` field below shows that the ELP tunnel on PE1 (192.168.0.1) has IP address 192.168.102.1:

```
user@PE1> show route table __default_evpn__.evpn.0 extensive
```

```
__default_evpn__.evpn.0: 2071 destinations, 2101 routes (2071 active, 0 holddown, 0 hidden)
1: 192.168.0.1:0::0::FFFF:FFFF/192 AD/ESI (1 entry, 1 announced)
    *EVPN    Preference: 170
              Next hop type: Indirect, Next hop index: 0
              Address: 0x8c06784
              Next-hop reference count: 32771, key opaque handle: 0x0, non-key opaque handle:
```

```

0x0
    Protocol next hop: 192.168.0.1

    Reroute address: 192.168.102.1
    Thread: junos-main

```

show route table bgp.evpn.0 extensive

The `show route table bgp.evpn.0 extensive` command includes the `Reroute address` field showing the device ELP tunnel address.

For example:

```

user@PE1> show route table bgp.evpn.0 extensive

bgp.evpn.0: 40116 destinations, 67780 routes (40116 active, 0 holddown, 0 hidden)
1: 192.168.0.1:0::0::FFFF:FFFF/192 AD/ESI (1 entry, 1 announced)
TSI:
Page 0 idx 0, (group iBGPGrp type Internal) Type 1 val 0xf820c88 (adv_entry)
  Advertised metrics:
    Flags: Nexthop Change
    Nexthop: Self
    Localpref: 100
    AS path: [55000] I

    Reroute address: 192.168.102.1
    Primary Routing Table: __default_evpn__.evpn.0
    Thread: junos-main

```

show interfaces (with ELP Tunnel VTEP Interfaces)

Use the `show interfaces` command with the ELP tunnel VTEP interface name to see when the device reroutes traffic on the ELP tunnel.

For example, the sample command output we saw earlier in ["show ethernet-switching vxlan-tunnel-end-point remote" on page 737](#) shows that the ELP tunnel VTEP interface name is `vtep.32831` (the `Interface` column with the remote VTEP for the tunnel to PE2).

If you enter the `show interfaces vtep.32831` command, before the link between CE1 and PE1 fails, the `Output packets` field shows that the traffic for the VTEP doesn't go over the ELP tunnel:

```
user@PE1> show interfaces vtep.32831
Logical interface vtep.32831 (Index 2757) (SNMP ifIndex 2585)
  Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
  VXLAN Endpoint Type: Shared ELP Remote, VXLAN Endpoint Address: 192.168.201.1, L3 Routing
Instance: default
  Input packets : 2
  Output packets: 1
  Protocol eth-switch, MTU: Unlimited
  Flags: Trunk-Mode
```

After the link between CE1 and PE1 fails, the same command shows that the traffic takes the ELP tunnel path for a short time (until source routing table convergence happens):

```
user@PE1> show interfaces vtep.32831
Logical interface vtep.32831 (Index 2757) (SNMP ifIndex 2585)
  Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
  VXLAN Endpoint Type: Shared ELP Remote, VXLAN Endpoint Address: 192.168.201.1, L3 Routing
Instance: default
  Input packets : 2
  Output packets: 3890
  Protocol eth-switch, MTU: Unlimited
  Flags: Trunk-Mode
```

SEE ALSO

[Dynamic Load Balancing in an EVPN-VXLAN Network | 723](#)

reroute-address

Setting Up a Layer 3 VXLAN Gateway

IN THIS CHAPTER

- [Using a Default Layer 3 Gateway to Route Traffic in an EVPN-VXLAN Overlay Network | 741](#)
- [Using a RIOT Loopback Port to Route Traffic in an EVPN-VXLAN Network | 748](#)
- [Supported Protocols on an IRB Interface in EVPN-VXLAN | 760](#)
- [VXLAN Layer 3 Gateways Using the Service Provider Style Interface Configuration | 762](#)

Using a Default Layer 3 Gateway to Route Traffic in an EVPN-VXLAN Overlay Network

IN THIS SECTION

- [Understanding the Default Gateway | 742](#)
- [Understanding the Redundant Default Gateway | 746](#)
- [Understanding Dynamic ARP Processing | 747](#)

Physical (bare-metal) servers in an Ethernet VPN-Virtual Extensible LAN (EVPN-VXLAN) environment rely on a default Layer 3 gateway to route their traffic from one virtual network (VN) to another physical server or a virtual machine (VM) in another VN. You can enable the default gateway functionality on a Juniper Networks device that acts as a Layer 3 VXLAN gateway. On a Layer 3 VXLAN gateway, you can configure an integrated routing and bridging (IRB) interface with a virtual gateway address (VGA), which in turn configures the IRB interface as a default Layer 3 gateway. You can configure an IRB interface with a VGA when using EVPN-VXLAN within a data center and across the Data Center Interconnect (DCI) solution.

Understanding the Default Gateway

To enable the default gateway function, you configure an IRB interface with a unique IP address and a media access control (MAC) address. In addition, you configure the IRB interface with a VGA, which must be an anycast IP address, and the Layer 3 VXLAN gateway automatically generates a MAC address.

When you specify an IPv4 address for the VGA, the Layer 3 VXLAN gateway automatically generates 00:00:5e:00:01:01 as the MAC address. When you specify an IPv6 address, the Layer 3 VXLAN gateway automatically generates 00:00:5e:00:02:01 as the MAC address.

On Juniper Networks devices that function as Layer 3 VXLAN gateways, you can explicitly configure an IPv4 or IPv6 MAC address for a default gateway by using the `virtual-gateway-v4-mac` or `virtual-gateway-v6-mac` configuration statement at the `[edit interfaces irb unit logical-unit-number]` hierarchy level. With this configuration, the device overrides the automatically generated MAC address with the configured MAC address.

A VGA and associated MAC address provide the default gateway function in a particular VN. You configure each host (physical server or VM) in the VN to use the VGA.

By using an anycast IP address as the VGA, when a VM is moved from one EVPN provider edge (PE) device to another in the same VN, the VM can use the same default gateway. In other words, you do not need to update the VM with a new default gateway IP address for MAC binding.

Layer 3 VXLAN gateways in an EVPN-VXLAN topology respond to Address Resolution Protocol (ARP) requests for the VGA and forward packets intended for the default gateway MAC address.



BEST PRACTICE: If you assign a VGA to your IRB interfaces, we recommend as a best practice to also configure a virtual gateway MAC (VMAC) address. You should assign the same VMAC to all IRB interfaces that you configure with the same VGA. That is, the VGA address used as the default gateway within a given VN also shares the same VMAC address on all IRB interfaces. The VGA and VMAC combination must be unique within each VN. Stated differently, within a VLAN you assign the same VGA and VMAC while between VNs you must configure unique VGA/VMAC combinations.

Table 44: VGA and VMAC Guidelines

Virtual Network	IRB VGA	IRB VMAC	Note
VN 1	10.0.1.254/24	00:05:85:00:01:01	Assign the same VGA and VMAC to all IRBs that service VN 1. These values differ from the ones assigned to the IRB for VN 2.
VN 2	10.0.2.254/24	00:05:85:00:02:02	Assign the same VGA and VMAC to all IRBs that service VN 2. These values differ from the ones assigned to the IRB for VN 1.

Following this recommendation avoids an asymmetric data paths for ARP requests and responses when the IRB interface sends ARP messages intended for an end-destination's MAC address.



BEST PRACTICE: On Juniper Networks devices that function as Layer 3 VXLAN gateways in an EVPN-VXLAN centrally-routed bridging overlay (EVPN-VXLAN topology with a two-layer IP fabric), we recommend that the IRB interface IP address is unique across the different Layer 3 VXLAN gateways for a given virtual network, and you configure a virtual gateway MAC address for the IRB. Following this recommendation avoids an asymmetric data path for the ARP request and response when the IRB interface sends ARP messages intended for an end-destination's MAC address. If you configure a virtual gateway MAC address for the IRB interface, we recommend you use a unique MAC address across the different Layer 3 VXLAN gateways, and in a given Layer 3 VXLAN gateway, use the same MAC address across different IRB units.

For IRB interfaces configured on QFX10000 switches in an EVPN-VXLAN edge-routed bridging overlay (EVPN-VXLAN topology with a two-layer IP fabric), you can alternatively configure each IRB interface on each Layer 3 VXLAN gateway in a VN with

the same MAC address. For more information, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway"](#) on page 845.



BEST PRACTICE:

Automatic ESI generation is enabled by default on devices in EVPN-VXLAN networks with EVPN multihoming for virtual gateway redundancy (see ["Understanding the Redundant Default Gateway"](#) on page 746). We recommend that you disable the automatic ESI generation for EVPN networks with edge-routed bridging overlays. To disable automatic ESI generation, include the `no-auto-virtual-gateway-esi` statement at the `[edit interfaces irb unit logical-unit-number]` hierarchy level.



NOTE: To troubleshoot an IRB interface, you can ping the IP address of the interface.

To troubleshoot a default gateway on an MX Series router, you can ping the VGA of the default gateway from a CE device. To support ping of the VGA, include the `virtual-gateway-accept-data` statement at the `[edit interfaces irb unit]` hierarchy of the preferred virtual gateway.

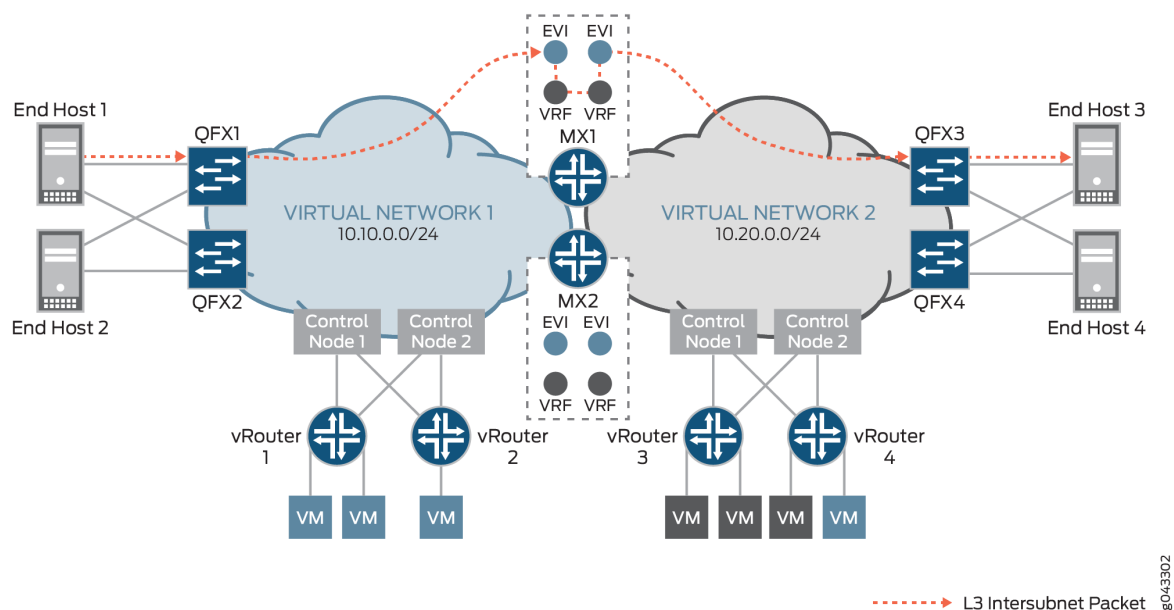
Additionally, you can ping the IP address of the CE device from the PE device (MX Series router). To support ping of the IP address of the CE device, include the preferred statement at `[edit interfaces irb unit logical-unit-number family {inet | inet6} address ip-address]` hierarchy using the unique IRB IP address. Otherwise, you must manually specify the unique IRB IP address as the source IP address when you ping the CE device.

For each IRB interface with a VGA configured, there are two sets of IP and MAC addresses—one set for the IRB interface itself and one set for the default gateway. As a result, the device advertises MAC routes for both IRB interface and default gateway. However, no default gateway extended community attribute is associated with the MAC route advertisement for the default gateway because all Layer 3 VXLAN gateways have the same anycast IP address and MAC binding.

Understanding How a Default Gateway Handles Known Unicast Traffic Between Virtual Networks

In the centrally-routed bridging overlay shown in [Figure 68 on page 745](#), MX Series routers function as Layer 3 VXLAN gateways and QFX5200 switches function as Layer 2 VXLAN gateways. End hosts 1 through 4 are physical servers that must communicate with each other.

Figure 68: Handling Known Unicast Traffic Between Virtual Networks



In this topology, end host 1 in VN1 (10.10.0.0/24) and end host 3 in VN2 (10.20.0.0/24) exchange known unicast packets. Before the exchange of packets between the two end hosts, assume that the hosts sent ARP requests to MX1, which is a Layer 3 VXLAN gateway, and that MX1 responded with the MAC address of a default gateway in VN1.

For example, end host 1 originates a packet and sends it to QFX1, which is a Layer 2 VXLAN gateway. QFX1 encapsulates the packet with a VXLAN header and sends it to MX1. For the inner destination MAC, the packet includes the MAC address of a default gateway in VN1. For the inner destination IP, the packet includes the IP address of end host 3. Upon receipt of the packet, MX1 de-encapsulates it, and after detecting the MAC address of the default gateway in the inner destination MAC field, performs a route lookup for end host 3's IP address in the L3-VRF routing table for VN1. After a route is found, the packet is routed to VN2 and based on the ARP route entry, the packet is encapsulated with a VXLAN header and sent to QFX3. QFX3 de-encapsulates the packet, and sends it to end host 3.



NOTE: The traffic flow and handling of known unicast traffic in an edge-routed bridging overlay are essentially the same as described in this section. The only difference is that in the edge-routed bridging overlay, a QFX Series switch that supports Layer 3 VXLAN gateway functionality acts as both Layer 2 and Layer 3 VXLAN gateways.

Understanding How a Default Gateway Handles Unknown Unicast Traffic Between Virtual Networks



NOTE: The information in this section applies to the traffic flow and handling of unknown unicast packets in both centrally-routed and edge-routed bridging overlays.

For unknown unicast traffic between VNs that is initiated by a physical server, an additional ARP request and response process is required at each stage. After the destination MAC addresses for both default gateway and host is resolved, the traffic flows in the same way as described in ["Understanding How a Default Gateway Handles Known Unicast Traffic Between Virtual Networks" on page 744](#).

Understanding the Redundant Default Gateway

The Juniper Networks devices that function as Layer 3 VXLAN gateways can also provide redundant default gateway functionality. A redundant default gateway prevents the loss of communication between physical servers in one VN and physical servers or VMs in another VN.

The redundant default gateway functionality is typically achieved in an EVPN-VXLAN topology where a provider edge (PE) device such as a Layer 2 VXLAN gateway or a Contrail vRouter is multihomed in active-active mode to multiple Layer 3 VXLAN gateways. On the Layer 3 VXLAN gateways, IRB interfaces are configured as default gateways. Note that each default gateway uses the same VGA and MAC address. In addition, the VGAs and MAC addresses are associated with the same Ethernet segment ID (ESI).

The ESI associated with the VGA and MAC address of the default gateway is automatically derived from an autonomous system (AS) and the VXLAN network identifier (VNI) for the VN. As a result, the default gateway MAC routes advertised by each Layer 3 VXLAN gateway for a given VN have the same ESI.

From the perspective of a Layer 2 VXLAN gateway or a Contrail vRouter that is multihomed to the Layer 3 VXLAN gateways, the addresses of each default gateway configured on each Layer 3 VXLAN gateway is the same. As a result, the PE devices build an equal-cost multipath (ECMP) next hop to reach each default gateway. Traffic that originates from a host and is destined for the MAC address of a default gateway is load balanced.

If one of the Layer 3 VXLAN gateways fails, the remote PE devices are notified of the withdrawing or purging of the next hop to the default gateway MAC address. The path to the failed Layer 3 VXLAN gateway is removed from the next-hop database. Despite the removal of the path, the default gateway that is configured on the remaining Layer 3 VXLAN gateway is still reachable, and the ARP entries for the hosts remain unchanged.

Understanding Dynamic ARP Processing

When a physical server needs to determine the MAC address of its default gateway, the physical server initiates an ARP request that includes the VFA of the default gateway. In a centrally-routed bridging overlay, a Layer 2 VXLAN gateway typically receives the ARP request, encapsulates the request in a VXLAN header, and forwards the encapsulated packet to a Layer 3 VXLAN gateway. In an edge-routed bridging overlay, a Layer 2 and 3 VXLAN gateway typically receives the ARP request from the directly connected physical server.

Upon receipt of the ARP request, the Layer 3 VXLAN gateway de-encapsulates the packet if appropriate, learns the IP and MAC binding of the physical server, and creates an ARP entry in its database. The Layer 3 VXLAN gateway then replies with the MAC address of the default gateway.

In a centrally-routed bridging overlay, the ARP response is encapsulated with a VXLAN header and unicast back to the Layer 2 VXLAN gateway. The Layer 2 VXLAN gateway de-encapsulates the ARP response and forward the packet to the physical server.

In an edge-routed bridging overlay, the ARP response is unicast back to the directly connected physical server.

In a situation where a physical server in VN1 originates a packet that is destined for a physical server in VN2, the Layer 3 VXLAN gateway searches its database for an ARP entry for the destination physical server. If a match is not found, the Layer 3 VXLAN gateway initiates an ARP request that includes the IP and MAC addresses of the IRB interface that is mapped to VN2, and sends the request to the destination physical server. The destination physical server learns the IP/MAC binding of the IRB interface, and adds or refreshes the ARP entry in its database accordingly. The physical server then unicasts an ARP response, which includes the MAC address of the IRB interface, back to the Layer 3 VXLAN gateway,

RELATED DOCUMENTATION

[EVPN Multihoming Overview | 162](#)

[Understanding EVPN with VXLAN Data Plane Encapsulation | 588](#)

[EVPN Over VXLAN Encapsulation Configuration Overview for QFX Series and EX4600 Switches | 628](#)

[EVPN-over-VXLAN Supported Functionality | 599](#)

Using a RIOT Loopback Port to Route Traffic in an EVPN-VXLAN Network

SUMMARY

You can configure a RIOT loopback port on a device that doesn't support native VXLAN routing. With this feature, the device can serve as a Layer 3 VXLAN gateway in an EVPN-VXLAN fabric.

IN THIS SECTION

- [Loopback Port Solution for Routing in and out of VXLAN Tunnels \(RIOT\) for Layer 3 VXLAN Gateway Support | 748](#)
- [Configure a RIOT Loopback Port on a Layer 3 VXLAN Gateway Leaf Device | 754](#)

Loopback Port Solution for Routing in and out of VXLAN Tunnels (RIOT) for Layer 3 VXLAN Gateway Support

IN THIS SECTION

- [RIOT Loopback Solution Overview | 748](#)
- [How RIOT Loopback Processing Works | 750](#)
- [Asymmetric Type 2 Routes with RIOT Loopback Processing | 752](#)
- [Symmetric Type 2 and Type 5 Routes with RIOT Loopback Processing | 752](#)
- [IRB Interface Status Dependency on RIOT Loopback Port State | 753](#)

Some Juniper Networks EVPN-VXLAN fabric devices, such as QFX5210 switches, don't have native support for routing in and out of VXLAN tunnels (RIOT). Starting in Junos OS Release 21.3R1, you can configure a loopback port on supporting devices to perform RIOT operations in a two-pass process. With this solution, you can use the device as a Layer 3 VXLAN gateway device in an EVPN-VXLAN edge-routed bridging overlay (ERB) fabric.

RIOT Loopback Solution Overview

You can configure a Layer 3 VXLAN gateway using a RIOT loopback port in an EVPN-VXLAN ERB fabric with:

- MAC VRF routing instances—either VLAN-based or VLAN-aware bundle service type.

See ["MAC-VRF Routing Instance Type Overview" on page 38](#) for more on MAC-VRF routing instances.

- Enterprise style interface configuration.

See *Flexible Ethernet Services Encapsulation* and ["Understanding Flexible Ethernet Services Support With EVPN-VXLAN" on page 654](#) for details on enterprise style and flexible Ethernet services configuration.

- EVPN Type 2 routing (with both the asymmetric and symmetric integrated routing and bridging [IRB] models) and EVPN Type 5 routing.

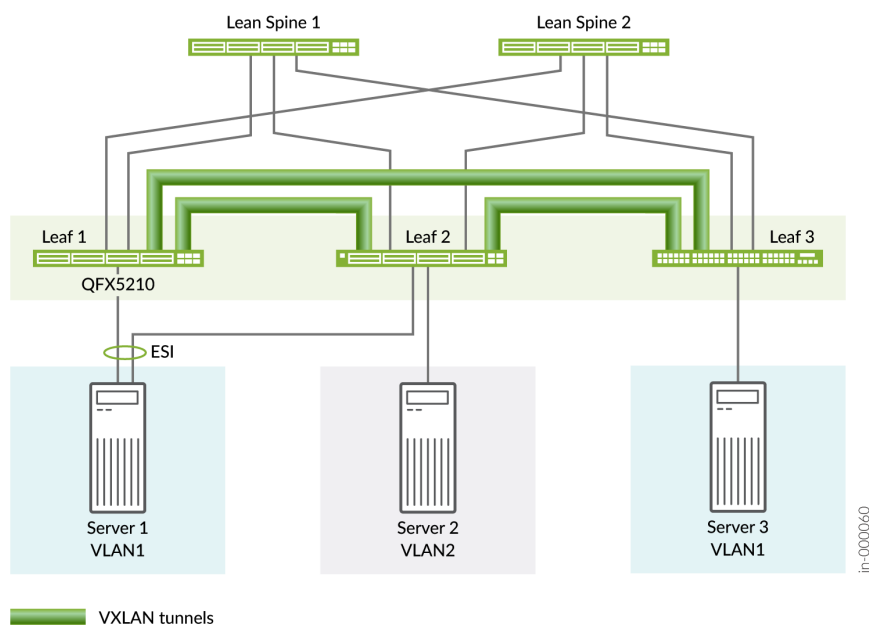
Starting in Junos OS Release 21.3R1-S1 and 21.4R1, you can enable symmetric EVPN Type 2 routing on QFX5210 switches that act as Layer 3 gateways. See ["Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics" on page 59](#) for more details on the symmetric IRB model.



NOTE: We support EVPN Type 5 routing using only the symmetric IRB model. This is the default behavior when you configure a routing instance to use Type 5 routes with the `ip-prefix-routes` statement. See ["Understanding EVPN Pure Type 5 Routes" on page 50](#) for an overview of EVPN Type 5 routes and other EVPN route types.

The following figure shows an EVPN-VXLAN ERB fabric. The leaf devices route traffic through VXLAN tunnels to the other leaf devices in the fabric. Leaf 1 is a QFX5210 switch that doesn't support native VXLAN routing in and out of the VXLAN tunnel.

Figure 69: EVPN-VXLAN ERB Overlay Fabric with a RIOT Loopback Layer 3 Gateway Leaf Device

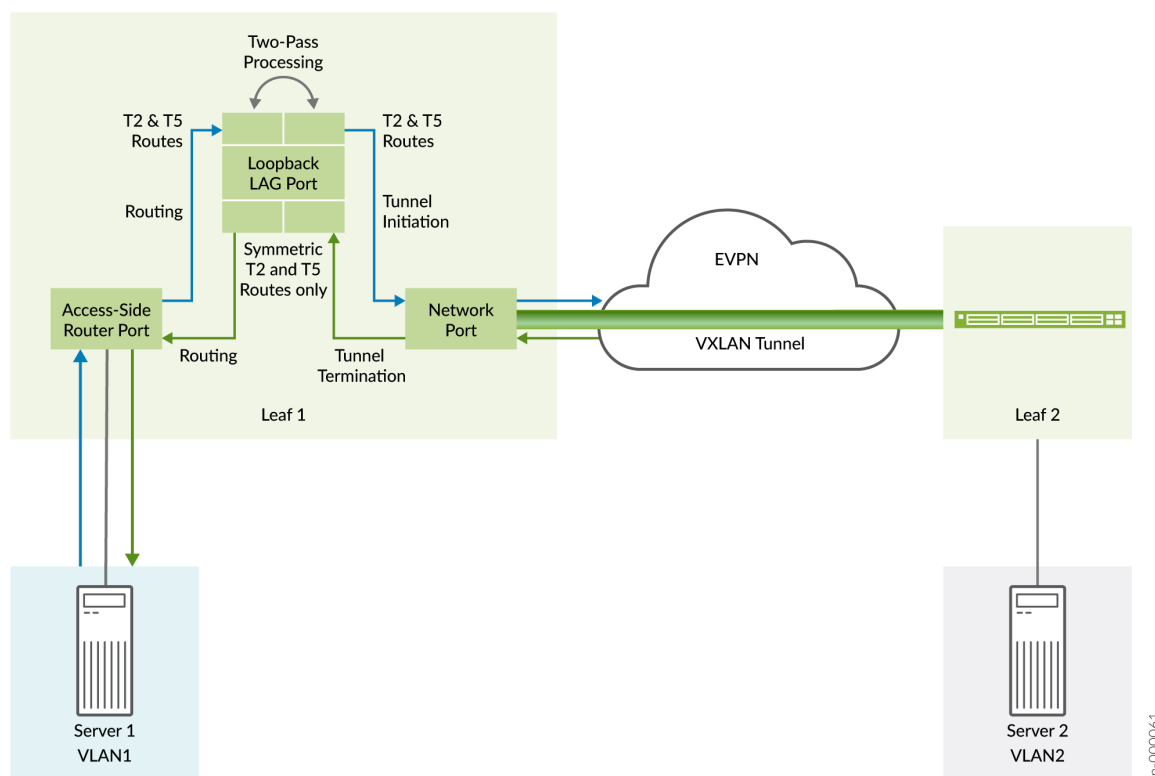


For Leaf 1 to serve as Layer 3 VXLAN gateway, you need to configure the RIOT loopback port solution on that device. RIOT loopback routing is transparent to the other leaf devices that connect through VXLAN tunnels in the fabric. You can include Layer 3 gateway leaf devices in the same fabric that use RIOT loopback port routing with devices that use native VXLAN routing.

How RIOT Loopback Processing Works

The following figure shows how the RIOT loopback process routes traffic in or out of a VXLAN tunnel that connects to another leaf device.

Figure 70: RIOT Loopback Two-Pass Processing



You configure the RIOT loopback port as an Ethernet link aggregation group (LAG). Adjust the number of member links depending on the bandwidth of VXLAN traffic that uses the loopback path. You can use any network ports for the LAG that aren't already used for another purpose. The device automatically turns off MAC learning on the RIOT loopback LAG so the port doesn't learn its own MAC address when traffic passes through it.

The traffic flows through the RIOT loopback LAG in the first pass, then loops back into the RIOT loopback LAG for the second pass. What happens during the RIOT loopback process depends on the direction of traffic flow and the type of routes.

The device uses the RIOT loopback process for:

- Access port to network port routing and VXLAN tunnel initiation with asymmetric or symmetric Type 2 routing and Type 5 routing.
- Network port to access port VXLAN tunnel termination and routing with symmetric Type 2 routing and Type 5 routing only.

The device doesn't need to use the RIOT loopback LAG in the following cases:

- Access port to access port with asymmetric Type 2 routing.

The device routes the traffic locally through the IRB interfaces on the device as usual (no VXLAN bridging needed).

- Network port to access port with asymmetric Type 2 routing.

In this case, the ingress VTEP already routed the traffic exiting the VXLAN tunnel to the destination VLAN. The device uses normal Layer 2 VXLAN traffic processing to bridge the traffic on the destination VLAN.

We describe more about the RIOT loopback process with different EVPN route types next. Also, see ["Configure a RIOT Loopback Port on a Layer 3 VXLAN Gateway Leaf Device" on page 754](#) for details on what you need to configure for each EVPN route type.

Asymmetric Type 2 Routes with RIOT Loopback Processing

With asymmetric Type 2 routing, all VLANs extend over the EVPN network on all devices. The integrated bridging and forwarding (IRB) actions on the two VXLAN tunnel endpoints (VTEP) differ as follows:

- The ingress VTEP IRB interface routes the traffic from the source VLAN to the destination VLAN. Then the device bridges the traffic on the destination VLAN across the VXLAN tunnel.



NOTE: With asymmetric routing, you must configure all destination VLANs on the ingress VTEP even if the device doesn't serve hosts on all VLANs.

- The egress VTEP receives the traffic on the destination VLAN, and then forwards the traffic on the destination VLAN. The egress VTEP doesn't need to route the traffic.

This means that devices don't need the RIOT loopback process with asymmetric Type 2 routes on VXLAN tunnel traffic coming *into* the device from the EVPN network.

Symmetric Type 2 and Type 5 Routes with RIOT Loopback Processing

For symmetric Type 2 routes and Type 5 routes to work with the RIOT loopback process, the device uses an extra VLAN for each tenant Layer 3 (L3) virtual routing and forwarding (VRF) instance. Each extra VLAN maps to the VXLAN network identifier (VNI) for the corresponding L3 VRF instance.



NOTE: Symmetric Type 2 routing requires Layer 3 reachability between the host subnetworks. An L3 VRF in which you have enabled Type 5 IP prefix routes can provide Layer 3 connectivity.

As a result, the RIOT loopback solution uses a VRF with Type 5 routes enabled to support the Layer 3 connectivity for symmetric Type 2 routing across the VXLAN tunnels. You use the same L3 VRF instance for both symmetric Type 2 routing and Type 5 routing.

The extra VLAN enables symmetric Type 2 or Type 5 routing in both directions (to and from the VXLAN tunnel) as follows:

- Access port to network port traffic:

In the first pass, the RIOT loopback process routes the traffic out of the RIOT loopback port. In the second pass for tunnel initiation, the RIOT loopback process needs the VNI for the corresponding L3 VRF instance. The RIOT loopback process uses the extra VLAN's VLAN-to-VNI mapping for that purpose.

- Network port to access port traffic

When terminating the VXLAN tunnel, the device needs a VLAN tag with which to send the traffic out of the RIOT loopback port. The RIOT loopback process adds the extra VLAN ID as the VLAN tag in the first pass. In the second pass, the RIOT loopback process uses the VLAN tag to find the corresponding L3 VRF instance to do the route lookup.

IRB Interface Status Dependency on RIOT Loopback Port State

Layer 3 VXLAN gateway devices route traffic between VLANs using IRB interfaces. On devices with RIOT loopback processing, all IRB interfaces you configure for VXLAN routing depend on the RIOT loopback LAG. As a result, the RIOT loopback LAG must be available to process VXLAN traffic before the IRB interfaces are able to route traffic. For this feature to work, the device must consider the RIOT loopback LAG state when determining IRB interface status.

With RIOT loopback configuration, you configure the device to include the state of another local interface when evaluating the status of an IRB interface. In this case, the local interface is the RIOT loopback LAG.

Use the `local-interface name` statement at the `[edit interfaces irb unit unit-number interface-state]` hierarchy. Specify *name* as the logical interface name of the RIOT loopback LAG for the unit.

Also, to configure the delay to ensure that the RIOT loopback LAG is up before the device evaluates the IRB interface as up, you configure the `hold-time up seconds` option at the `[edit interfaces irb unit unit-number interface-state]` hierarchy. This value is the time the device waits after the RIOT loopback interface is up before it includes that state when evaluating the IRB interface status.

The RIOT loopback LAG usually remains up unless you change its configuration. As a result, we recommend to set `hold-time up` to a higher value based on the scale of routes in your network. A higher

value helps prevent the IRB logical interfaces from flapping. We recommend that you try approximately 120 seconds for medium-to-large scale deployments.

Configure a RIOT Loopback Port on a Layer 3 VXLAN Gateway Leaf Device

Follow these steps to configure a device to use the RIOT loopback process so it can operate as a Layer 3 VXLAN gateway using EVPN Type 2 or Type 5 routing. Some configuration steps are common for asymmetric Type 2 routing, symmetric Type 2 routing, and EVPN Type 5 routing, such as:

- Configure the RIOT loopback LAG and the IRB interfaces for VXLAN routing.
- Set the RIOT loopback LAG as the interface that handles RIOT loopback processing on the device.

You perform a few extra steps for symmetric Type 2 and Type 5 routing, including:

- Configure an extra VLAN for each EVPN L3 VRF instance. You don't use this VLAN for any other purpose. The L3 VRF instances provide the Layer 3 connectivity on which the device transfers VXLAN packets for both symmetric Type 2 routing and Type 5 routing.
- Configure an IRB interface on each extra VLAN.
- Map the extra VLAN to the VXLAN network identifier (VNI) of the corresponding L3 VRF instance.

See ["Symmetric Type 2 and Type 5 Routes with RIOT Loopback Processing" on page 752](#) for more information on the extra VLAN for symmetric Type 2 and Type 5 routing. See ["How RIOT Loopback Processing Works" on page 750](#) for more information on the differences in the RIOT loopback process among the supported route types.



NOTE: Devices that require the RIOT loopback solution to act as a Layer 3 VXLAN gateway include the following statement in the default configuration:

```
set protocols evpn riot-loopback
```

This statement globally enables the RIOT loopback process on the device. You don't need to configure this statement explicitly.

To configure RIOT loopback processing:

1. Define an aggregated Ethernet interface for the RIOT loopback port. Use any network ports on the device in the RIOT loopback LAG that you aren't already using for network traffic.

The sample configuration below first allocates some number of aggregated Ethernet interfaces, and uses an available one (ae0) for the RIOT loopback LAG. For simplicity, this configuration includes one link in the RIOT loopback LAG, which must be up for the interface to be up. You can adjust the

number of member links in the RIOT loopback LAG depending on the bandwidth of VXLAN traffic that uses the loopback path.

```
set chassis aggregated-devices ethernet device-count number

set interfaces xe-0/0/10:1 ether-options 802.3ad ae0
set interfaces ae0 aggregated-ether-options loopback
set interfaces ae0 aggregated-ether-options minimum-links 1
```

2. Configure the RIOT loopback LAG in the enterprise style with:

- Flexible VLAN tagging.
- Flexible Ethernet services encapsulation (so the interface can have multiple logical units).

For example:

```
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
```

3. Configure the RIOT loopback LAG interface in enterprise style as a member of all VLANs (units) that have IRB interfaces for which the device does Layer 3 VXLAN gateway routing. Also, configure the interface in trunk mode for each unit.

For example, here the fabric serves three VXLAN VLANs: V100, V110, and V120. Configure the RIOT loopback LAG in each VLAN with interface-mode trunk:

```
set interfaces ae0 unit 100 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 100 family ethernet-switching vlan members V100

set interfaces ae0 unit 110 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 110 family ethernet-switching vlan members V110

set interfaces ae0 unit 120 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 120 family ethernet-switching vlan members V120
```

4. Configure an IRB interface for each VLAN (unit) used for VXLAN routing. This step isn't specific to the RIOT loopback process. However, it is a required part of the EVPN-VXLAN fabric setup. You use these IRB interfaces in subsequent steps.

For example, configure IRB interfaces for units 100, 110, and 120:

```
set interfaces irb unit 100 family inet address 10.100.1.254/16
set interfaces irb unit 100 family inet6 address 2001:db8::0192:0100:0001:0254/96
set interfaces irb unit 100 mac 00:01:01:00:00:fe

set interfaces irb unit 110 family inet address 10.110.1.254/16
set interfaces irb unit 110 family inet6 address 2001:db8::0192:0110:0001:0254/96
set interfaces irb unit 110 mac 00:01:01:10:00:fe

set interfaces irb unit 120 family inet address 10.120.1.254/16
set interfaces irb unit 120 family inet6 address 2001:db8::0192:0120:0001:0254/96
set interfaces irb unit 120 mac 00:01:01:20:00:fe
```

5. For each IRB interface, set the RIOT loopback LAG as a local interface whose state the device includes in evaluating the IRB interface state (up or down). Use the `local-interface name` statement at the `[edit interfaces irb unit unit-number interface-state hierarchy`. Specify the logical interface name of the RIOT loopback LAG for the local interface name. Also set the `hold-time up` option to ensure the RIOT loopback LAG is up before the device evaluates the IRB interface as up. See ["IRB Interface Status Dependency on RIOT Loopback Port State" on page 753](#) for more information on why we need this step.

For example, for IRB interfaces configured on units 100, 110, and 120, set the local interface to the RIOT loopback LAG logical interface name. Specify a hold time for each IRB—we recommend 120 seconds in this case for a medium-to-large scale deployment:

```
set interfaces irb unit 100 interface-state local-interface ae0.100
set interfaces irb unit 100 interface-state hold-time up 120

set interfaces irb unit 110 interface-state local-interface ae0.110
set interfaces irb unit 110 interface-state hold-time up 120

set interfaces irb unit 120 interface-state local-interface ae0.120
set interfaces irb unit 120 interface-state hold-time up 120
```

6. Set the RIOT loopback LAG as the interface the device uses for the RIOT loopback process for all VXLAN routing. Use the statement `loopback-port loopback-port` at the `[edit forwarding options vxlan-routing]` hierarchy level. With this statement, specify the physical interface name of the RIOT loopback port.

For example, in our sample configuration in earlier steps, the RIOT loopback LAG is ae0:

```
set forwarding-options vxlan-routing loopback-port ae0
```

7. Define each VXLAN VLAN. Set the IRB interfaces as Layer 3 IRB interfaces for each VLAN, and map the VLANs to VNI values. We require this step for VXLAN gateway configuration; it isn't specific to RIOT loopback configuration.

For example:

```
set vlans V100 vlan-id 100
set vlans V100 l3-interface irb.100
set vlans V100 vxlan vni 1100

set vlans V110 vlan-id 110
set vlans V110 l3-interface irb.110
set vlans V110 vxlan vni 1110
```

8. (Symmetric Type 2 and Type 5 use cases only) Configure an additional VLAN for each L3 VRF instance. The RIOT loopback process uses this VLAN and its corresponding VNI to support symmetric Type 2 and Type 5 routing. See ["Symmetric Type 2 and Type 5 Routes with RIOT Loopback Processing" on page 752](#) and ["Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics" on page 59](#) for details. This VLAN is dedicated to this purpose and must be different from any tenant VLANs or VXLAN VLANs the device hosts.

This step combines the earlier steps where you configure the RIOT loopback LAG as part of the VXLAN VLANs. You do the same for this extra VLAN, including:

- Configure the VLAN with an IRB interface.
- Configure the RIOT loopback LAG logical interface for this unit in trunk mode.
- Set the RIOT loopback LAG interface as an IRB-enabled member of this VLAN.
- Set the RIOT loopback LAG as a local interface whose state the device includes in evaluating the IRB interface state (up or down).

For example, define an IRB-enabled VLAN named V-L3-RIOT1 with VLAN ID 999. Include the RIOT loopback LAG as a part of this VLAN. Also set the other parameters listed above that enable the RIOT loopback process:

```
set vlans V-L3-RIOT1 vlan-id 999
set vlans V-L3-RIOT1 l3-interface irb.999
```

```

set interfaces ae0 unit 999 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 999 family ethernet-switching vlan members V-L3-RIOT1

set interfaces irb unit 999 family inet
set interfaces irb unit 999 mac 00:01:01:00:00:fe

set interfaces irb unit 999 interface-state local-interface ae0.999
set interfaces irb unit 999 interface-state hold-time up 120

```



NOTE: Repeat this step to create an extra VLAN for each L3 VRF instance.

9. (Symmetric Type 2 and Type 5 use cases only) Configure the extra VLAN's IRB logical interface in the L3 VRF instances where you enable Type 5 routing. Both symmetric Type 2 and Type 5 routing require this configuration for the Layer 3 connectivity. Map the additional VLAN to a VNI that matches the EVPN encapsulation VNI you configure in the L3 VRF instance.



NOTE: To enable Type 5 routing in an EVPN-VXLAN fabric, you set up an L3 VRF instance. In that instance, you configure the `ip-prefix-routes vni vni-value` statement at the `[edit routing-instances type-5-instance-name protocols evpn]` hierarchy level. This *vni-value* is the value you map to the extra VLAN.

Note that QFX5210 switches don't support asymmetric VNI values on either side of a VXLAN tunnel for a given VRF. To support EVPN Type 5 routing and symmetric IRB routing with EVPN Type 2 routes on QFX5210 switches, you must configure the same L3 VNI value for a given VRF on each of the leaf devices.

We don't include all of the standard EVPN-VXLAN L3 VRF instance configuration here. See ["EVPN Type 5 Route with VXLAN Encapsulation for EVPN-VXLAN" on page 47](#) for more information on Type 5 routes. Also see [Configuring EVPN Type 5 for QFX10000 Series Switches: Configuration Example](#) for an example configuration with Type 5 routing between two QFX Series devices in an EVPN network.

For example, if you configure the L3 VRF instance L3-VRF with an EVPN-VXLAN encapsulation VNI value of 5000 as follows:

```

set routing-instances L3-VRF protocols evpn ip-prefix-routes vni 5000

```

then map the extra VLAN from step 8 (V-L3-RIOT1 with VLAN ID 999) to VNI 5000:

```
set routing-instances L3-VRF interface irb.999
set vlans V-L3-RIOT1 vxlan vni 5000
```



NOTE: Repeat this step for the extra VLAN and VNI for each L3 VRF instance.

10. (Symmetric Type 2 use case only) Enable symmetric Type 2 routing in the L3 VRF instances from step 9 if you want to use symmetric routing with EVPN Type 2 routes.

Type 2 routing is asymmetric by default, so you must explicitly enable symmetric routing using the `irb-symmetric-routing vni vni` configuration statement at the [edit routing-instances *name* protocols evpn] hierarchy. You must specify the VNI as the same EVPN-VXLAN encapsulation VNI that you set for EVPN Type 5 routing in 9.

For example, following the previous configuration steps, enable symmetric routing with Type 2 routes using VNI 5000:

```
set routing-instances L3-VRF protocols evpn irb-symmetric-routing vni 5000
```



NOTE: Repeat this step for the extra VLAN and VNI for each L3 VRF instance.

11. (Symmetric Type 2 and Type 5 use cases only) Finally, configure the `riot-loopback` statement at the [edit vlans *name* vxlan] hierarchy. This statement sets the VLAN from step 8 as the extra RIOT loopback VLAN for symmetric Type 2 and Type 5 routing.

For example:

```
set vlans V-L3-RIOT1 vxlan riot-loopback
```



NOTE: Repeat this step for the extra VLAN for each L3 VRF instance.

SEE ALSO

[Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway | 845](#)

[Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics | 59](#)

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
21.4R1	Starting in Junos OS Release 21.3R1-S1 and 21.4R1, you can enable symmetric EVPN Type 2 routing on QFX5210 switches that act as Layer 3 gateways.

Supported Protocols on an IRB Interface in EVPN-VXLAN

EVPN-VXLAN provides logical layer 2 connectivity over a layer 3 network. It is a logical overlay that is decoupled from the underlay network. The layer 2 subnets in the EVPN-VXLAN network are uniquely identified by the virtual network identifier (VNI) where devices configured with the same VNI are considered to be in the same logical subnet. The devices in the same logical subnet can communicate directly with each other when they are connected to the same virtual gateway. To route traffic when devices with different VNIs are connected to the same gateway, you must configure an IRB interface on the VXLAN gateway to route traffic. This allows protocol adjacencies to be established between the layer 3 gateway IRB interface and customer edge devices in the following scenarios:

- Support networking functions when there are firewalls or load balancers connected between the host and the gateway device.
- Support inter-VNI routing when VNIs for the same tenant extend across data centers.

[Figure 71 on page 761](#) illustrates a simple leaf-spine (centrally-routed bridging overlay) topology in an EVPN-VXLAN network with the VXLAN gateway on spine devices. For host 1 to communicate with hosts in other VNIs, you must configure the IRB interfaces on the VXLAN gateway on the spine devices.



NOTE: When configuring an ESI interface for multihomed devices, Junos OS only supports routing protocols that are configured on an IRB in the all-active multihoming mode in a centrally-routed bridging overlay EVPN-VXLAN network.

Figure 71: Leaf-spine Topology in an EVPN-VXLAN network.

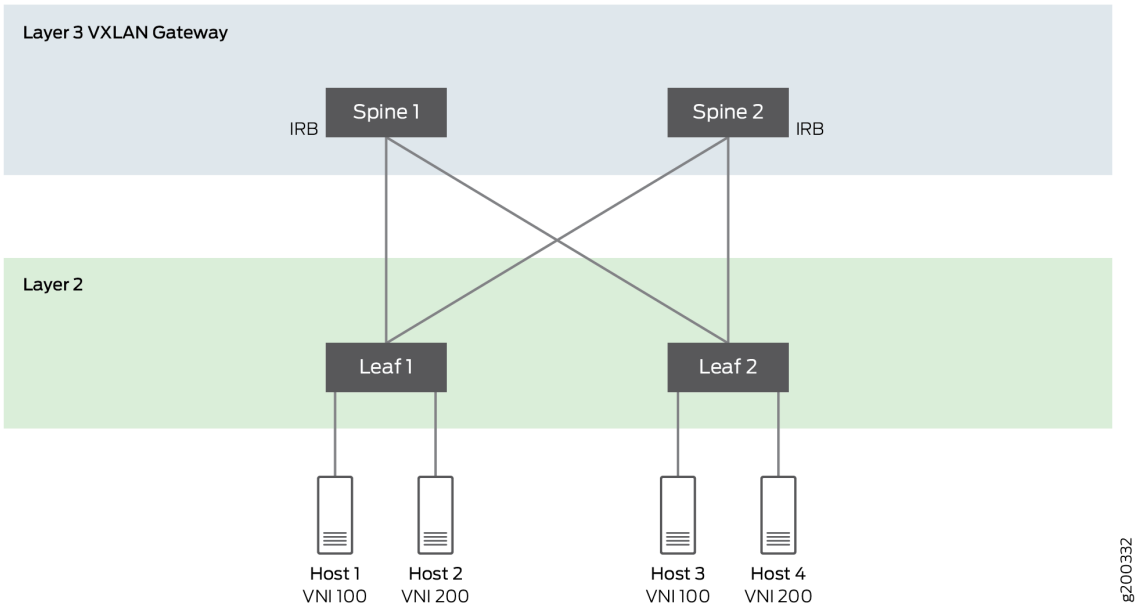
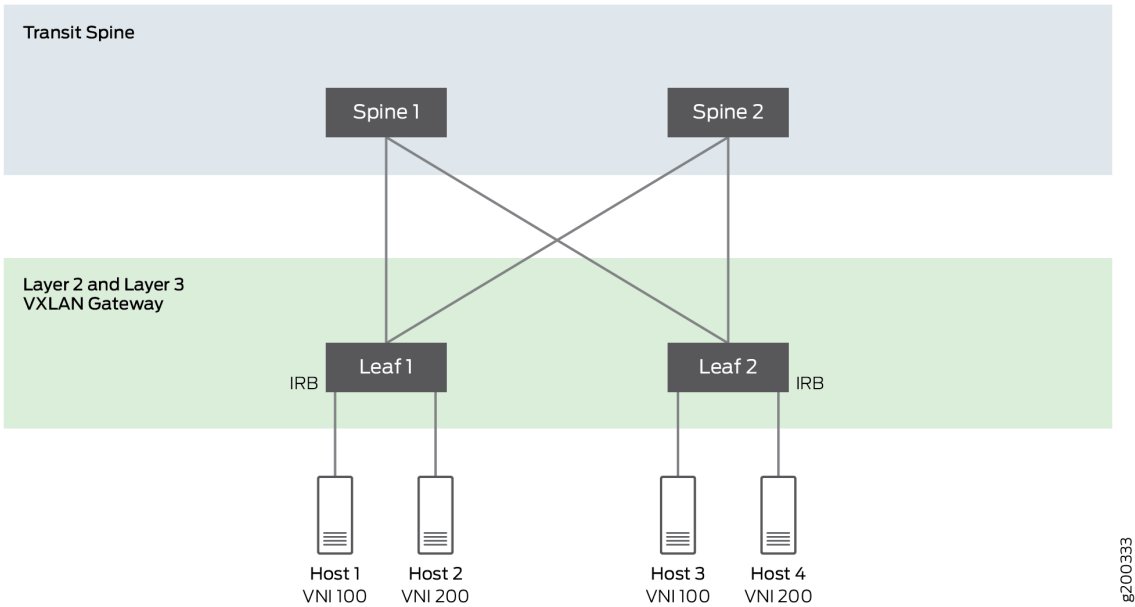


Figure 72 on page 761 illustrates a collapsed leaf-spine (edge-routed bridging) topology with VXLAN gateways on leaf devices. For host 1 to communicate with hosts in other VNIs, you must configure IRB interfaces on the VXLAN gateway on the leaf devices.

Figure 72: Collapsed Leaf-spine Topology in an EVPN-VXLAN network.



Junos OS supports the following protocols on the IRB in the EVPN-VXLAN overlay network:

- OSPF
- IS-IS
- BGP (eBGP and iBGP)
- Bidirectional forwarding detection (BFD) support for ISIS, OSPF, BGP and static routing.

RELATED DOCUMENTATION

Understanding OSPF Routing Instances

[Understanding IS-IS Configuration](#)

[Understanding External BGP Peering Sessions](#)

[Understanding Internal BGP Peering Sessions](#)

[Understanding BFD for Static Routes for Faster Network Failure Detection](#)

VXLAN Layer 3 Gateways Using the Service Provider Style Interface Configuration

IN THIS SECTION

- [Benefits | 763](#)
- [Layer 3 Gateway in an ERB Overlay Using the Service Provider Style Interface Configuration | 764](#)
- [Layer 3 Gateway in a CRB Overlay Configuration Using the Service Provider Style Interface Configuration | 766](#)

In platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS) deployments, customer cloud and private cloud providers often use the same leaf device to forward their network traffic over a VXLAN core network. Customer cloud and private cloud providers can now use the service provider style interface configuration CLI to configure a leaf device to act as a Layer 3 gateway. They can also use the service provider style CLI to configure multiple interfaces that map to the same VLAN on the same leaf device.

The service provider style CLI is available on the following devices:

- EX4650, QFX5110, QFX5120-32C, QFX5120-48T, QFX5120-48Y, QFX5120-48YM—Starting in Junos OS Release 22.2R1

You can use the service provider style CLI to perform these tasks:

- Configure multiple logical interfaces on a single physical interface.
- Configure your leaf device as a Layer 3 gateway in edge-routed bridging (ERB) and centrally-routed bridging (CRB) overlays.
- Map an integrated routing and bridging (IRB) interface to a virtual network identifier (VNI).
- Perform VXLAN routing.
- Configure an ERB overlay.
- Configure a CRB overlay.
- Configure a VLAN ID.
- Configure a VLAN ID as none.
- Configure a VLAN-aware bundle service.
- Configure a virtual gateway address.



NOTE: We require that you configure the `no-gateway-community` option at the `[edit routing-instances EVPN-instance-name protocols evpn default-gateway]` hierarchy level in each EVPN routing instance in which you configure an IRB interface with a virtual gateway address. You can alternatively configure the `no-gateway-community` option globally at the `[edit protocols evpn default-gateway]` hierarchy level on platforms that support configuration at this level, as this example shows. See *default-gateway* for details on using the `no-gateway-community` option.

- Map an IRB interface to one VLAN or multiple IRB interfaces to multiple VLANs.
- Assign a VLAN ID to an IRB interface with same VLAN ID as the VLANS IDs or assign a different VLAN ID.

Benefits

- You can use the service provider style interface style to configure a leaf device to act as a Layer 3 gateway.
- You can also use the service provider style interface configuration to configure multiple interfaces on a single leaf device and map them to the same VLAN.

Layer 3 Gateway in an ERB Overlay Using the Service Provider Style Interface Configuration

Leaf 1

```

set chassis aggregated-devices ethernet device-count 20
set interfaces et-2/0/5 description "To PORT-1 --> 1/13"
set interfaces et-2/0/5 flexible-vlan-tagging
set interfaces et-2/0/5 encapsulation extended-vlan-bridge
set interfaces et-2/0/5 unit 100 vlan-id 100
set interfaces et-2/0/5 unit 200 vlan-id 200
set interfaces et-2/0/8 description "To Spine-1 in ae31"
set interfaces et-2/0/8 ether-options 802.3ad ae31
set interfaces et-2/0/11 description "To Spine-1 in ae31"
set interfaces et-2/0/11 ether-options 802.3ad ae31
set interfaces ae31 description "To Spine-1"
set interfaces ae31 aggregated-ether-options lacp active
set interfaces ae31 aggregated-ether-options lacp periodic fast
set interfaces ae31 aggregated-ether-options lacp system-id 11:11:11:11:11:11
set interfaces ae31 unit 0 family inet address 10.40.1.2/24
set interfaces et-2/0/9 description "To spine-2"
set interfaces et-2/0/9 unit 0 family inet address 10.50.1.2/24
set interfaces et-2/0/12:0 description "To CE-1 in ae12"
set interfaces et-2/0/12:0 ether-options 802.3ad ae12
set interfaces ae12 description "To CE-1"
set interfaces ae12 flexible-vlan-tagging
set interfaces ae12 encapsulation extended-vlan-bridge
set interfaces ae12 esi 00:12:12:12:12:12:12:12:12:12
set interfaces ae12 esi all-active
set interfaces ae12 aggregated-ether-options lacp active
set interfaces ae12 aggregated-ether-options lacp periodic fast
set interfaces ae12 aggregated-ether-options lacp system-id 12:12:12:12:12:12
set interfaces ae12 unit 100 vlan-id 100
set interfaces ae12 unit 200 vlan-id 200
set interfaces lo0 unit 0 family inet address 10.10.10.10/24 primary
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 family inet address 10.100.1.1/24 preferred
set interfaces irb unit 100 family inet address 10.100.1.1/24 virtual-gateway-address
10.100.1.254
set interfaces irb unit 100 family inet6 address abcd::0010:0100:0001:0001/96 preferred
set interfaces irb unit 100 family inet6 address abcd::abcd::0010:0100:0001:0001/96 virtual-

```

```

gateway-address abcd::0192:0100:0001:0254
set interfaces irb unit 100 virtual-gateway-v4-mac 00:10:01:00:01:fe
set interfaces irb unit 100 virtual-gateway-v6-mac 00:10:01:00:02:fe
set interfaces irb unit 200 virtual-gateway-accept-data
set interfaces irb unit 200 family inet address 10.200.1.1/24 preferred
set interfaces irb unit 200 family inet address 10.200.1.1/24 virtual-gateway-address
10.200.1.254
set interfaces irb unit 200 family inet6 address abcd::abcd::0010:0200:0001:0001/24 preferred
set interfaces irb unit 200 family inet6 address abcd::abcd::0010:0200:0001:0001/24 virtual-
gateway-address abcd::0010:0200:0001:0254
set interfaces irb unit 200 virtual-gateway-v4-mac 00:10:01:00:01:fe
set interfaces irb unit 200 virtual-gateway-v6-mac 00:10:01:00:00:01:fe
set routing-instances evpn_vlan_aware instance-type mac-vrf
set routing-instances evpn_vlan_aware protocols evpn encapsulation vxlan
set routing-instances evpn_vlan_aware protocols evpn default-gateway no-gateway-community
set routing-instances evpn_vlan_aware vtep-source-interface lo0.0
set routing-instances evpn_vlan_aware service-type vlan-aware
set routing-instances evpn_vlan_aware route-distinguisher 1010:100200
set routing-instances evpn_vlan_aware vrf-target target:666:100200
set routing-instances evpn_vlan_aware vlans V100 vlan-id 100
set routing-instances evpn_vlan_aware vlans V100 interface et-2/0/5.100
set routing-instances evpn_vlan_aware vlans V100 interface ae12.100
set routing-instances evpn_vlan_aware vlans V100 l3-interface irb.100
set routing-instances evpn_vlan_aware vlans V100 vxlan vni 1100
set routing-instances evpn_vlan_aware vlans V200 vlan-id 200
set routing-instances evpn_vlan_aware vlans V200 interface et-2/0/5.200
set routing-instances evpn_vlan_aware vlans V200 interface ae12.200
set routing-instances evpn_vlan_aware vlans V200 l3-interface irb.200
set routing-instances evpn_vlan_aware vlans V200 vxlan vni 1200
set routing-options router-id 10.10.10.10
set routing-options autonomous-system 666
set protocols bgp group vteps type internal
set protocols bgp group vteps local-address 10.10.10.10
set protocols bgp group vteps family evpn signaling
set protocols bgp group vteps neighbor 10.30.30.30
set protocols bgp group vteps neighbor 10.20.20.20
set protocols ospf area 0.0.0.0 interface et-2/0/9.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ae31.0

```

Spine 1

```

set chassis aggregated-devices ethernet device-count 20
set interfaces et-0/0/10 description "To Leaf-1 in ae31"
set interfaces et-0/0/10 ether-options 802.3ad ae31
set interfaces et-0/0/13 description "To Leaf-1 in ae31"
set interfaces et-0/0/13 ether-options 802.3ad ae31
set interfaces ae31 description "To Leaf-1"
set interfaces ae31 aggregated-ether-options lacp active
set interfaces ae31 aggregated-ether-options lacp periodic fast
set interfaces ae31 aggregated-ether-options lacp system-id 30:30:30:30:30:30
set interfaces ae31 unit 0 family inet address 10.40.1.1/24
set interfaces et-0/0/1 description "To Leaf-2"
set interfaces et-0/0/1 unit 0 family inet address 10.30.1.1/24
set interfaces et-0/0/16 description "To Leaf-3"
set interfaces et-0/0/16 unit 0 family inet address 10.40.1.1/24
set interfaces lo0 unit 0 family inet address 10.50.40.1/24 primary
set protocols ospf area 0.0.0.0 interface et-0/0/1.0
set protocols ospf area 0.0.0.0 interface et-0/0/16.0
set protocols ospf area 0.0.0.0 interface ae31.0

```

Layer 3 Gateway in a CRB Overlay Configuration Using the Service Provider Style Interface Configuration

In this sample configuration, you use the service provider style interface configuration to create a Layer 3 gateway, CRB overlay, virtual gateway address, MAC-VRF instance, and VLANs with VLAN IDs. For brevity, we're only providing configurations for one leaf device and one spine.

Leaf 1

```

set chassis aggregated-devices ethernet device-count 20
set interfaces et-2/0/5 description "To PORT-1 --> 1/13"
set interfaces et-2/0/5 flexible-vlan-tagging
set interfaces et-2/0/5 encapsulation extended-vlan-bridge
set interfaces et-2/0/5 unit 100 vlan-id 100
set interfaces et-2/0/5 unit 200 vlan-id 200
set interfaces et-2/0/8 description "To Spine-1 in ae31"
set interfaces et-2/0/8 ether-options 802.3ad ae31
set interfaces et-2/0/11 description "To Spine-1 in ae31"
set interfaces et-2/0/11 ether-options 802.3ad ae31

```

```

set interfaces ae31 description "To Spine-1"
set interfaces ae31 aggregated-ether-options lacp active
set interfaces ae31 aggregated-ether-options lacp periodic fast
set interfaces ae31 aggregated-ether-options lacp system-id 11:11:11:11:11:11
set interfaces ae31 unit 0 family inet address 10.4.1.2/24
set interfaces et-2/0/9 description "To spine-2"
set interfaces et-2/0/9 unit 0 family inet address 10.5.1.2/24
set interfaces et-2/0/12:0 description "To CE-1 in ae12"
set interfaces et-2/0/12:0 ether-options 802.3ad ae12
set interfaces ae12 description "To CE-1"
set interfaces ae12 flexible-vlan-tagging
set interfaces ae12 encapsulation extended-vlan-bridge
set interfaces ae12 esi 00:12:12:12:12:12:12:12:12:12
set interfaces ae12 esi all-active
set interfaces ae12 aggregated-ether-options lacp active
set interfaces ae12 aggregated-ether-options lacp periodic fast
set interfaces ae12 aggregated-ether-options lacp system-id 12:12:12:12:12:12
set interfaces ae12 unit 100 vlan-id 100
set interfaces ae12 unit 200 vlan-id 200
set interfaces lo0 unit 0 family inet address 10.10.10.10/24 primary
set routing-instances evpn_vlan_aware instance-type mac-vrf
set routing-instances evpn_vlan_aware protocols evpn encapsulation vxlan
set routing-instances evpn_vlan_aware vtep-source-interface lo0.0
set routing-instances evpn_vlan_aware service-type vlan-aware
set routing-instances evpn_vlan_aware route-distinguisher 1010:100200
set routing-instances evpn_vlan_aware vrf-target target:666:100200
set routing-instances evpn_vlan_aware vlans V100 interface et-2/0/5.100
set routing-instances evpn_vlan_aware vlans V100 interface ae12.100
set routing-instances evpn_vlan_aware vlans V100 vxlan vni 1100
set routing-instances evpn_vlan_aware vlans V200 interface et-2/0/5.200
set routing-instances evpn_vlan_aware vlans V200 interface ae12.200
set routing-instances evpn_vlan_aware vlans V200 vxlan vni 1200
set routing-options router-id 10.10.10.10
set routing-options autonomous-system 666
set protocols bgp group vteps type internal
set protocols bgp group vteps local-address 10.10.10.10
set protocols bgp group vteps family evpn signaling
set protocols bgp group vteps neighbor 10.1.1.30
set protocols bgp group vteps neighbor 10.5.5.50
set protocols bgp group vteps neighbor 10.4.4.40
set protocols bgp group vteps neighbor 10.2.2.20
set protocols ospf area 0.0.0.0 interface et-2/0/9.0

```



```
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ae31.0
```

Spine 1

```
set chassis aggregated-devices ethernet device-count 20
set interfaces et-0/0/0 description "To CE-3 in ae34"
set interfaces et-0/0/0 ether-options 802.3ad ae34
set interfaces ae34 description "To CE-1"
set interfaces ae34 flexible-vlan-tagging
set interfaces ae34 encapsulation extended-vlan-bridge
set interfaces ae34 esi 00:34:34:34:34:34:34:34:34:34
set interfaces ae34 esi all-active
set interfaces ae34 aggregated-ether-options lacp active
set interfaces ae34 aggregated-ether-options lacp periodic fast
set interfaces ae34 aggregated-ether-options lacp system-id 34:34:34:34:34:34
set interfaces ae34 unit 100 vlan-id 100
set interfaces ae34 unit 200 vlan-id 200
set interfaces et-0/0/10 description "To Leaf-1 in ae31"
set interfaces et-0/0/10 ether-options 802.3ad ae31
set interfaces ae31 description "To Leaf-1"
set interfaces ae31 aggregated-ether-options lacp active
set interfaces ae31 aggregated-ether-options lacp periodic fast
set interfaces ae31 aggregated-ether-options lacp system-id 30:30:30:30:30:30
set interfaces ae31 unit 0 family inet address 10.40.1.1/24
set interfaces et-0/0/11 description "To PORT-5 --> 1/1"
set interfaces et-0/0/11 flexible-vlan-tagging
set interfaces et-0/0/11 speed 40g
set interfaces et-0/0/11 encapsulation extended-vlan-bridge
set interfaces et-0/0/11 unit 100 vlan-id 100
set interfaces et-0/0/11 unit 200 vlan-id 200
set interfaces et-0/0/12 description "To PORT- --> 1/5"
set interfaces et-0/0/12 flexible-vlan-tagging
set interfaces et-0/0/12 encapsulation extended-vlan-bridge
set interfaces et-0/0/12 unit 100 vlan-id 100
set interfaces et-0/0/12 unit 200 vlan-id 200
set interfaces et-0/0/13 description "To Leaf-1 in ae31"
set interfaces et-0/0/13 ether-options 802.3ad ae31
set interfaces et-0/0/1 description "To Leaf-2"
set interfaces et-0/0/1 unit 0 family inet address 10.40.1.1/24
set interfaces et-0/0/16 description "To Leaf-3"
```

```

set interfaces et-0/0/16 unit 0 family inet address 10.30.1.1/24
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 family inet address 10.100.1.1/16 preferred
set interfaces irb unit 100 family inet address 10.100.1.1/16 virtual-gateway-address
10.100.1.254
set interfaces irb unit 100 family inet6 address abcd::0010:0100:0001:0001/16 preferred
set interfaces irb unit 100 family inet6 address abcd::0010:0100:0001:0001/96 virtual-gateway-
address abcd::0010:0100:0001:0254
set interfaces irb unit 100 virtual-gateway-v4-mac 00:10:01:00:01:fe
set interfaces irb unit 100 virtual-gateway-v6-mac 00:10:01:00:02:fe
set interfaces irb unit 200 virtual-gateway-accept-data
set interfaces irb unit 200 family inet address 10.200.1.1/16 preferred
set interfaces irb unit 200 family inet address 10.200.1.1/16 virtual-gateway-address
10.200.1.254
set interfaces irb unit 200 family inet6 address abcd::0010:0200:0001:0001/96 preferred
set interfaces irb unit 200 family inet6 address abcd::0010:0200:0001:0001/96 virtual-gateway-
address abcd::0010:0200:0001:0254
set interfaces irb unit 200 virtual-gateway-v4-mac 00:20:02:00:01:fe
set interfaces irb unit 200 virtual-gateway-v6-mac 00:20:02:00:02:fe
set interfaces lo0 unit 0 family inet address 10.300.40.40/32 primary
set routing-instances evpn_vlan_aware instance-type mac-vrf
set routing-instances evpn_vlan_aware protocols evpn encapsulation vxlan
set routing-instances evpn_vlan_aware vtep-source-interface lo0.0
set routing-instances evpn_vlan_aware service-type vlan-aware
set routing-instances evpn_vlan_aware route-distinguisher 4040:100200
set routing-instances evpn_vlan_aware vrf-target target:666:100200
set routing-instances evpn_vlan_aware vlans V100 vlan-id 100
set routing-instances evpn_vlan_aware vlans V100 interface et-0/0/11.100
set routing-instances evpn_vlan_aware vlans V100 interface et-0/0/12.100
set routing-instances evpn_vlan_aware vlans V100 interface ae34.100
set routing-instances evpn_vlan_aware vlans V100 l3-interface irb.100
set routing-instances evpn_vlan_aware vlans V100 vxlan vni 1100
set routing-instances evpn_vlan_aware vlans V200 vlan-id 200
set routing-instances evpn_vlan_aware vlans V200 interface et-0/0/11.200
set routing-instances evpn_vlan_aware vlans V200 interface et-0/0/12.200
set routing-instances evpn_vlan_aware vlans V200 interface ae34.200
set routing-instances evpn_vlan_aware vlans V200 l3-interface irb.200
set routing-instances evpn_vlan_aware vlans V200 vxlan vni 1200
set routing-options router-id 40.40.40.40
set routing-options autonomous-system 666
set protocols bgp group vteps type internal
set protocols bgp group vteps local-address 10.10.10.10
set protocols bgp group vteps family evpn signaling

```

```
set protocols bgp group vteps neighbor 10.20.30.30
set protocols bgp group vteps neighbor 10.30.50.50
set protocols bgp group vteps neighbor 10.40.10.10
set protocols bgp group vteps neighbor 10.50.20.20
set protocols ospf area 0.0.0.0 interface et-0/0/1.0
set protocols ospf area 0.0.0.0 interface et-0/0/16.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ae31.0
```

Configuring an EVPN-VXLAN Centrally-Routed Bridged Overlay

IN THIS CHAPTER

- [Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric | 771](#)
- [Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric Using MX Routers as Spines | 808](#)

Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric

IN THIS SECTION

- [Requirements | 772](#)
- [Overview | 772](#)
- [Spine 1: Underlay Network Configuration | 776](#)
- [Spine 1: Overlay Network Configuration | 779](#)
- [Spine 1: Access Profile Configuration | 781](#)
- [Spine 2: Full Configuration | 785](#)
- [Leaf 1: Underlay Network Configuration | 787](#)
- [Leaf 1: Overlay Network Configuration | 789](#)
- [Leaf 1: Access Profile Configuration | 791](#)
- [Leaf 2: Full Configuration | 792](#)
- [Leaf 3: Full Configuration | 794](#)
- [Leaf 4: Full Configuration | 795](#)
- [Verification | 796](#)
- [Spine 1 and 2: Route Leaking \(Optional\) | 803](#)
- [Verification with Route Leaking \(Optional\) | 805](#)

Modern data centers rely on an IP fabric. An IP fabric uses BGP-based Ethernet VPN (EVPN) signaling in the control plane and Virtual Extensible LAN (VXLAN) encapsulation in the data plane. This technology provides a standards-based, high-performance solution for Layer 2 (L2) bridging within a VLAN and for routing between VLANs.

In most cases a one-to-one relationship exists between a user VLAN and a VXLAN network identifier (VNI). As a result, the abbreviations VLAN and VXLAN are often used interchangeably. By default, VXLAN encapsulation strips any ingress VLAN tag when received from an access port. The rest of the Ethernet frame is encapsulated in VXLAN for transport across the fabric. At the egress point, the VXLAN encapsulation is stripped and the VLAN tag (if any) is reinserted before the frame is sent to the attached device.

This is an example of an EVPN-VXLAN IP fabric based on a centrally routed bridging (CRB) architecture. Integrated routing and bridging (IRB) interfaces provide Layer 3 (L3) connectivity to servers and VMs that belong to different VLANs and networks. These IRB interfaces serve as the default gateway for inter-VLAN traffic within a fabric. They also serve as destinations that are remote to the fabric, such as in the case of Data Center Interconnect (DCI). In a CRB design you define the IRB interfaces on the spine devices only. Such a design is therefore referred to as being centrally routed, as all routing occurs on the spines.

For an example of an edge-routed bridging (ERB) design, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway" on page 845](#)

For background information about EVPN-VXLAN technology and supported architectures, see [EVPN Primer](#).

Requirements

The original example used the following hardware and software components:

- Two QFX10002 switches (Spine 1 and Spine 2) running Junos OS Release 15.1X53-D30 software.
- Four QFX5100 switches (Leaf 1 through Leaf 4) running Junos OS Release 14.1X53-D30 software.
 - Updated and re-validated using Junos OS Release 20.4R1.
 - See the [hardware summary](#) for a list of supported platforms.

Overview

IN THIS SECTION

- [Topology](#) | [773](#)

In this example, physical servers that support three groups of users (meaning three VLANs) require the following connectivity:

1. Servers A and C should be able to communicate at L2. These servers must share a subnet, and therefore, a VLAN.
2. Servers B and D must be on separate VLANs to isolate broadcast. These servers must be able to communicate at L3.
3. Servers A and C should not be able to communicate with Servers B and D.

To meet these connectivity requirements these protocols and technologies are used:

- EVPN establishes an L2 virtual bridge to connect servers A and C and places servers B and D into their respective VLANs.
- Within the EVPN topology, BGP exchanges route information.
- VXLAN tunnels the L2 frames through the underlying L3 fabric. The use of VXLAN encapsulation preserves the L3 fabric for use by routing protocols.
- IRB interfaces route IP packets between the VLANs.

Again, the IRB interfaces are configured on the spine devices only, for centrally routed bridging (CRB). In this design the spine devices function as L3 gateways for the various servers, VMs, or containerized workloads connected to the access ports on the leaf switches. When the workloads exchange data within their own VLANs, the leaf bridges the traffic. The resulting VXLAN encapsulated traffic is then sent through the spines as underlay (IP) traffic. For intra-VLAN traffic the VXLAN virtual tunnel endpoint (VTEP) functionality of the spine is not used. Intra-VLAN traffic is sent between the VTEPs in the source and destination leaf.

In contrast, inter-VLAN (and inter-fabric) traffic must be routed. This traffic is encapsulated into VXLAN and bridged by the leaf to the spine. The leaves know to send this traffic to the spine because the source, that needs routing targets the destination MAC address of the VLAN's default gateway. In other words, frames that are sent to the IRB's MAC address are forwarded at L2 to the spine device.

At the spine, the L2 encapsulation is removed to accommodate an L3 route lookup in the routing instance associated with the VLAN/IRB. For inter-VLAN traffic the spine determines the destination VLAN and the corresponding VXLAN VNI from its route lookup. The spine then re-encapsulates the traffic and sends it through the underlay to the target leaf or leaves.

Topology

The simple IP Clos topology shown in [Figure 73 on page 774](#) includes two spine switches, four leaf switches, and four servers. Each leaf switch has a connection to each of the spine switches for redundancy.

The server networks are segmented into 3 VLANs, each of which is mapped to a VXLAN Virtual Network Identifier (VNI). VLAN v101 supports servers A and C, and VLANs v102 and v103 support servers B and D, respectively. See [Table 45 on page 775](#) for configuration parameters.

Figure 73: EVPN-VXLAN Topology

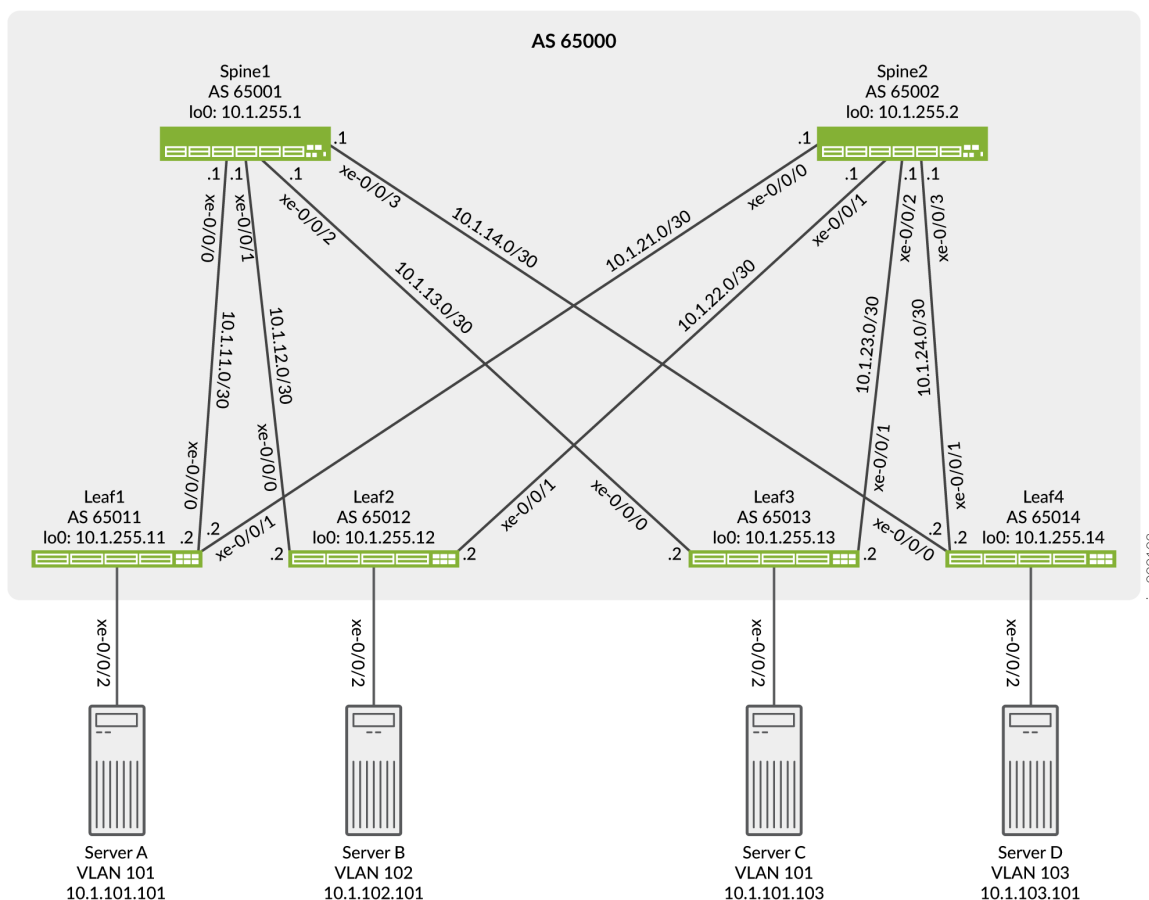


Figure 74: EVPN-VXLAN Logical Topology

The logical topology shows the expected connectivity. In this example one routing instance is used to connect servers A and C using VLAN 101, and one routing instance is used to connect servers B and D using VLANs 102 and 103. The servers are only able to communicate with other servers that are in the same routing instance by default.

Because servers A and C share the same VLAN, the servers communicate at L2. Therefore, the IRB interface is not needed for servers A and C to communicate. We define an IRB interface in the routing

instance as a best practice to enable future L3 connectivity. In contrast, Servers B and D require L3 connectivity through their respective IRB interfaces to communicate, given that these servers are in different VLANs running on unique IP subnets.

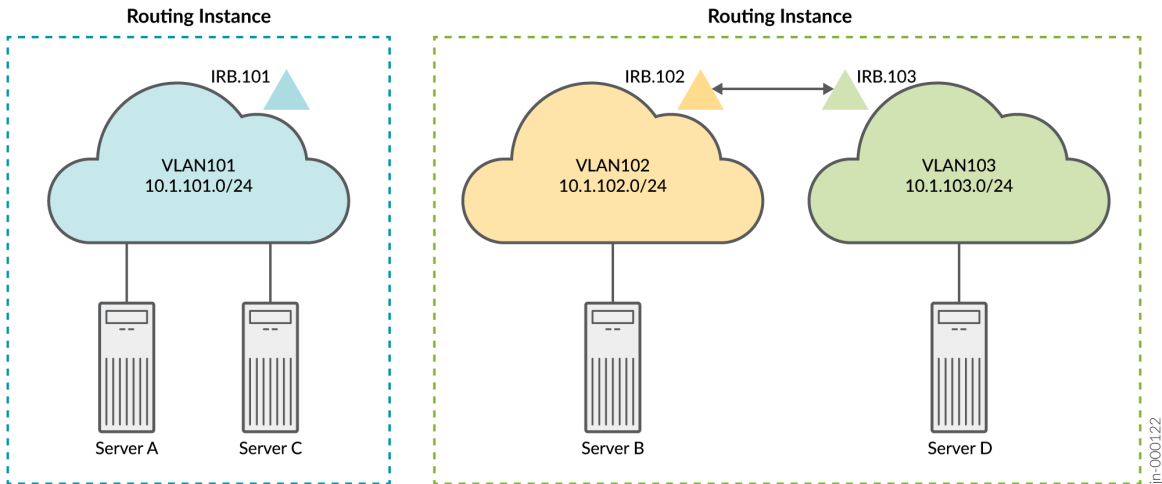


Table 45 on page 775 provides key parameters, including the IRB interfaces, configured for each network. An IRB interface supports each VLAN and routes data packets over the VLAN from the other VLANs.

Table 45: Key VLAN and VXLAN Parameters

Parameters	Servers A and C	Servers B and C
VLAN	v101	v102
		v103
VXLAN VNI	101	102
		103
VLAN ID	101	102
		103

Table 45: Key VLAN and VXLAN Parameters (*Continued*)

Parameters	Servers A and C	Servers B and C
IRB interface	irb.101	irb.102
		irb.103

Keep the following in mind when you configure the parameters in [Table 45 on page 775](#). You must:

- Associate each VLAN with a unique IP subnet, and therefore a unique IRB interface.
- Assign each VLAN a unique VXLAN network identifier (VNI).
- Specify each IRB interface as part of an L3 virtual routing forwarding (VRF) instance, or you can lump the interfaces together in the default switch instance. This example uses VRF instances to enforce separation between the user community (VLANs).
- Include in the configuration of each IRB interface a default gateway address, which you specify with the `virtual-gateway-address` configuration statement at the `[interfaces irb unit logical-unit-number family inet address ip-address]` hierarchy level. Configuring a virtual gateway sets up a redundant default gateway for each IRB interface.

Spine 1: Underlay Network Configuration

IN THIS SECTION

- [CLI Quick Configuration | 776](#)
- [Spine 1: Configure the Underlay Network | 777](#)

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 configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.1/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.12.1/30
set interfaces xe-0/0/2 unit 0 family inet address 10.1.13.1/30
```

```

set interfaces xe-0/0/3 unit 0 family inet address 10.1.14.1/30
set interfaces lo0 unit 0 family inet address 10.1.255.1/32 primary
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.1
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay local-as 65001
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.2 peer-as 65011
set protocols bgp group underlay neighbor 10.1.12.2 peer-as 65012
set protocols bgp group underlay neighbor 10.1.13.2 peer-as 65013
set protocols bgp group underlay neighbor 10.1.14.2 peer-as 65014

```

Spine 1: Configure the Underlay Network

Step-by-Step Procedure

To configure the underlay network on Spine 1:

1. Configure the L3 fabric interfaces.

```

[edit]
user@Spine1# set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.1/30
user@Spine1# set interfaces xe-0/0/1 unit 0 family inet address 10.1.12.1/30
user@Spine1# set interfaces xe-0/0/2 unit 0 family inet address 10.1.13.1/30
user@Spine1# set interfaces xe-0/0/3 unit 0 family inet address 10.1.14.1/30

```

2. Specify an IP address for the loopback interface. This IP address serves as the source IP address in the outer header of VXLAN-encapsulated packets.

```

[edit]
user@Spine1# set interfaces lo0 unit 0 family inet address 10.1.255.1/32 primary

```

3. Configure the routing options. The configuration includes a reference to a load-balancing policy to enable the use of equal cost multiple path (ECMP) routing through the underlay.

```
[edit]
user@Spine1# set routing-options router-id 10.1.255.1
user@Spine1# set routing-options autonomous-system 65000
user@Spine1# set routing-options forwarding-table export load-balancing-policy
```

4. Configure a BGP group for the external BGP (EBGP)-based underlay. Note that multipath is included in the BGP configuration to allow use of multiple equal-cost paths . Normally BGP uses a tie-breaking algorithm that selects the single best path.

```
[edit]
user@Spine1# set protocols bgp group underlay type external
user@Spine1# set protocols bgp group underlay export send-direct
user@Spine1# set protocols bgp group underlay local-as 65001
user@Spine1# set protocols bgp group underlay multipath multiple-as
user@Spine1# set protocols bgp group underlay neighbor 10.1.11.2 peer-as 65011
user@Spine1# set protocols bgp group underlay neighbor 10.1.12.2 peer-as 65012
user@Spine1# set protocols bgp group underlay neighbor 10.1.13.2 peer-as 65013
user@Spine1# set protocols bgp group underlay neighbor 10.1.14.2 peer-as 65014
```

5. Configure the per-packet load-balancing policy.

```
[edit]
user@Spine1# set policy-options policy-statement load-balancing-policy then load-balance per-packet
```

6. Configure a policy to advertise direct interface routes into the underlay. At a minimum you must advertise the loopback interface (lo0) routes into the underlay.

```
[edit]
user@Spine1# set policy-options policy-statement send-direct term 1 from protocol direct
user@Spine1# set policy-options policy-statement send-direct term 1 then accept
```

Spine 1: Overlay Network Configuration

IN THIS SECTION

- [CLI Quick Configuration | 779](#)
- [Configuring the Overlay Network on Spine 1 | 779](#)

CLI Quick Configuration

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

```
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.1.255.1
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn cluster 10.1.1.1
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.1.255.2
set protocols bgp group evpn neighbor 10.1.255.11
set protocols bgp group evpn neighbor 10.1.255.12
set protocols bgp group evpn neighbor 10.1.255.13
set protocols bgp group evpn neighbor 10.1.255.14
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn multicast-mode ingress-replication
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.1:1
set switch-options vrf-target target:65000:1
set switch-options vrf-target auto
```

Configuring the Overlay Network on Spine 1

Step-by-Step Procedure

To configure the overlay network on Spine 1:

1. Configure an internal BGP (IBGP)-based EVPN-VXLAN overlay. Note that the EVPN address family is configured to support advertisement of EVPN routes. In this case we define an overlay peering to Spine 2 for spine-to-spine connectivity. As with the underlay, we also enabled BGP multipath in the overlay.



NOTE: Some IP fabrics use an EBGp-based EVPN-VXLAN overlay. For an example of an IP fabric that uses EBGp for both the underlay and overlay, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway" on page 845](#). Note that the choice of EBGp or IBGP for the overlay does not negatively affect the fabric architecture. Both centrally routed bridging (CRB) and edge-routed bridging (ERB) designs support either overlay type.

```
[edit]
user@Spine1# set protocols bgp group evpn type internal
user@Spine1# set protocols bgp group evpn local-address 10.1.255.1
user@Spine1# set protocols bgp group evpn family evpn signaling
user@Spine1# set protocols bgp group evpn cluster 10.1.1.1
user@Spine1# set protocols bgp group evpn multipath
user@Spine1# set protocols bgp group evpn neighbor 10.1.255.2
user@Spine1# set protocols bgp group evpn neighbor 10.1.255.11
user@Spine1# set protocols bgp group evpn neighbor 10.1.255.12
user@Spine1# set protocols bgp group evpn neighbor 10.1.255.13
user@Spine1# set protocols bgp group evpn neighbor 10.1.255.14
```

2. Configure VXLAN encapsulation for the L2 frames exchanged between the L2 VXLAN VTEPs.

```
[edit]
user@Spine1# set protocols evpn encapsulation vxlan
```

3. Configure the default gateway option `no-gateway-community` for protocols EVPN.

```
[edit]
user@Spine1# set protocols evpn default-gateway no-gateway-community
```



NOTE: When a virtual-gateway-address is used, a VRRP based MAC "00:00:5e:00:01:01" is used on both the spines, so MAC sync is not needed. See *default gateway* for more information.

4. Specify how multicast traffic is replicated in the fabric.

```
[edit]
user@Spine1# set protocols evpn multicast-mode ingress-replication
```

5. Configure default routing instance options (virtual switch type).

```
[edit]
user@Spine1# set switch-options vtep-source-interface lo0.0
user@Spine1# set switch-options route-distinguisher 10.1.255.1:1
user@Spine1# set switch-options vrf-target target:65000:1
user@Spine1# set switch-options vrf-target auto
```

Spine 1: Access Profile Configuration

IN THIS SECTION

- [CLI Quick Configuration | 781](#)
- [Configuring the Access Profiles for Spine 1 | 783](#)

CLI Quick Configuration

The access profile or access port configuration involves the settings needed to attach server workloads, BMSs or VMs, to the access (leaf) switches. This step involves definition of the device's VLAN along with the routing instance and IRB configuration that provide user isolation and L3 routing, respectively.

Because this is an example of a centrally routed bridging (CRB) fabric, the routing instances and integrated routing and bridging (IRB) interfaces are defined on the spine devices only. The leaf devices in a CRB fabric have only L2 VXLAN functionality.

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



NOTE: When [proxy-macip-advertisement](#) is enabled, the L3 gateway advertises MAC and IP routes (MAC+IP type 2 routes) on behalf of L2 VXLAN gateways in EVPN-VXLAN

networks. This behavior is not supported on EVPN-MPLS. Starting in Junos OS Release 20.2R2, the following warning message appears when you enable proxy-macip-advertisement:

WARNING: Only EVPN VXLAN supports proxy-macip-advertisement configuration,

The message appears when you change your configuration, save your configuration, or use the show command to display your configuration

```
set interfaces irb unit 101 proxy-macip-advertisement
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.1/24 virtual-gateway-address
10.1.101.254
set interfaces irb unit 102 proxy-macip-advertisement
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 family inet address 10.1.102.1/24 virtual-gateway-address
10.1.102.254
set interfaces irb unit 103 proxy-macip-advertisement
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 family inet address 10.1.103.1/24 virtual-gateway-address
10.1.103.254
set protocols evpn vni-options vni 101 vrf-target target:65000:101
set protocols evpn vni-options vni 102 vrf-target target:65000:102
set protocols evpn vni-options vni 103 vrf-target target:65000:103
set protocols evpn extended-vni-list 101
set protocols evpn extended-vni-list 102
set protocols evpn extended-vni-list 103
set routing-instances serverAC instance-type vrf
set routing-instances serverAC interface irb.101
set routing-instances serverAC route-distinguisher 10.1.255.1:13
set routing-instances serverAC vrf-target target:65000:13
set routing-instances serverBD instance-type vrf
set routing-instances serverBD interface irb.102
set routing-instances serverBD interface irb.103
set routing-instances serverBD route-distinguisher 10.1.255.1:24
set routing-instances serverBD vrf-target target:65000:24
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
```

```
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103
```

Configuring the Access Profiles for Spine 1

Step-by-Step Procedure

To configure profiles for the server networks:

1. Configure the IRB interfaces that support routing among VLANs 101, 102, and 103.

```
[edit]
user@Spine1# set interfaces irb unit 101 proxy-macip-advertisement
user@Spine1# set interfaces irb unit 101 virtual-gateway-accept-data
user@Spine1# set interfaces irb unit 101 family inet address 10.1.101.1/24 virtual-gateway-
address 10.1.101.254
user@Spine1# set interfaces irb unit 102 proxy-macip-advertisement
user@Spine1# set interfaces irb unit 102 virtual-gateway-accept-data
user@Spine1# set interfaces irb unit 102 family inet address 10.1.102.1/24 virtual-gateway-
address 10.1.102.254
user@Spine1# set interfaces irb unit 103 proxy-macip-advertisement
user@Spine1# set interfaces irb unit 103 virtual-gateway-accept-data
user@Spine1# set interfaces irb unit 103 family inet address 10.1.103.1/24 virtual-gateway-
address 10.1.103.254
```

2. Specify which virtual network identifiers (VNIs) are included in the EVPN-VXLAN domain.

```
[edit]
user@Spine1# set protocols evpn extended-vni-list 101
user@Spine1# set protocols evpn extended-vni-list 102
user@Spine1# set protocols evpn extended-vni-list 103
```

3. Configure a route target for each VNI.

```
[edit]
user@Spine1# set protocols evpn vni-options vni 101 vrf-target target:65000:101
user@Spine1# set protocols evpn vni-options vni 102 vrf-target target:65000:102
user@Spine1# set protocols evpn vni-options vni 103 vrf-target target:65000:103
```




NOTE: In the original configuration, the spine devices run Junos OS Release 15.1X53-D30, and the leaf devices run 14.1X53-D30. In these software releases, when you include the `vrf-target` configuration statement in the `[edit protocols evpn vni-options vni]` hierarchy level, you must also include the `export` option. Note that later Junos OS releases do not require this option. As a result the configurations in this updated example omit the `export` option.

4. Configure the routing instance for servers A and C.

```
[edit]
user@Spine1# set routing-instances serverAC instance-type vrf
user@Spine1# set routing-instances serverAC interface irb.101
user@Spine1# set routing-instances serverAC route-distinguisher 10.1.255.1:13
user@Spine1# set routing-instances serverAC vrf-target target:65000:13
```

5. Configure the routing instance for servers B and D.

```
[edit]
user@Spine1# set routing-instances serverBD instance-type vrf
user@Spine1# set routing-instances serverBD interface irb.102
user@Spine1# set routing-instances serverBD interface irb.103
user@Spine1# set routing-instances serverBD route-distinguisher 10.1.255.1:24
user@Spine1# set routing-instances serverBD vrf-target target:65000:24
```

6. Configure VLANs v101, v102, and v103, and associate the corresponding VNIs and IRB interfaces with each VLAN.

```
[edit]
user@Spine1# set vlans v101 vlan-id 101
user@Spine1# set vlans v101 l3-interface irb.101
user@Spine1# set vlans v101 vxlan vni 101
user@Spine1# set vlans v102 vlan-id 102
user@Spine1# set vlans v102 l3-interface irb.102
user@Spine1# set vlans v102 vxlan vni 102
user@Spine1# set vlans v103 vlan-id 103
user@Spine1# set vlans v103 l3-interface irb.103
user@Spine1# set vlans v103 vxlan vni 103
```

Spine 2: Full Configuration

IN THIS SECTION

- [CLI Quick Configuration | 785](#)

CLI Quick Configuration

The Spine 2 configuration is similar to that of Spine 1, so we provide the full configuration instead of a step-by-step configuration. To quickly configure this example, copy the following commands, paste the commands into a text file, remove any line breaks, change any details necessary to match your configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` from configuration mode.

```
set interfaces xe-0/0/0 unit 0 family inet address 10.1.21.1/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.22.1/30
set interfaces xe-0/0/2 unit 0 family inet address 10.1.23.1/30
set interfaces xe-0/0/3 unit 0 family inet address 10.1.24.1/30
set interfaces irb unit 101 proxy-macip-advertisement
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.2/24 virtual-gateway-address
10.1.101.254
set interfaces irb unit 102 proxy-macip-advertisement
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 family inet address 10.1.102.2/24 virtual-gateway-address
10.1.102.254
set interfaces irb unit 103 proxy-macip-advertisement
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 family inet address 10.1.103.2/24 virtual-gateway-address
10.1.103.254
set interfaces lo0 unit 0 family inet address 10.1.255.2/32 primary
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-instances serverAC instance-type vrf
set routing-instances serverAC interface irb.101
set routing-instances serverAC route-distinguisher 10.1.255.1:13
set routing-instances serverAC vrf-target target:65000:13
set routing-instances serverBD instance-type vrf
```

```

set routing-instances serverBD interface irb.102
set routing-instances serverBD interface irb.103
set routing-instances serverBD route-distinguisher 10.1.255.1:24
set routing-instances serverBD vrf-target target:65000:24
set routing-options router-id 10.1.255.2
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay local-as 65002
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.21.2 peer-as 65011
set protocols bgp group underlay neighbor 10.1.22.2 peer-as 65012
set protocols bgp group underlay neighbor 10.1.23.2 peer-as 65013
set protocols bgp group underlay neighbor 10.1.24.2 peer-as 65014
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.1.255.2
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn cluster 10.2.2.2
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.1.255.1
set protocols bgp group evpn neighbor 10.1.255.11
set protocols bgp group evpn neighbor 10.1.255.12
set protocols bgp group evpn neighbor 10.1.255.13
set protocols bgp group evpn neighbor 10.1.255.14
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:65000:101
set protocols evpn vni-options vni 102 vrf-target target:65000:102
set protocols evpn vni-options vni 103 vrf-target target:65000:103
set protocols evpn extended-vni-list 101
set protocols evpn extended-vni-list 102
set protocols evpn extended-vni-list 103
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.2:1
set switch-options vrf-target target:65000:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102

```

```

set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Leaf 1: Underlay Network Configuration

IN THIS SECTION

- [CLI Quick Configuration | 787](#)
- [Configuring the Underlay Network for Leaf 1 | 788](#)

CLI Quick Configuration

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

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.21.2/30
set interfaces lo0 unit 0 family inet address 10.1.255.11/32 primary
set routing-options router-id 10.1.255.11
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay local-as 65011
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept

```

Configuring the Underlay Network for Leaf 1

Step-by-Step Procedure

To configure the underlay network for Leaf 1:

1. Configure the L3 interfaces.

```
[edit]
user@Leaf1# set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.2/30
user@Leaf1# set interfaces xe-0/0/1 unit 0 family inet address 10.1.21.2/30
```

2. Specify an IP address for the loopback interface. This IP address serves as the source IP address in the outer header of any VXLAN encapsulated packets.

```
[edit]
user@Leaf1# set interfaces lo0 unit 0 family inet address 10.1.255.11/32 primary
```

3. Set the routing options.

```
[edit]
user@Leaf1# set routing-options router-id 10.1.255.11
user@Leaf1# set routing-options autonomous-system 65000
user@Leaf1# set routing-options forwarding-table export load-balancing-policy
```

4. Configure an external BGP (EBGP) group that includes the spines as peers to handle underlay routing.

```
[edit]
user@Leaf1# set protocols bgp group underlay type external
user@Leaf1# set protocols bgp group underlay export send-direct
user@Leaf1# set protocols bgp group underlay local-as 65011
user@Leaf1# set protocols bgp group underlay multipath multiple-as
user@Leaf1# set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
user@Leaf1# set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002
```

5. Configure a policy that spreads traffic across multiple paths between the Juniper Networks switches.

```
[edit]
user@Leaf1# set policy-options policy-statement load-balancing-policy then load-balance per-
packet
```

6. Configure a policy to advertise direct interface routes. At a minimum the underlay must have full reachability to the device loopback addresses.

```
[edit]
user@Leaf1# set policy-options policy-statement send-direct term 1 from protocol direct
user@Leaf1# set policy-options policy-statement send-direct term 1 then accept
```

Leaf 1: Overlay Network Configuration

IN THIS SECTION

- [CLI Quick Configuration | 789](#)
- [Configuring the Overlay Network for Leaf 1 | 790](#)

CLI Quick Configuration

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

```
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.1.255.11
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.1.255.1
set protocols bgp group evpn neighbor 10.1.255.2
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.11:1
```

```
set switch-options vrf-target target:65000:1
set switch-options vrf-target auto
```

Configuring the Overlay Network for Leaf 1

Step-by-Step Procedure

To configure the overlay network for Leaf 1:

1. Configure an internal BGP (IBGP) group for the EVPN-VXLAN overlay network.

```
[edit]
user@Leaf1# set protocols bgp group evpn type internal
user@Leaf1# set protocols bgp group evpn local-address 10.1.255.11
user@Leaf1# set protocols bgp group evpn family evpn signaling
user@Leaf1# set protocols bgp group evpn multipath
user@Leaf1# set protocols bgp group evpn neighbor 10.1.255.1
user@Leaf1# set protocols bgp group evpn neighbor 10.1.255.2
```

2. Configure VXLAN encapsulation for the data packets exchanged between the EVPN neighbors.

```
[edit]
user@Leaf1# set protocols evpn encapsulation vxlan
```

3. Specify how multicast traffic is replicated in the EVPN-VXLAN environment.

```
[edit]
user@Leaf1# set protocols evpn multicast-mode ingress-replication
```

4. Configure the default routing instance options (virtual switch type).

```
[edit]
user@Leaf1# set switch-options vtep-source-interface lo0.0
user@Leaf1# set switch-options route-distinguisher 10.1.255.11:1
user@Leaf1# set switch-options vrf-target target:65000:1
user@Leaf1# set switch-options vrf-target auto
```

Leaf 1: Access Profile Configuration

IN THIS SECTION

- [CLI Quick Configuration | 791](#)
- [Configuring the Access Profile for Leaf 1 | 791](#)

CLI Quick Configuration

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

```
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members 101
set protocols evpn vni-options vni 101 vrf-target target:65000:101
set protocols evpn extended-vni-list 101
set vlans v101 vlan-id 101
set vlans v101 vxlan vni 101
```

Configuring the Access Profile for Leaf 1

Step-by-Step Procedure

To configure the profile for the server network:

1. Configure an L2 Ethernet interface for the connection with the physical server. This interface is associated with VLAN 101. In this example the access interface is configured as a trunk to support VLAN tagging. Untagged access interfaces are also supported.

```
[edit]
user@Leaf1# set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
user@Leaf1# set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members 101
```


2. Configure a route target for the virtual network identifier (VNI).

```
[edit]
user@Leaf1# set protocols evpn vni-options vni 101 vrf-target target:65000:101
```



NOTE: In the original configuration, the spine devices run Junos OS Release 15.1X53-D30, and the leaf devices run 14.1X53-D30. In these software releases, when you include the vrf-target configuration statement in the [edit protocols evpn vni-options vni] hierarchy level, you must also include the export option. Note that later Junos OS releases do not require this option, as reflected in the updated configurations used in this example.

3. Specify which VNIs are included in the EVPN-VXLAN domain.

```
[edit]
user@Leaf1# set protocols evpn extended-vni-list 101
```

4. Configure VLAN v101. The VLAN is mapped to the same VXLAN VNI that you configured on the spine devices. Note that the L3 integrated routing and bridging (IRB) interface is not specified on the leaf devices. This is because in centrally routed bridging (CRB) the leaves perform L2 bridging only.

```
[edit]
user@Leaf1# set vlans v101 vlan-id 101
user@Leaf1# set vlans v101 vxlan vni 101
```

Leaf 2: Full Configuration

IN THIS SECTION

- [CLI Quick Configuration | 792](#)

CLI Quick Configuration

The Leaf 2 configuration is similar to that of Leaf 1, so we provide the full configuration instead of a step-by-step configuration. To quickly configure this example, copy the following commands, paste the

commands into a text file, remove any line breaks, change any details necessary to match your configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter commit from configuration mode.

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.12.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.22.2/30
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members 102
set interfaces lo0 unit 0 family inet address 10.1.255.12/32 primary
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.12
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay local-as 65012
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay as-override
set protocols bgp group underlay neighbor 10.1.12.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.22.1 peer-as 65002
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.1.255.12
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.1.255.1
set protocols bgp group evpn neighbor 10.1.255.2
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 102 vrf-target target:65000:102
set protocols evpn extended-vni-list 102
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.12:1
set switch-options vrf-target target:65000:1
set switch-options vrf-target auto
set vlans v102 vlan-id 102
set vlans v102 vxlan vni 102

```

Leaf 3: Full Configuration

IN THIS SECTION

- [CLI Quick Configuration | 794](#)

CLI Quick Configuration

The Leaf 3 configuration is similar to that of Leaf 1, so we provide the full configuration instead of a step-by-step configuration. To quickly configure this example, copy the following commands, paste the commands into a text file, remove any line breaks, change any details necessary to match your configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` from configuration mode.

```
set interfaces xe-0/0/0 unit 0 family inet address 10.1.13.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.23.2/30
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members 101
set interfaces lo0 unit 0 family inet address 10.1.255.13/32 primary
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.13
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay local-as 65013
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.13.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.23.1 peer-as 65002
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.1.255.13
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.1.255.1
set protocols bgp group evpn neighbor 10.1.255.2
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
```

```

set protocols evpn vni-options vni 101 vrf-target target:65000:101
set protocols evpn extended-vni-list 101
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.13:1
set switch-options vrf-target target:65000:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 vxlan vni 101

```

Leaf 4: Full Configuration

IN THIS SECTION

- [CLI Quick Configuration | 795](#)

CLI Quick Configuration

The Leaf 4 configuration is similar to that of Leaf 1, so we provide the full configuration instead of step-by-step configuration. To quickly configure this example, copy the following commands, paste the commands into a text file, remove any line breaks, change any details necessary to match your configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` from configuration mode.

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.14.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.24.2/30
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members 103
set interfaces lo0 unit 0 family inet address 10.1.255.14/32 primary
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.14
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay local-as 65014
set protocols bgp group underlay multipath multiple-as

```

```

set protocols bgp group underlay neighbor 10.1.14.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.24.1 peer-as 65002
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.1.255.14
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.1.255.1
set protocols bgp group evpn neighbor 10.1.255.2
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 103 vrf-target target:65000:103
set protocols evpn extended-vni-list 103
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.14:1
set switch-options vrf-target target:65000:1
set switch-options vrf-target auto
set vlans v103 vlan-id 103
set vlans v103 vxlan vni 103

```

Verification

IN THIS SECTION

- [Verify the IRB Interfaces | 796](#)
- [Verifying the Routing Instances | 798](#)
- [Verifying Dynamic MAC Addresses Learning | 800](#)
- [Verifying Routes in the Routing Instances | 801](#)
- [Verify Connectivity | 802](#)

Confirm that the integrated routing and bridging (IRB) interfaces are working properly:

Verify the IRB Interfaces

Purpose

Verify the configuration of the IRB interfaces on Spine 1 and Spine 2.

Action

From operational mode, enter the `show interfaces irb` command.

```

user@Spine1> show interfaces irb
Physical interface: irb, Enabled, Physical link is Up
  Interface index: 640, SNMP ifIndex: 505
  Type: Ethernet, Link-level type: Ethernet, MTU: 1514
  Device flags   : Present Running
  Interface flags: SNMP-Traps
  Link type      : Full-Duplex
  Link flags     : None
  Current address: 02:05:86:71:57:00, Hardware address: 02:05:86:71:57:00
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

Logical interface irb.101 (Index 558) (SNMP ifIndex 583)
  Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
  Bandwidth: 1Gbps
  Routing Instance: default-switch Bridging Domain: v101
  Input packets : 7
  Output packets: 13
  Protocol inet, MTU: 1514
  Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 2, Curr new hold cnt: 0, NH drop
  cnt: 0
    Flags: Sendbcast-pkt-to-re, Is-Primary
    Addresses, Flags: Is-Default Is-Preferred Is-Primary
      Destination: 10.1.101/24, Local: 10.1.101.1, Broadcast: 10.1.101.255
      Destination: 10.1.101/24, Local: 10.1.101.254, Broadcast: 10.1.101.255

Logical interface irb.102 (Index 582) (SNMP ifIndex 584)
  Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
  Bandwidth: 1Gbps
  Routing Instance: default-switch Bridging Domain: v102
  Input packets : 2
  Output packets: 6
  Protocol inet, MTU: 1514
  Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 1, Curr new hold cnt: 0, NH drop
  cnt: 0
    Flags: Sendbcast-pkt-to-re
    Addresses, Flags: Is-Preferred Is-Primary

```

```

Destination: 10.1.102/24, Local: 10.1.102.1, Broadcast: 10.1.102.255
Destination: 10.1.102/24, Local: 10.1.102.254, Broadcast: 10.1.102.255

Logical interface irb.103 (Index 580) (SNMP ifIndex 585)
  Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
  Bandwidth: 1Gbps
  Routing Instance: default-switch Bridging Domain: v103
  Input packets : 2
  Output packets: 6
  Protocol inet, MTU: 1514
  Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 1, Curr new hold cnt: 0, NH drop
  cnt: 0
  Flags: Sendbcast-pkt-to-re, Is-Primary
  Addresses, Flags: Is-Default Is-Preferred Is-Primary
    Destination: 10.1.103/24, Local: 10.1.103.1, Broadcast: 10.1.103.255
    Destination: 10.1.103/24, Local: 10.1.103.254, Broadcast: 10.1.103.255

```

Meaning

The sample output from Spine 1 verifies the following:

- IRB interfaces irb.101, irb.102, and irb.103 are configured.
- The physical interface upon which the IRB interfaces are configured is up and running.
- Each IRB interface is properly mapped to its respective VLAN.
- The configuration of each IRB interface correctly reflects the IP address and destination (virtual gateway address) assigned to it.

Verifying the Routing Instances

Purpose

Verify that the routing instances for servers A and B, and for servers C and D, are properly configured on Spine 1 and Spine 2.

Action

From operational mode, enter the `show route instance routing-instance-name extensive` command routing instances `serverAC` and `serverBD`.

```
user@Spine1> show route instance serverAC extensive
serverAC:
  Router ID: 10.1.101.1
  Type: vrf                State: Active
  Interfaces:
    irb.101
  Route-distinguisher: 10.1.255.1:12
  Vrf-import: [ __vrf-import-serverAC-internal__ ]
  Vrf-export: [ __vrf-export-serverAC-internal__ ]
  Vrf-import-target: [ target:65000:12 ]
  Vrf-export-target: [ target:65000:12 ]
  Fast-reroute-priority: low
  Tables:
    serverAC.inet.0      : 3 routes (3 active, 0 holddown, 0 hidden)
    serverAC.iso.0       : 0 routes (0 active, 0 holddown, 0 hidden)
    serverAC.inet6.0     : 1 routes (1 active, 0 holddown, 0 hidden)
    serverAC.mdt.0       : 0 routes (0 active, 0 holddown, 0 hidden)
```

```
user@Spine1> show route instance serverBD extensive
serverBD:
  Router ID: 10.1.102.1
  Type: vrf                State: Active
  Interfaces:
    irb.102
    irb.103
  Route-distinguisher: 10.1.255.1:34
  Vrf-import: [ __vrf-import-serverBD-internal__ ]
  Vrf-export: [ __vrf-export-serverBD-internal__ ]
  Vrf-import-target: [ target:65000:34 ]
  Vrf-export-target: [ target:65000:34 ]
  Fast-reroute-priority: low
  Tables:
    serverBD.inet.0      : 6 routes (6 active, 0 holddown, 0 hidden)
    serverBD.iso.0       : 0 routes (0 active, 0 holddown, 0 hidden)
```



```
serverBD.inet6.0      : 1 routes (1 active, 0 holddown, 0 hidden)
serverBD.mdt.0       : 0 routes (0 active, 0 holddown, 0 hidden)
```

Meaning

In the sample output from Spine 1, the routing instances for servers A and C and for servers B and D show the loopback interface and IRB interfaces that are associated with each group. The output also shows the actual route distinguisher, virtual routing and forwarding (VRF) import, and VRF export policy configuration.

Verifying Dynamic MAC Addresses Learning

Purpose

Verify that for VLANs v101, v102, and v103, a dynamic MAC address is installed in the Ethernet switching tables on all leaves.

Action

From operational mode, enter the `show ethernet-switching table` command.

```
user@Leaf1> show ethernet-switching table
```

```
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, 0 - ovsdb MAC)
```

```
Ethernet switching table : 5 entries, 5 learned
```

```
Routing instance : default-switch
```

Vlan name	MAC address	MAC flags	Logical interface	SVLBNH/ VENH Index	Active source
v101	00:00:5e:00:01:01	DR	esi.1746		
05:00:00:fd:e8:00:00:00:65:00					
v101	00:50:56:93:87:58	D	xe-0/0/2.0		
v101	00:50:56:93:ab:f6	D	vtep.32770		
10.1.255.13					
v101	02:05:86:71:27:00	D	vtep.32771		
10.1.255.1					
v101	02:05:86:71:5f:00	D	vtep.32769		
10.1.255.2					

Meaning

The sample output from Leaf 1 indicates that it has learned the MAC address 00:00:5e:00:01:01 for its virtual gateway (IRB). This is the MAC address that the attached servers use to reach their default gateway. Because the same virtual IP/MAC is configured on both spines, the virtual IP is treated as an ESI LAG to support active forwarding to both spines without the risk of packet loops. The output also indicates that Leaf 1 learned the IRB MAC addresses for Spine 1 and Spine 2, which function as VTEPs.

Verifying Routes in the Routing Instances

Purpose

Verify that the correct routes are in the routing instances.

Action

From operational mode, enter the `show route table routing-instance-name.inet.0` command.

```
user@Spine1> show route table serverAC.inet.0

serverAC.inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.1.101.0/24      *[Direct/0] 2d 01:34:44
                  > via irb.101
10.1.101.1/32     *[Local/0] 2d 01:34:44
                  Local via irb.101
10.1.101.254/32   *[Local/0] 2d 01:34:44
                  Local via irb.101

user@Spine1> show route table serverBD.inet.0

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

10.1.102.0/24     *[Direct/0] 2d 01:34:51
                  > via irb.102
10.1.102.1/32     *[Local/0] 2d 01:34:51
                  Local via irb.102
10.1.102.254/32   *[Local/0] 2d 01:34:51
                  Local via irb.102
```

```

10.1.103.0/24      *[Direct/0] 2d 01:34:51
                   > via irb.103
10.1.103.1/32     *[Local/0] 2d 01:34:51
                   Local via irb.103
10.1.103.254/32   *[Local/0] 2d 01:34:51
                   Local via irb.103

```

Meaning

The sample output from Spine 1 indicates that the routing instance for servers A and C has the IRB interface routes associated with VLAN 101 and that the routing instance for server B and D has the IRB interfaces routes associated with VLANs 102 and 103.

Based on the routes in each table, it is clear that servers A and C in VLAN 101 cannot reach servers in C and D in VLANs 102 or 103. The output also shows that the common table that houses routes for servers B and D allows L3 communications through their IRB interfaces.

Verify Connectivity

Purpose

Verify that servers A and C can ping each other and servers B and D can ping each other.

Action

Run the ping command from the servers.

```

user@serverA> ping 10.1.101.103 count 2
PING 10.1.101.103 (10.1.101.103): 56 data bytes
64 bytes from 10.1.101.103: icmp_seq=0 ttl=64 time=103.749 ms
64 bytes from 10.1.101.103: icmp_seq=1 ttl=64 time=116.325 ms

--- 10.1.101.103 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 103.749/110.037/116.325/6.288 ms

user@serverB> ping 10.1.103.101 count 2
PING 10.1.103.101 (10.1.103.101): 56 data bytes
64 bytes from 10.1.103.101: icmp_seq=0 ttl=63 time=103.346 ms
64 bytes from 10.1.103.101: icmp_seq=1 ttl=63 time=102.355 ms

```

```
--- 10.1.103.101 ping statistics ---  
2 packets transmitted, 2 packets received, 0% packet loss  
round-trip min/avg/max/stddev = 102.355/102.851/103.346/0.495 ms
```

Meaning

The sample output shows that server A can ping server C, and server B can ping server D. Servers A and C should not be able to ping servers B and D, and servers B and D should not be able to ping servers A and C.

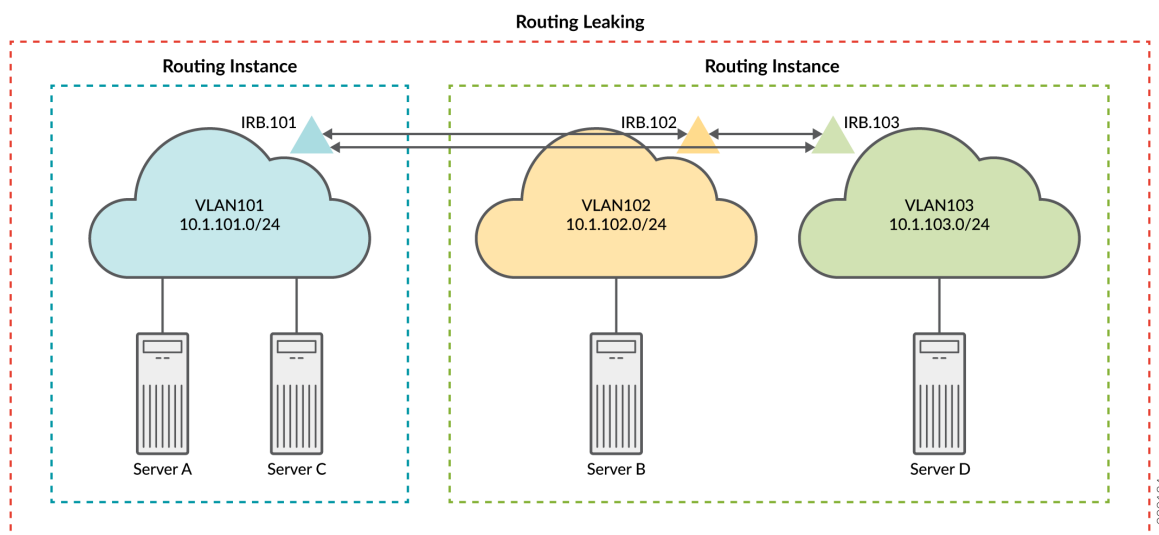
Spine 1 and 2: Route Leaking (Optional)

IN THIS SECTION

- [CLI Quick Configuration | 804](#)

Referring to [Figure 74 on page 774](#), recall that you configured three VLANs and two routing instances to provide connectivity for servers A and C in VLAN 101 and for servers B and D in VLANs 102 and 103, respectively. In this section you modify the configuration to leak routes between the two routing instances. [Figure 75 on page 804](#) shows the resulting logical connectivity after the integrated routing and bridging (IRB) routes are leaked.

Figure 75: EVPN-VXLAN Logical Topology with Route Leaking



With your routing information base (RIB) group modifications, you can expect the servers in VLAN 101 to reach the servers in both VLANs 102 and 103 using L3 connectivity.

CLI Quick Configuration

At this stage you have deployed a CRB-based EVPN fabric and have confirmed expected connectivity. That is, servers A and C can communicate at L2. Servers B and D (on VLANs 102 and 103, respectively) communicate through IRB routing in their shared routing instance. What if you want all servers to be able to ping each other? One option to solve this problem is to leak the routes between the routing instances. See [auto-export](#) for more information about leaking routes between virtual routing and forwarding (VRF) instances. To quickly configure this example, copy the following commands, paste the commands into a text file, remove any line breaks, change any details necessary to match your configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter commit from configuration mode.

```
set policy-options policy-statement serverAC_vrf_imp term 1 from community serverBD
set policy-options policy-statement serverAC_vrf_imp term 1 then accept
set policy-options policy-statement serverBD_vrf_imp term 1 from community serverAC
set policy-options policy-statement serverBD_vrf_imp term 1 then accept
set policy-options community serverAC members target:65000:13
set policy-options community serverBD members target:65000:24
set routing-instances serverAC routing-options auto-export
set routing-instances serverAC vrf-import serverAC_vrf_imp
```

```
set routing-instances serverBD routing-options auto-export
set routing-instances serverBD vrf-import serverBD_vrf_imp
```

Verification with Route Leaking (Optional)

IN THIS SECTION

- [Verifying Routes with Route Leaking \(Optional\) | 805](#)
- [Verifying Connectivity with Route Leaking \(Optional\) | 807](#)

Verifying Routes with Route Leaking (Optional)

Purpose

Verify that the correct routes are in the routing instances.

Action

From operational mode, enter the `show route table routing-instance-name.inet.0` command.

```
user@Spine1> show route table serverAC.inet.0

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

10.1.101.0/24    *[Direct/0] 2d 02:18:50
                 > via irb.101
10.1.101.1/32   *[Local/0] 2d 02:18:50
                 Local via irb.101
10.1.101.254/32 *[Local/0] 2d 02:18:50
                 Local via irb.101
10.1.102.0/24   *[Direct/0] 00:31:21
                 > via irb.102
10.1.102.1/32   *[Local/0] 00:31:21
                 Local via irb.102
10.1.102.254/32 *[Local/0] 00:31:21
                 Local via irb.102
10.1.103.0/24   *[Direct/0] 00:31:21
```

```

                > via irb.103
10.1.103.1/32    *[Local/0] 00:31:21
                Local via irb.103
10.1.103.254/32 *[Local/0] 00:31:21
                Local via irb.103

user@Spine1> show route table serverBD.inet.0

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

10.1.101.0/24    *[Direct/0] 00:32:00
                > via irb.101
10.1.101.1/32    *[Local/0] 00:32:00
                Local via irb.101
10.1.101.254/32  *[Local/0] 00:32:00
                Local via irb.101
10.1.102.0/24    *[Direct/0] 2d 02:19:29
                > via irb.102
10.1.102.1/32    *[Local/0] 2d 02:19:29
                Local via irb.102
10.1.102.254/32  *[Local/0] 2d 02:19:29
                Local via irb.102
10.1.103.0/24    *[Direct/0] 2d 02:19:29
                > via irb.103
10.1.103.1/32    *[Local/0] 2d 02:19:29
                Local via irb.103
10.1.103.254/32  *[Local/0] 2d 02:19:29
                Local via irb.103

```

Meaning

The sample output from Spine 1 indicates that both routing instances now have the integrated routing and bridging (IRB) interface routes associated with all three VLANs. Because you copied routes between the instance tables, the net result is the same as if you configured all three VLANs in a common routing instance. Thus, you can expect full L3 connectivity between servers in all three VLANs.

Verifying Connectivity with Route Leaking (Optional)

Purpose

Verify that servers A and C can ping servers B and D.

Action

Run the ping command from the servers.

```
user@serverA> ping 10.1.102.101 count 2
PING 10.1.102.101 (10.1.102.101): 56 data bytes
64 bytes from 10.1.102.101: icmp_seq=0 ttl=63 time=102.448 ms
64 bytes from 10.1.102.101: icmp_seq=1 ttl=63 time=102.384 ms

--- 10.1.102.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 102.384/102.416/102.448/0.032 ms

user@serverA> ping 10.1.103.101 count 2
PING 10.1.103.101 (10.1.103.101): 56 data bytes
64 bytes from 10.1.103.101: icmp_seq=0 ttl=63 time=103.388 ms
64 bytes from 10.1.103.101: icmp_seq=1 ttl=63 time=102.623 ms

--- 10.1.103.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 102.623/103.006/103.388/0.382 ms

user@serverC> ping 10.1.102.101 count 2
PING 10.1.102.101 (10.1.102.101): 56 data bytes
64 bytes from 10.1.102.101: icmp_seq=0 ttl=63 time=167.580 ms
64 bytes from 10.1.102.101: icmp_seq=1 ttl=63 time=168.075 ms

--- 10.1.102.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 167.580/167.827/168.075/0.248 ms

user@serverC> ping 10.1.103.101 count 2
PING 10.1.103.101 (10.1.103.101): 56 data bytes
64 bytes from 10.1.103.101: icmp_seq=0 ttl=63 time=103.673 ms
64 bytes from 10.1.103.101: icmp_seq=1 ttl=63 time=115.090 ms
```



```

--- 10.1.103.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 103.673/109.382/115.090/5.709 ms

```

Meaning

The sample output shows that server A can ping server B and server D. It also shows that server C can ping server B and server D. This confirms the expected full connectivity among the servers and their VLANs.

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
15.1X53-D30	Starting with Junos OS Release 15.1X53-D30, you can use integrated routing and bridging (IRB) interfaces to route between VLANs using Virtual Extensible LAN (VXLAN) encapsulation.

RELATED DOCUMENTATION

[Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric Using MX Routers as Spines | 808](#)

[Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway | 845](#)

[Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric With a Virtual Gateway | 866](#)

[Understanding EVPN with VXLAN Data Plane Encapsulation | 588](#)

[EVPN Primer](#)

Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric Using MX Routers as Spines

IN THIS SECTION

[Requirements | 809](#)

- Overview | 810
- Topology | 810
- Configuration | 811
- Verification | 830

This example shows how to configure EVPN and VXLAN on an IP fabric to support optimal forwarding of Ethernet frames, provide network segmentation on a broad scale, enable control plane-based MAC learning, and many other advantages. This example is based on a centrally-routed with bridging (CRB) EVPN architecture in a 5-stage Clos fabric.

In the CRB architecture IRB interfaces provide Layer 3 connectivity to servers and VMS that belong to different VLANs and networks. These IRB interfaces serve as the default gateway for inter-VLAN traffic within a fabric, and also for destinations that are remote to the fabric, for example, in the case of Data Center Interconnect (DCI). In a CRB design you define the IRB interfaces on the spine devices only. Such a design is therefore referred to as being centrally routed, as all routing occurs on the spines.

For an example of an edge-routed bridging (ERB) design, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway" on page 845](#)

For background information on EVPN-VXLAN technology and supported architectures, see [EVPN Primer](#).

Requirements

The original example used the following hardware and software components:

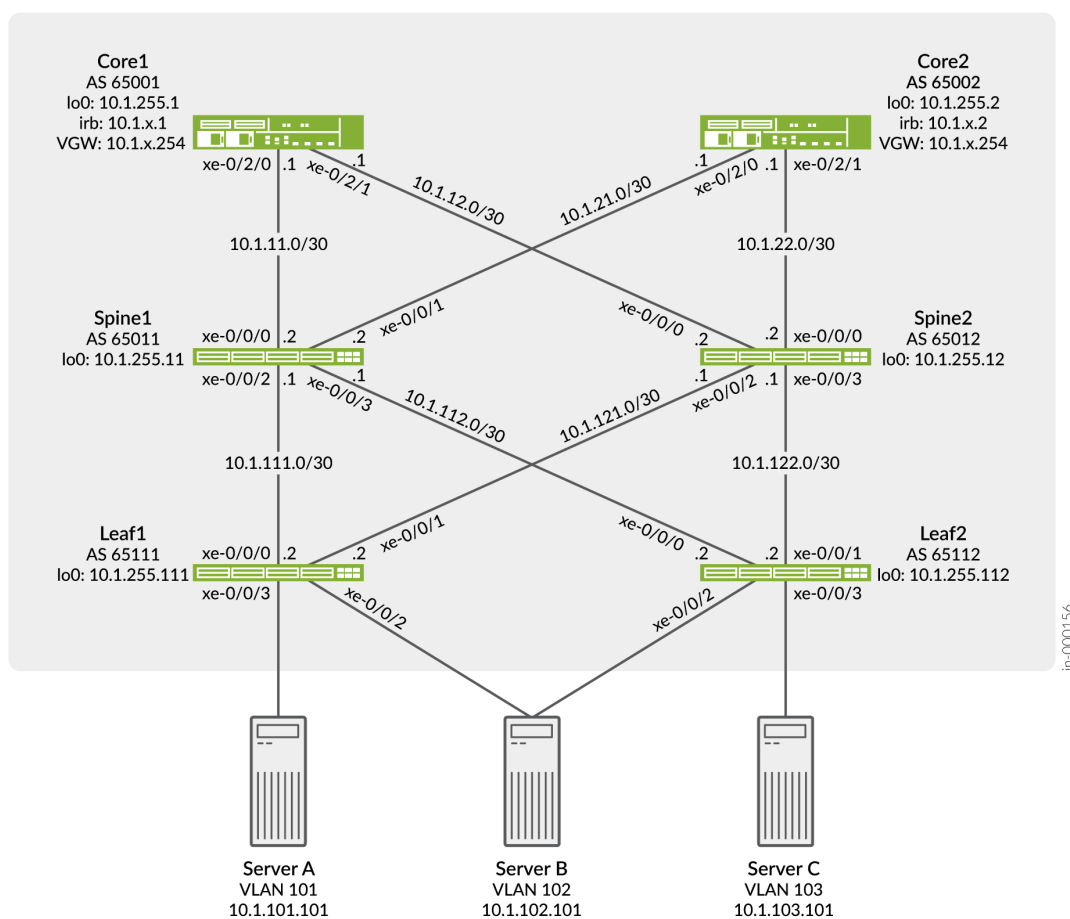
- Two Juniper Networks MX Series routers to act as IP gateways for the EVPN overlay
- Four Juniper Networks QFX5100 switches. Two of these switches act as PE devices in the EVPN topology, and the other two switches act as pure IP transport for the underlay.
- Junos OS Release 16.1 or later.
 - Updated and re-validated using Junos OS Release 21.3R1.9
- Starting with Junos OS Release 17.3R1, EVPN-VXLAN is also supported on EX9200 switches. Previously, only MPLS encapsulation was supported. In this example, the EX9200 switch would function as an IP gateway for the EVPN overlay. There are some configuration differences between MX Series routers and EX9200 switches. The configuration section later in this topic has more information about the configuration that is specific to an EX9200.
 - See the [hardware summary](#) for a list of supported platforms.

Overview

Ethernet VPNs (EVPNs) enable you to connect groups of dispersed customer sites using Layer 2 virtual bridges, and Virtual Extensible LANs (VXLANs) enable you to stretch Layer 2 connection over an intervening Layer 3 network, while providing network segmentation like a VLAN, but without the scaling limitation of traditional VLANs. EVPN with VXLAN encapsulation handles Layer 2 connectivity at the scale required by cloud service providers, and replaces limiting protocols like STP, freeing up your Layer 3 network to use more robust routing protocols.

This example configuration shows how to configure EVPN with VXLAN encapsulation. In this example, the MX Series routers are named Core-1 and Core-2. The QFX5100 switches are named Leaf-1, Leaf-2, Spine-1, and Spine-2. The core routers act as IP gateways for the EVPN overlay, the leaf switches act as PE devices in the EVPN topology, and the spine switches act as pure IP transport for the underlay (also known as a "lean spine").

Topology



In our sample topology we demonstrate server access using both untagged and trunked (tagged) interfaces. A trunk interface uses explicit VLAN tagging. Both server A and C are configured for trunking while server B uses an untagged access interface to both leaves.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 811](#)
- [Configuring Leaf-1 | 820](#)
- [Configuring Spine-1 | 825](#)
- [Configuring Core-1 | 826](#)

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.

Leaf-1

```
set system host-name leaf-1
set chassis aggregated-devices ethernet device-count 2
set interfaces xe-0/0/0 unit 0 family inet address 10.1.111.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.121.2/30
set interfaces xe-0/0/2 ether-options 802.3ad ae0
set interfaces xe-0/0/3 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/3 unit 0 family ethernet-switching vlan members v101
set interfaces ae0 esi 00:01:01:01:01:01:01:01:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:00:01:01:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode access
set interfaces ae0 unit 0 family ethernet-switching vlan members v102
set interfaces lo0 unit 0 family inet address 10.1.255.111/32
set policy-options policy-statement lo0 from family inet
set policy-options policy-statement lo0 from protocol direct
set policy-options policy-statement lo0 from route-filter 0.0.0.0/0 prefix-length-range /32-/32
set policy-options policy-statement lo0 then accept
```

```

set policy-options policy-statement load-balance term 1 then load-balance per-packet
set policy-options policy-statement vrf-imp term t1 from community com101
set policy-options policy-statement vrf-imp term t1 then accept
set policy-options policy-statement vrf-imp term t2 from community com102
set policy-options policy-statement vrf-imp term t2 then accept
set policy-options policy-statement vrf-imp term t3 from community com103
set policy-options policy-statement vrf-imp term t3 then accept
set policy-options policy-statement vrf-imp term t5 then reject
set policy-options community com101 members target:65000:101
set policy-options community com102 members target:65000:102
set policy-options community com103 members target:65000:103
set routing-options router-id 10.1.255.111
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balance
set routing-options forwarding-table ecmp-fast-reroute
set protocols bgp group underlay type external
set protocols bgp group underlay export lo0
set protocols bgp group underlay local-as 65111
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.111.1 peer-as 65011
set protocols bgp group underlay neighbor 10.1.121.1 peer-as 65012
set protocols bgp group EVPN_VXLAN_CORE type internal
set protocols bgp group EVPN_VXLAN_CORE local-address 10.1.255.111
set protocols bgp group EVPN_VXLAN_CORE family evpn signaling
set protocols bgp group EVPN_VXLAN_CORE neighbor 10.1.255.1
set protocols bgp group EVPN_VXLAN_CORE neighbor 10.1.255.2
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:65000:101
set protocols evpn vni-options vni 102 vrf-target target:65000:102
set protocols evpn extended-vni-list 101
set protocols evpn extended-vni-list 102
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.111:1
set switch-options vrf-import vrf-imp
set switch-options vrf-target target:65000:1
set vlans v101 vlan-id 101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 vxlan vni 102

```

Leaf-2

```

set system host-name leaf-2
set chassis aggregated-devices ethernet device-count 2
set interfaces xe-0/0/0 unit 0 family inet address 10.1.112.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.122.2/30
set interfaces xe-0/0/2 ether-options 802.3ad ae0
set interfaces xe-0/0/3 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/3 unit 0 family ethernet-switching vlan members v103
set interfaces ae0 esi 00:01:01:01:01:01:01:01:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:00:01:01:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode access
set interfaces ae0 unit 0 family ethernet-switching vlan members v102
set interfaces lo0 unit 0 family inet address 10.1.255.112/32
set policy-options policy-statement lo0 from family inet
set policy-options policy-statement lo0 from protocol direct
set policy-options policy-statement lo0 from route-filter 0.0.0.0/0 prefix-length-range /32-/32
set policy-options policy-statement lo0 then accept
set policy-options policy-statement load-balance term 1 then load-balance per-packet
set policy-options policy-statement vrf-imp term t1 from community com101
set policy-options policy-statement vrf-imp term t1 then accept
set policy-options policy-statement vrf-imp term t2 from community com102
set policy-options policy-statement vrf-imp term t2 then accept
set policy-options policy-statement vrf-imp term t3 from community com103
set policy-options policy-statement vrf-imp term t3 then accept
set policy-options policy-statement vrf-imp term t5 then reject
set policy-options community com101 members target:65000:101
set policy-options community com102 members target:65000:102
set policy-options community com103 members target:65000:103
set routing-options router-id 10.1.255.112
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balance
set routing-options forwarding-table ecmp-fast-reroute
set protocols bgp group underlay type external
set protocols bgp group underlay export lo0
set protocols bgp group underlay local-as 65112
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.112.1 peer-as 65011
set protocols bgp group underlay neighbor 10.1.122.1 peer-as 65012
set protocols bgp group EVPN_VXLAN_CORE type internal

```

```

set protocols bgp group EVPN_VXLAN_CORE local-address 10.1.255.112
set protocols bgp group EVPN_VXLAN_CORE family evpn signaling
set protocols bgp group EVPN_VXLAN_CORE neighbor 10.1.255.1
set protocols bgp group EVPN_VXLAN_CORE neighbor 10.1.255.2
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 102 vrf-target target:65000:102
set protocols evpn vni-options vni 103 vrf-target target:65000:103
set protocols evpn extended-vni-list 102
set protocols evpn extended-vni-list 103
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.112:1
set switch-options vrf-import vrf-imp
set switch-options vrf-target target:65000:1
set vlans v102 vlan-id 102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 vxlan vni 103

```

Spine-1

```

set system host-name spine-1
set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.21.2/30
set interfaces xe-0/0/2 unit 0 family inet address 10.1.111.1/30
set interfaces xe-0/0/3 unit 0 family inet address 10.1.112.1/30
set interfaces lo0 unit 0 family inet address 10.1.255.11/32
set policy-options policy-statement lo0 from family inet
set policy-options policy-statement lo0 from protocol direct
set policy-options policy-statement lo0 from route-filter 0.0.0.0/0 prefix-length-range /32-/32
set policy-options policy-statement lo0 then accept
set policy-options policy-statement load-balance term 1 then load-balance per-packet
set routing-options router-id 10.1.255.11
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balance
set routing-options forwarding-table ecmp-fast-reroute
set protocols bgp group underlay type external
set protocols bgp group underlay export lo0
set protocols bgp group underlay local-as 65011
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002

```

```
set protocols bgp group underlay neighbor 10.1.111.2 peer-as 65111
set protocols bgp group underlay neighbor 10.1.112.2 peer-as 65112
```

Spine-2

```
set system host-name spine-2
set interfaces xe-0/0/0 unit 0 family inet address 10.1.12.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.22.2/30
set interfaces xe-0/0/2 unit 0 family inet address 10.1.121.1/30
set interfaces xe-0/0/3 unit 0 family inet address 10.1.122.1/30
set interfaces lo0 unit 0 family inet address 10.1.255.12/32
set policy-options policy-statement lo0 from family inet
set policy-options policy-statement lo0 from protocol direct
set policy-options policy-statement lo0 from route-filter 0.0.0.0/0 prefix-length-range /32-/32
set policy-options policy-statement lo0 then accept
set policy-options policy-statement load-balance term 1 then load-balance per-packet
set routing-options router-id 10.1.255.12
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balance
set routing-options forwarding-table ecmp-fast-reroute
set protocols bgp group underlay type external
set protocols bgp group underlay export lo0
set protocols bgp group underlay local-as 65012
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.12.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.22.1 peer-as 65002
set protocols bgp group underlay neighbor 10.1.121.2 peer-as 65111
set protocols bgp group underlay neighbor 10.1.122.2 peer-as 65112
```

Core-1

```
set system host-name core-1
set interfaces xe-0/2/0 unit 0 family inet address 10.1.11.1/30
set interfaces xe-0/2/1 unit 0 family inet address 10.1.12.1/30
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.1/24 virtual-gateway-address
10.1.101.254
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 family inet address 10.1.102.1/24 virtual-gateway-address
10.1.102.254
set interfaces irb unit 103 virtual-gateway-accept-data
```



```

set interfaces irb unit 103 family inet address 10.1.103.1/24 virtual-gateway-address
10.1.103.254
set interfaces lo0 unit 0 family inet address 10.1.255.1/32
set policy-options policy-statement VS_VLAN101_IMP term ESI from community comm-leaf
set policy-options policy-statement VS_VLAN101_IMP term ESI then accept
set policy-options policy-statement VS_VLAN101_IMP term VS_VLAN101 from community comm-VS_VLAN101
set policy-options policy-statement VS_VLAN101_IMP term VS_VLAN101 then accept
set policy-options policy-statement VS_VLAN102_IMP term ESI from community comm-leaf
set policy-options policy-statement VS_VLAN102_IMP term ESI then accept
set policy-options policy-statement VS_VLAN102_IMP term VS_VLAN102 from community comm-VS_VLAN102
set policy-options policy-statement VS_VLAN102_IMP term VS_VLAN102 then accept
set policy-options policy-statement VS_VLAN103_IMP term ESI from community comm-leaf
set policy-options policy-statement VS_VLAN103_IMP term ESI then accept
set policy-options policy-statement VS_VLAN103_IMP term VS_VLAN103 from community comm-VS_VLAN103
set policy-options policy-statement VS_VLAN103_IMP term VS_VLAN103 then accept
set policy-options policy-statement lo0 from family inet
set policy-options policy-statement lo0 from protocol direct
set policy-options policy-statement lo0 from route-filter 0.0.0.0/0 prefix-length-range /32-/32
set policy-options policy-statement lo0 then accept
set policy-options policy-statement load-balance term 1 then load-balance per-packet
set policy-options community comm-VS_VLAN101 members target:65000:101
set policy-options community comm-VS_VLAN102 members target:65000:102
set policy-options community comm-VS_VLAN103 members target:65000:103
set policy-options community comm-leaf members target:65000:1
set routing-instances VRF_Tenant_A instance-type vrf
set routing-instances VRF_Tenant_A interface irb.101
set routing-instances VRF_Tenant_A route-distinguisher 10.1.255.1:1010
set routing-instances VRF_Tenant_A vrf-target target:65000:101
set routing-instances VRF_Tenant_B instance-type vrf
set routing-instances VRF_Tenant_B interface irb.102
set routing-instances VRF_Tenant_B route-distinguisher 10.1.255.1:1020
set routing-instances VRF_Tenant_B vrf-target target:65000:102
set routing-instances VRF_Tenant_C instance-type vrf
set routing-instances VRF_Tenant_C interface irb.103
set routing-instances VRF_Tenant_C route-distinguisher 10.1.255.1:1030
set routing-instances VRF_Tenant_C vrf-target target:65000:103
set routing-instances VS_VLAN101 instance-type virtual-switch
set routing-instances VS_VLAN101 protocols evpn encapsulation vxlan
set routing-instances VS_VLAN101 protocols evpn extended-vni-list 101
set routing-instances VS_VLAN101 protocols evpn multicast-mode ingress-replication
set routing-instances VS_VLAN101 vtep-source-interface lo0.0
set routing-instances VS_VLAN101 bridge-domains bd101 vlan-id 101
set routing-instances VS_VLAN101 bridge-domains bd101 routing-interface irb.101

```

```

set routing-instances VS_VLAN101 bridge-domains bd101 vxlan vni 101
set routing-instances VS_VLAN101 route-distinguisher 10.1.255.1:101
set routing-instances VS_VLAN101 vrf-import VS_VLAN101_IMP
set routing-instances VS_VLAN101 vrf-target target:65000:101
set routing-instances VS_VLAN102 instance-type virtual-switch
set routing-instances VS_VLAN102 protocols evpn encapsulation vxlan
set routing-instances VS_VLAN102 protocols evpn extended-vni-list 102
set routing-instances VS_VLAN102 protocols evpn multicast-mode ingress-replication
set routing-instances VS_VLAN102 vtep-source-interface lo0.0
set routing-instances VS_VLAN102 bridge-domains bd102 vlan-id 102
set routing-instances VS_VLAN102 bridge-domains bd102 routing-interface irb.102
set routing-instances VS_VLAN102 bridge-domains bd102 vxlan vni 102
set routing-instances VS_VLAN102 route-distinguisher 10.1.255.1:102
set routing-instances VS_VLAN102 vrf-import VS_VLAN102_IMP
set routing-instances VS_VLAN102 vrf-target target:65000:102
set routing-instances VS_VLAN103 instance-type virtual-switch
set routing-instances VS_VLAN103 protocols evpn encapsulation vxlan
set routing-instances VS_VLAN103 protocols evpn extended-vni-list 103
set routing-instances VS_VLAN103 protocols evpn multicast-mode ingress-replication
set routing-instances VS_VLAN103 vtep-source-interface lo0.0
set routing-instances VS_VLAN103 bridge-domains bd103 vlan-id 103
set routing-instances VS_VLAN103 bridge-domains bd103 routing-interface irb.103
set routing-instances VS_VLAN103 bridge-domains bd103 vxlan vni 103
set routing-instances VS_VLAN103 route-distinguisher 10.1.255.1:103
set routing-instances VS_VLAN103 vrf-import VS_VLAN103_IMP
set routing-instances VS_VLAN103 vrf-target target:65000:103
set routing-options router-id 10.1.255.1
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balance
set routing-options forwarding-table ecmp-fast-reroute
set protocols bgp group underlay type external
set protocols bgp group underlay export lo0
set protocols bgp group underlay local-as 65001
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.2 peer-as 65011
set protocols bgp group underlay neighbor 10.1.12.2 peer-as 65012
set protocols bgp group EVPN_VXLAN type internal
set protocols bgp group EVPN_VXLAN local-address 10.1.255.1
set protocols bgp group EVPN_VXLAN family evpn signaling
set protocols bgp group EVPN_VXLAN cluster 10.1.1.1
set protocols bgp group EVPN_VXLAN multipath
set protocols bgp group EVPN_VXLAN neighbor 10.1.255.111

```

```
set protocols bgp group EVPN_VXLAN neighbor 10.1.255.112
set protocols bgp group EVPN_VXLAN neighbor 10.1.255.2
```

Core-2

```
set system host-name core-2
set interfaces xe-0/2/0 unit 0 family inet address 10.1.21.1/30
set interfaces xe-0/2/1 unit 0 family inet address 10.1.22.1/30
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.2/24 virtual-gateway-address
10.1.101.254
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 family inet address 10.1.102.2/24 virtual-gateway-address
10.1.102.254
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 family inet address 10.1.103.2/24 virtual-gateway-address
10.1.103.254
set interfaces lo0 unit 0 family inet address 10.1.255.2/32
set policy-options policy-statement VS_VLAN101_IMP term ESI from community comm-leaf
set policy-options policy-statement VS_VLAN101_IMP term ESI then accept
set policy-options policy-statement VS_VLAN101_IMP term VS_VLAN101 from community comm-VS_VLAN101
set policy-options policy-statement VS_VLAN101_IMP term VS_VLAN101 then accept
set policy-options policy-statement VS_VLAN102_IMP term ESI from community comm-leaf
set policy-options policy-statement VS_VLAN102_IMP term ESI then accept
set policy-options policy-statement VS_VLAN102_IMP term VS_VLAN102 from community comm-VS_VLAN102
set policy-options policy-statement VS_VLAN102_IMP term VS_VLAN102 then accept
set policy-options policy-statement VS_VLAN103_IMP term ESI from community comm-leaf
set policy-options policy-statement VS_VLAN103_IMP term ESI then accept
set policy-options policy-statement VS_VLAN103_IMP term VS_VLAN103 from community comm-VS_VLAN103
set policy-options policy-statement VS_VLAN103_IMP term VS_VLAN103 then accept
set policy-options policy-statement lo0 from family inet
set policy-options policy-statement lo0 from protocol direct
set policy-options policy-statement lo0 from route-filter 0.0.0.0/0 prefix-length-range /32-/32
set policy-options policy-statement lo0 then accept
set policy-options policy-statement load-balance term 1 then load-balance per-packet
set policy-options community comm-VS_VLAN101 members target:65000:101
set policy-options community comm-VS_VLAN102 members target:65000:102
set policy-options community comm-VS_VLAN103 members target:65000:103
set policy-options community comm-leaf members target:65000:1
set routing-instances VRF_Tenant_A instance-type vrf
set routing-instances VRF_Tenant_A interface irb.101
set routing-instances VRF_Tenant_A route-distinguisher 10.1.255.2:1010
```

```

set routing-instances VRF_Tenant_A vrf-target target:65000:101
set routing-instances VRF_Tenant_B instance-type vrf
set routing-instances VRF_Tenant_B interface irb.102
set routing-instances VRF_Tenant_B route-distinguisher 10.1.255.2:1020
set routing-instances VRF_Tenant_B vrf-target target:65000:102
set routing-instances VRF_Tenant_C instance-type vrf
set routing-instances VRF_Tenant_C interface irb.103
set routing-instances VRF_Tenant_C route-distinguisher 10.1.255.2:1030
set routing-instances VRF_Tenant_C vrf-target target:65000:103
set routing-instances VS_VLAN101 instance-type virtual-switch
set routing-instances VS_VLAN101 protocols evpn encapsulation vxlan
set routing-instances VS_VLAN101 protocols evpn extended-vni-list 101
set routing-instances VS_VLAN101 protocols evpn multicast-mode ingress-replication
set routing-instances VS_VLAN101 vtep-source-interface lo0.0
set routing-instances VS_VLAN101 bridge-domains bd101 vlan-id 101
set routing-instances VS_VLAN101 bridge-domains bd101 routing-interface irb.101
set routing-instances VS_VLAN101 bridge-domains bd101 vxlan vni 101
set routing-instances VS_VLAN101 route-distinguisher 10.1.255.2:101
set routing-instances VS_VLAN101 vrf-import VS_VLAN101_IMP
set routing-instances VS_VLAN101 vrf-target target:65000:101
set routing-instances VS_VLAN102 instance-type virtual-switch
set routing-instances VS_VLAN102 protocols evpn encapsulation vxlan
set routing-instances VS_VLAN102 protocols evpn extended-vni-list 102
set routing-instances VS_VLAN102 protocols evpn multicast-mode ingress-replication
set routing-instances VS_VLAN102 vtep-source-interface lo0.0
set routing-instances VS_VLAN102 bridge-domains bd102 vlan-id 102
set routing-instances VS_VLAN102 bridge-domains bd102 routing-interface irb.102
set routing-instances VS_VLAN102 bridge-domains bd102 vxlan vni 102
set routing-instances VS_VLAN102 route-distinguisher 10.1.255.2:102
set routing-instances VS_VLAN102 vrf-import VS_VLAN102_IMP
set routing-instances VS_VLAN102 vrf-target target:65000:102
set routing-instances VS_VLAN103 instance-type virtual-switch
set routing-instances VS_VLAN103 protocols evpn encapsulation vxlan
set routing-instances VS_VLAN103 protocols evpn extended-vni-list 103
set routing-instances VS_VLAN103 protocols evpn multicast-mode ingress-replication
set routing-instances VS_VLAN103 vtep-source-interface lo0.0
set routing-instances VS_VLAN103 bridge-domains bd103 vlan-id 103
set routing-instances VS_VLAN103 bridge-domains bd103 routing-interface irb.103
set routing-instances VS_VLAN103 bridge-domains bd103 vxlan vni 103
set routing-instances VS_VLAN103 route-distinguisher 10.1.255.2:103
set routing-instances VS_VLAN103 vrf-import VS_VLAN103_IMP
set routing-instances VS_VLAN103 vrf-target target:65000:103
set routing-options router-id 10.1.255.2

```

```

set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balance
set routing-options forwarding-table ecmp-fast-reroute
set protocols bgp group underlay type external
set protocols bgp group underlay export lo0
set protocols bgp group underlay local-as 65002
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.21.2 peer-as 65011
set protocols bgp group underlay neighbor 10.1.22.2 peer-as 65012
set protocols bgp group EVPN_VXLAN type internal
set protocols bgp group EVPN_VXLAN local-address 10.1.255.2
set protocols bgp group EVPN_VXLAN family evpn signaling
set protocols bgp group EVPN_VXLAN cluster 10.2.2.2
set protocols bgp group EVPN_VXLAN multipath
set protocols bgp group EVPN_VXLAN neighbor 10.1.255.111
set protocols bgp group EVPN_VXLAN neighbor 10.1.255.112
set protocols bgp group EVPN_VXLAN neighbor 10.1.255.1

```

EX9200 Configuration

On EX9200 switches, the `vlan` statement is used instead of `bridge-domains`, and the `l3-interface` statement is used instead of `routing-interface`.

The following example shows how to configure these statements. All other configuration shown for MX Series routers in this example also applies to EX9200 switches.

```

set routing-instances VS_VLAN300 vlan vlan1300 vlan-id 300
set routing-instances VS_VLAN300 vlan vlan1300 l3-interface irb.1300

```



NOTE: In this example, wherever `bridge-domains` or `routing-interface` statements are used, to configure on EX9200 switches, use `vlan` and `l3-interface` instead.

Configuring Leaf-1

Step-by-Step Procedure

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



NOTE: The steps for configuring Leaf-2 are similar to Leaf-1 and therefore we will only show the step-by-step procedures for Leaf-1.

To configure Leaf-1:

1. Set the system hostname.

```
[edit]
user@leaf-1# set system host-name leaf-1
```

2. Configure routing options. The *load-balance* export policy is configured in the next step.

```
[edit]
user@leaf-1# set routing-options router-id 10.1.255.111
user@leaf-1# set routing-options autonomous-system 65000
user@leaf-1# set routing-options forwarding-table export load-balance
user@leaf-1# set routing-options forwarding-table ecmp-fast-reroute
```

3. Configure the load balancing policy.

```
[edit policy-options policy-statement load-balance]
user@leaf-1# set term 1 then load-balance per-packet
```

4. Configure the underlay EBGP to the spine devices. The *lo0* export policy is configured in the next step.

```
[edit]
user@leaf-1# set protocols bgp group underlay type external
user@leaf-1# set protocols bgp group underlay export lo0
user@leaf-1# set protocols bgp group underlay local-as 65111
user@leaf-1# set protocols bgp group underlay multipath multiple-as
user@leaf-1# set protocols bgp group underlay neighbor 10.1.111.1 peer-as 65011
user@leaf-1# set protocols bgp group underlay neighbor 10.1.121.1 peer-as 65012
```

5. Configure a policy to advertise the loopback address into the underlay. In this example you write a portable policy that is loopback address agnostic, by matching only direct routes with a /32 prefix

length. The result is a policy that matches any loopback address and is reusable across all devices in the topology.

```
[edit policy-options policy-statement lo0]
user@leaf-1# set from family inet
user@leaf-1# set from protocol direct
user@leaf-1# set from route-filter 0.0.0.0/0 prefix-length-range /32-/32
user@leaf-1# set then accept
```

6. Configure switch options The virtual tunnel endpoint interface is lo0.0, which must be reachable through the underlay routing protocol. The route distinguisher must be unique across all switches in the network to ensure all route advertisements within MP-BGP overlay are globally unique. The VRF table target on the QFX Series switch is, at a minimum, the community the switch sends attaches to all ESI (Type-1) routes. The `vrf-import vrf-imp` statement defines the target community list, which is imported into the `default-switch.evpn.0` instance from the `bgp.evpn.0` table.

```
[edit]
user@leaf-1# set switch-options vtep-source-interface lo0.0
user@leaf-1# set switch-options route-distinguisher 10.1.255.111:1
user@leaf-1# set switch-options vrf-import vrf-imp
user@leaf-1# set switch-options vrf-target target:65000:1
```

7. Configure the VRF table import policy.

```
[edit]
user@leaf-1# set policy-options policy-statement vrf-imp term t1 from community com101
user@leaf-1# set policy-options policy-statement vrf-imp term t1 then accept
user@leaf-1# set policy-options policy-statement vrf-imp term t2 from community com102
user@leaf-1# set policy-options policy-statement vrf-imp term t2 then accept
user@leaf-1# set policy-options policy-statement vrf-imp term t3 from community com103
user@leaf-1# set policy-options policy-statement vrf-imp term t3 then accept
user@leaf-1# set policy-options policy-statement vrf-imp term t5 then reject
```

8. Configure the related communities.

```
[edit]
user@leaf-1# set policy-options community com101 members target:65000:101
```

```

user@leaf-1# set policy-options community com102 members target:65000:102
user@leaf-1# set policy-options community com103 members target:65000:103

```

9. Configure the extended virtual network identifier (VNI) list to establish the VNIs you want to be part of the EVPN domain. You also configure ingress replication; in EVPN-VXLAN ingress-replication is used to handle multicast without requiring a multicast capable underlay. Different route targets are specified for each VXLAN network identifier instance under vni-routing-options.

```

[edit]
user@leaf-1# set protocols evpn encapsulation vxlan
user@leaf-1# set protocols evpn multicast-mode ingress-replication
user@leaf-1# set protocols evpn vni-options vni 101 vrf-target target:65000:101
user@leaf-1# set protocols evpn vni-options vni 102 vrf-target target:65000:102
user@leaf-1# set protocols evpn extended-vni-list 101
user@leaf-1# set protocols evpn extended-vni-list 102

```

10. Map locally significant VLAN IDs to globally significant VXLAN network identifiers.

```

[edit]
user@leaf-1# set vlans v101 vlan-id 101
user@leaf-1# set vlans v101 vxlan vni 101
user@leaf-1# set vlans v102 vlan-id 102
user@leaf-1# set vlans v102 vxlan vni 102

```

11. Configure the EVPN capable IBGP overlay sessions.

```

[edit]
user@leaf-1# set protocols bgp group EVPN_VXLAN_CORE type internal
user@leaf-1# set protocols bgp group EVPN_VXLAN_CORE local-address 10.1.255.111
user@leaf-1# set protocols bgp group EVPN_VXLAN_CORE family evpn signaling
user@leaf-1# set protocols bgp group EVPN_VXLAN_CORE neighbor 10.1.255.1
user@leaf-1# set protocols bgp group EVPN_VXLAN_CORE neighbor 10.1.255.2

```



NOTE: Some IP fabrics use an EBGp based EVPN-VXLAN overlay. For an example of an IP fabric that uses EBGp for both the underlay and overlay, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway" on page 845](#). Note that the choice of EBGp vs IBGP for the overlay does

not impact on the fabric architecture. Both CRB and edge-routed bridging (ERB) designs support either type of overlay.

12. Configure the fabric interfaces.

```
[edit]
user@leaf-1# set interfaces xe-0/0/0 unit 0 family inet address 10.1.111.2/30
user@leaf-1# set interfaces xe-0/0/1 unit 0 family inet address 10.1.121.2/30
```

13. Configure the access interfaces. Note again that we demonstrate a mix of access and trunk interfaces for server attachment.

```
[edit]
user@leaf-1# set interfaces xe-0/0/2 ether-options 802.3ad ae0
user@leaf-1# set interfaces xe-0/0/3 unit 0 family ethernet-switching interface-mode trunk
user@leaf-1# set interfaces xe-0/0/3 unit 0 family ethernet-switching vlan members v101
```

14. Configure the LACP-enabled LAG interface. The ESI value is globally unique across the entire EVPN domain. The all-active configuration statement ensures that all PE routers to which this multihomed tenant is attached to can forward traffic from the CE device, such that all CE links are actively used.

```
[edit]
user@leaf-1# set interfaces ae0 esi 00:01:01:01:01:01:01:01:01
user@leaf-1# set interfaces ae0 esi all-active
user@leaf-1# set interfaces ae0 aggregated-ether-options lacp active
user@leaf-1# set interfaces ae0 aggregated-ether-options lacp system-id 00:00:00:01:01:01
user@leaf-1# set interfaces ae0 unit 0 family ethernet-switching interface-mode access
user@leaf-1# set interfaces ae0 unit 0 family ethernet-switching vlan members v102
```

15. Configure the loopback interface address.

```
[edit]
user@leaf-1# set interfaces lo0 unit 0 family inet address 10.1.255.111/32
```

Configuring Spine-1

Step-by-Step Procedure

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



NOTE: The steps for configuring Spine-2 are similar to Spine-1 and therefore we will only show the step-by-step procedures for Spine-1.

To configure Spine-1:

1. Set the system hostname.

```
[edit]
user@spine-1# set system host-name spine-1
```

2. Configure the routing options.

```
[edit]
user@spine-1# set routing-options router-id 10.1.255.11
user@spine-1# set routing-options autonomous-system 65000
user@spine-1# set routing-options forwarding-table export load-balance
user@spine-1# set routing-options forwarding-table ecmp-fast-reroute
```

3. Configure a load balancing policy.

```
[edit policy-options policy-statement load-balance]
user@spine-1# set term 1 then load-balance per-packet
```

4. Configure the EBGP underlay with peering to the leaf and core devices. The *lo0* policy that advertises the *lo0* address is applied in this step; the configuration of the policy itself is shown in the next step.

```
[edit]
user@spine-1# set protocols bgp group underlay type external
user@spine-1# set protocols bgp group underlay export lo0
user@spine-1# set protocols bgp group underlay local-as 65011
```

```

user@spine-1# set protocols bgp group underlay multipath multiple-as
user@spine-1# set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
user@spine-1# set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002
user@spine-1# set protocols bgp group underlay neighbor 10.1.111.2 peer-as 65111
user@spine-1# set protocols bgp group underlay neighbor 10.1.112.2 peer-as 65112

```

5. Configure a policy named *lo0* to advertise */32* routes. The policy matches on the loopback address, without specifying any specific IP. In this way the same policy is reusable on any fabric device.

```

[edit policy-options policy-statement lo0]
user@spine-1# set from family inet
user@spine-1# set from protocol direct
user@spine-1# set from route-filter 0.0.0.0/0 prefix-length-range /32-/32
user@spine-1# set then accept

```

Configuring Core-1

Step-by-Step Procedure

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



NOTE: The steps for configuring Core-2 are similar to Core-1 and therefore we will only show the step-by-step procedures for Core-1.

To configure Core-1:

1. Set the system hostname.

```

[edit]
user@core-1# set system host-name core-1

```

2. Configure the routing options. The *load-balance* policy is applied during this step. You create the policy in the next step

```

[edit]
user@core-1# set routing-options router-id 10.1.255.1
user@core-1# set routing-options autonomous-system 65000

```

```
user@core-1# set routing-options forwarding-table export load-balance
user@core-1# set routing-options forwarding-table ecmp-fast-reroute
```

3. Configure a load balancing policy named *load-balance*.

```
[edit policy-options policy-statement load-balance]
user@core-1# set term 1 then load-balance per-packet
```

4. Configure the BGP underlay peering. The *lo0* policy that advertises the loopback address is applied during this step. You configure this policy in the next step.

```
[edit]
user@core-1# set protocols bgp group underlay type external
user@core-1# set protocols bgp group underlay export lo0
user@core-1# set protocols bgp group underlay local-as 65001
user@core-1# set protocols bgp group underlay multipath multiple-as
user@core-1# set protocols bgp group underlay neighbor 10.1.11.2 peer-as 65011
user@core-1# set protocols bgp group underlay neighbor 10.1.12.2 peer-as 65012
```

5. Configure a policy named *lo0* to advertise loopback routes.

```
[edit policy-options policy-statement lo0]
user@core-1# set from family inet
user@core-1# set from protocol direct
user@core-1# set lo0 from route-filter 0.0.0.0/0 prefix-length-range /32-/32
user@core-1# set lo0 then accept
```

6. A large portion of Core-1's configuration takes place in the [routing-instance] hierarchy. Configure the virtual routers and configure a unique VRF table import policy for each virtual switch.

```
[edit]
user@core-1# set routing-instances VRF_Tenant_A instance-type vrf
user@core-1# set routing-instances VRF_Tenant_A interface irb.101
user@core-1# set routing-instances VRF_Tenant_A route-distinguisher 10.1.255.1:1010
user@core-1# set routing-instances VRF_Tenant_A vrf-target target:65000:101
user@core-1# set routing-instances VRF_Tenant_B instance-type vrf
user@core-1# set routing-instances VRF_Tenant_B interface irb.102
user@core-1# set routing-instances VRF_Tenant_B route-distinguisher 10.1.255.1:1020
user@core-1# set routing-instances VRF_Tenant_B vrf-target target:65000:102
```

```

user@core-1# set routing-instances VRF_Tenant_C instance-type vrf
user@core-1# set routing-instances VRF_Tenant_C interface irb.103
user@core-1# set routing-instances VRF_Tenant_C route-distinguisher 10.1.255.1:1030
user@core-1# set routing-instances VRF_Tenant_C vrf-target target:65000:103
user@core-1# set routing-instances VS_VLAN101 instance-type virtual-switch
user@core-1# set routing-instances VS_VLAN101 protocols evpn encapsulation vxlan
user@core-1# set routing-instances VS_VLAN101 protocols evpn extended-vni-list 101
user@core-1# set routing-instances VS_VLAN101 protocols evpn multicast-mode ingress-
replication
user@core-1# set routing-instances VS_VLAN101 vtep-source-interface lo0.0
user@core-1# set routing-instances VS_VLAN101 bridge-domains bd101 vlan-id 101
user@core-1# set routing-instances VS_VLAN101 bridge-domains bd101 routing-interface irb.101
user@core-1# set routing-instances VS_VLAN101 bridge-domains bd101 vxlan vni 101
user@core-1# set routing-instances VS_VLAN101 route-distinguisher 10.1.255.1:101
user@core-1# set routing-instances VS_VLAN101 vrf-import VS_VLAN101_IMP
user@core-1# set routing-instances VS_VLAN101 vrf-target target:65000:101
user@core-1# set routing-instances VS_VLAN102 instance-type virtual-switch
user@core-1# set routing-instances VS_VLAN102 protocols evpn encapsulation vxlan
user@core-1# set routing-instances VS_VLAN102 protocols evpn extended-vni-list 102
user@core-1# set routing-instances VS_VLAN102 protocols evpn multicast-mode ingress-
replication
user@core-1# set routing-instances VS_VLAN102 vtep-source-interface lo0.0
user@core-1# set routing-instances VS_VLAN102 bridge-domains bd102 vlan-id 102
user@core-1# set routing-instances VS_VLAN102 bridge-domains bd102 routing-interface irb.102
user@core-1# set routing-instances VS_VLAN102 bridge-domains bd102 vxlan vni 102
user@core-1# set routing-instances VS_VLAN102 route-distinguisher 10.1.255.1:102
user@core-1# set routing-instances VS_VLAN102 vrf-import VS_VLAN102_IMP
user@core-1# set routing-instances VS_VLAN102 vrf-target target:65000:102
user@core-1# set routing-instances VS_VLAN103 instance-type virtual-switch
user@core-1# set routing-instances VS_VLAN103 protocols evpn encapsulation vxlan
user@core-1# set routing-instances VS_VLAN103 protocols evpn extended-vni-list 103
user@core-1# set routing-instances VS_VLAN103 protocols evpn multicast-mode ingress-
replication
user@core-1# set routing-instances VS_VLAN103 vtep-source-interface lo0.0
user@core-1# set routing-instances VS_VLAN103 bridge-domains bd103 vlan-id 103
user@core-1# set routing-instances VS_VLAN103 bridge-domains bd103 routing-interface irb.103
user@core-1# set routing-instances VS_VLAN103 bridge-domains bd103 vxlan vni 103
user@core-1# set routing-instances VS_VLAN103 route-distinguisher 10.1.255.1:103
user@core-1# set routing-instances VS_VLAN103 vrf-import VS_VLAN103_IMP
user@core-1# set routing-instances VS_VLAN103 vrf-target target:65000:103

```

7. Configure the policy for each routing instance.

```
[edit policy-options]
user@core-1# set policy-statement VS_VLAN101_IMP term ESI from community comm-leaf
user@core-1# set policy-statement VS_VLAN101_IMP term ESI then accept
user@core-1# set policy-statement VS_VLAN101_IMP term VS_VLAN101 from community comm-
VS_VLAN101
user@core-1# set policy-statement VS_VLAN101_IMP term VS_VLAN101 then accept
user@core-1# set policy-statement VS_VLAN102_IMP term ESI from community comm-leaf
user@core-1# set policy-statement VS_VLAN102_IMP term ESI then accept
user@core-1# set policy-statement VS_VLAN102_IMP term VS_VLAN102 from community comm-
VS_VLAN102
user@core-1# set policy-statement VS_VLAN102_IMP term VS_VLAN102 then accept
user@core-1# set policy-statement VS_VLAN103_IMP term ESI from community comm-leaf
user@core-1# set policy-statement VS_VLAN103_IMP term ESI then accept
user@core-1# set policy-statement VS_VLAN103_IMP term VS_VLAN103 from community comm-
VS_VLAN103
user@core-1# set policy-statement VS_VLAN103_IMP term VS_VLAN103 then accept
```

8. Configure the communities . Make sure that the *comm-leaf* policy accepts routes tagged with target 65000:1. This ensures that all virtual switches import the Type-1 ESI routes from all leafs.

```
[edit]
user@core-1# set policy-options community comm-VS_VLAN101 members target:65000:101
user@core-1# set policy-options community comm-VS_VLAN102 members target:65000:102
user@core-1# set policy-options community comm-VS_VLAN103 members target:65000:103
user@core-1# set policy-options community comm-leaf members target:65000:1
```

9. Configure the IRB interfaces. Every IRB has a virtual gateway address, which is a shared MAC address and IP address across Core-1 and Core-2.

```
[edit interfaces irb]
user@core-1# set unit 101 virtual-gateway-accept-data
user@core-1# set unit 101 family inet address 10.1.101.1/24 virtual-gateway-address
10.1.101.254
user@core-1# set unit 102 virtual-gateway-accept-data
user@core-1# set unit 102 family inet address 10.1.102.1/24 virtual-gateway-address
10.1.102.254
user@core-1# set unit 103 virtual-gateway-accept-data
```

```
user@core-1# set unit 103 family inet address 10.1.103.1/24 virtual-gateway-address
10.1.103.254
```

10. Configure the IBGP overlay sessions towards Leaf-1 and Leaf-2. We've include a peering between the Core devices for route sharing between Core devices.

```
[edit]
user@core-1# set protocols bgp group EVPN_VXLAN type internal
user@core-1# set protocols bgp group EVPN_VXLAN local-address 10.1.255.1
user@core-1# set protocols bgp group EVPN_VXLAN family evpn signaling
user@core-1# set protocols bgp group EVPN_VXLAN cluster 10.1.1.1
user@core-1# set protocols bgp group EVPN_VXLAN multipath
user@core-1# set protocols bgp group EVPN_VXLAN neighbor 10.1.255.111
user@core-1# set protocols bgp group EVPN_VXLAN neighbor 10.1.255.112
user@core-1# set protocols bgp group EVPN_VXLAN neighbor 10.1.255.2
```

Verification

IN THIS SECTION

- [Verifying MAC Reachability to a Single-Homed CE Device \(Leaf-1\) | 830](#)
- [Verifying MAC Reachability to a Single-Homed CE Device \(Type-2\) | 831](#)
- [Verifying Imported Routes | 833](#)
- [Verifying the Layer 2 Address Learning Daemon Copy | 834](#)
- [Verifying the Kernel-Level Forwarding Table | 835](#)
- [Verifying MAC Reachability to a Multihomed CE Device | 837](#)
- [Verifying EVPN, Layer 2 Address Learning Daemon, and the Kernel-Forwarding Tables for Multihomed CE Device | 838](#)

Verifying MAC Reachability to a Single-Homed CE Device (Leaf-1)

Purpose

Verify MAC reachability to Tenant_A. This user is single-homed to Leaf-1. First, verify that the MAC address is learned locally on Leaf-1. Leaf-1 generates the Type-2 EVPN route only after it learns the MAC address.

Action

Verify that the MAC address is learned locally on Leaf-1.

```
lab@leaf-1> show ethernet-switching table vlan-id 101
```

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 4 entries, 4 learned

Routing instance : default-switch

Vlan name	MAC address	MAC flags	Logical interface	SVLBNH/ VENH Index	Active source
v101	00:00:5e:00:01:01	DRP	esi.1749		
05:00:00:fd:e8:00:00:65:00					
v101	2c:6b:f5:54:95:f0	DR	vtep.32770		
10.1.255.2					
v101	2c:6b:f5:ef:73:f0	DR	vtep.32769		
10.1.255.1					
v101	56:04:15:00:bb:02	D	xe-0/0/3.0		

Meaning

The output shows that MAC 56:04:15:00:bb:02 is successfully learned from the Tenant_A CE device, which is Server A on the xe-0/0/3.0 interface.

Verifying MAC Reachability to a Single-Homed CE Device (Type-2)

Purpose

Verify MAC reachability to a single-homed CE device (Type-2)

Action

Verify the generation of the Type-2 route to Core-1.

```
lab@leaf-1> show route advertising-protocol bgp 10.1.255.1 evpn-mac-address 56:04:15:00:bb:02
```

bgp.evpn.0: 50 destinations, 91 routes (50 active, 0 holddown, 0 hidden)


```

Prefix                Nexthop                MED    Lclpref    AS path
2:10.1.255.111:1::101::56:04:15:00:bb:02/304 MAC/IP
*                      Self                                100      I
2:10.1.255.111:1::101::56:04:15:00:bb:02::10.1.101.101/304 MAC/IP
*                      Self                                100      I

default-switch.evpn.0: 47 destinations, 87 routes (47 active, 0 holddown, 0 hidden)
Prefix                Nexthop                MED    Lclpref    AS path
2:10.1.255.111:1::101::56:04:15:00:bb:02/304 MAC/IP
*                      Self                                100      I
2:10.1.255.111:1::101::56:04:15:00:bb:02::10.1.101.101/304 MAC/IP
*                      Self                                100      I

__default_evpn__.evpn.0: 3 destinations, 4 routes (3 active, 0 holddown, 0 hidden)

```

Meaning

The output shows that the MAC and MAC/IP are being advertised.

On Core-1, the EVPN Type-2 route is received into bgp.evpn.0.

```

lab@core-1> show route receive-protocol bgp 10.1.255.111 evpn-mac-address 56:04:15:00:bb:02
extensive table bgp.evpn.0

bgp.evpn.0: 52 destinations, 68 routes (52 active, 0 holddown, 0 hidden)
* 2:10.1.255.111:1::101::56:04:15:00:bb:02/304 MAC/IP (2 entries, 1 announced)
  Import Accepted
  Route Distinguisher: 10.1.255.111:1
  Route Label: 101
  ESI: 00:00:00:00:00:00:00:00:00:00
  Nexthop: 10.1.255.111
  Localpref: 100
  AS path: I
  Communities: target:65000:101 encapsulation:vlan(0x8)

* 2:10.1.255.111:1::101::56:04:15:00:bb:02::10.1.101.101/304 MAC/IP (2 entries, 1 announced)
  Import Accepted
  Route Distinguisher: 10.1.255.111:1
  Route Label: 101
  ESI: 00:00:00:00:00:00:00:00:00:00
  Nexthop: 10.1.255.111
  Localpref: 100

```

```
AS path: I
Communities: target:65000:101 encapsulation:vxlan(0x8)
```

The output shows the Type-2 routes for 56:04:15:00:bb:02. The route distinguisher is from Leaf-1 and is set to 10.1.255.111:1.

Verifying Imported Routes

Purpose

Verify that the EVPN Type-2 route is imported.

Action

On Core-1, verify whether EVPN Type-2 routes are successfully imported from the `bgp.evpn.0` table into the EVPN switch instance.

Meaning

The output shows that, in Tenant_A's virtual switch, the EVPN Type-2 route is advertised with the correct target, target:1:101. Use the extensive option to review the Type-2 route in greater detail.

```
lab@core-1> show route table VS_VLAN101.evpn.0 evpn-mac-address 56:04:15:00:bb:02
```

```
VS_VLAN101.evpn.0: 18 destinations, 25 routes (18 active, 0 holddown, 0 hidden)
```

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

```
2:10.1.255.111:1::101::56:04:15:00:bb:02/304 MAC/IP
```

```
*[BGP/170] 1w1d 20:50:01, localpref 100, from 10.1.255.111
```

```
AS path: I, validation-state: unverified
```

```
> to 10.1.11.2 via xe-0/2/0.0
```

```
[BGP/170] 3d 02:56:43, localpref 100, from 10.1.255.2
```

```
AS path: I, validation-state: unverified
```

```
> to 10.1.11.2 via xe-0/2/0.0
```

```
2:10.1.255.111:1::101::56:04:15:00:bb:02::10.1.101.101/304 MAC/IP
```

```
*[BGP/170] 1w1d 20:50:01, localpref 100, from 10.1.255.111
```

```
AS path: I, validation-state: unverified
```

```
> to 10.1.11.2 via xe-0/2/0.0
```

```
[BGP/170] 3d 02:56:43, localpref 100, from 10.1.255.2
```

```
AS path: I, validation-state: unverified
```

```
> to 10.1.11.2 via xe-0/2/0.0
```

The output shows that Core-1 receives two copies. The first is the advertisement from Leaf-1 (Source: 10.1.255.111). The second is the advertisement from Core-2 (Source: 10.1.255.2).

Verifying the Layer 2 Address Learning Daemon Copy

Purpose

Verify the Layer 2 address learning daemon copy.

Action

Verify the Layer 2 address learning daemon copy by entering the `show bridge-mac table` command.

Meaning

The output shows that 56:04:15:00:bb:02 is reachable through the vtep.32771 logical interface to Leaf-1.

```
lab@core-1> show bridge mac-table instance VS_VLAN101
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, FU - Fast Update)

Routing instance : VS_VLAN101
Bridging domain : bd101, VLAN : 101
```

MAC address	MAC flags	Logical interface	Active source
00:00:5e:00:01:01	DRP	esi.722	05:00:00:fd:e8:00:00:00:65:00
2c:6b:f5:54:95:f0	DR	vtep.32779	10.1.255.2
56:04:15:00:bb:02	DR	vtep.32771	10.1.255.111



NOTE: On EX9200 switches, the `show ethernet-switching table-instance instance-name` command corresponds to the `show bridge mac-table instance instance-name` command used here for MX Series routers

Verifying the Kernel-Level Forwarding Table

Purpose

Verify the kernel-level forwarding table, next hop identifier, and Layer 2 MAC table and hardware.

Action

Query the kernel-level forwarding table, correlate the index next hop identifier with the correct virtual network identifier, and review the Layer 2 MAC table and hardware.

Meaning

Tenant_A's MAC, 56:04:15:00:bb:02, is reachable through index 687.

```
lab@core-1> show route forwarding-table family bridge vpn VS_VLAN101
Routing table: VS_VLAN101.evpn-vxlan
VPLS:
Destination      Type RtRef Next hop      Type Index  NhRef Netif
default          perm   0              dscd   664    1
vtep.32771       intf   0              comp   687    7
vtep.32774       intf   0              comp   691    4
vtep.32779       intf   0              comp   716    7

Routing table: VS_VLAN101.evpn-vxlan
Bridging domain: bd101.evpn-vxlan
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,
Destination      Type RtRef Next hop      Type Index  NhRef Netif
00:00:5e:00:01:01/48 user    0              indr  1048579  2
                                comp    722    2
2c:6b:f5:54:95:f0/48 user    0              comp   716    7
56:04:15:00:bb:02/48 user    0              comp   687    7
0x30003/51       user    0              comp   705    2
```

Correlate index 687 (NH-Id) with the correct virtual network identifier 101 and remote VTEP-ID of 10.1.255.111.

```
lab@core-1> show 12-learning vxlan-tunnel-end-point remote
Logical System Name      Id  SVTEP-IP      IFL  L3-Idx  SVTEP-Mode  ELP-SVTEP-IP
<default>                0   10.1.255.1    lo0.0  0
```

RVTEP-IP	L2-RTT	IFL-Idx	Interface	NH-Id	RVTEP-Mode	ELP-
IP	Flags					
10.1.255.2	VS_VLAN101	377	vtep.32779	716	RNVE	
VNID	MC-Group-IP					
101	0.0.0.0					
RVTEP-IP	L2-RTT	IFL-Idx	Interface	NH-Id	RVTEP-Mode	ELP-
IP	Flags					
10.1.255.111	VS_VLAN101	369	vtep.32771	687	RNVE	
VNID	MC-Group-IP					
101	0.0.0.0					
RVTEP-IP	L2-RTT	IFL-Idx	Interface	NH-Id	RVTEP-Mode	ELP-
IP	Flags					
10.1.255.112	VS_VLAN101	372	vtep.32774	691	RNVE	
10.1.255.2	VS_VLAN102	376	vtep.32778	715	RNVE	
VNID	MC-Group-IP					
102	0.0.0.0					
RVTEP-IP	L2-RTT	IFL-Idx	Interface	NH-Id	RVTEP-Mode	ELP-
IP	Flags					
10.1.255.111	VS_VLAN102	370	vtep.32772	688	RNVE	
VNID	MC-Group-IP					
102	0.0.0.0					
RVTEP-IP	L2-RTT	IFL-Idx	Interface	NH-Id	RVTEP-Mode	ELP-
IP	Flags					
10.1.255.112	VS_VLAN102	373	vtep.32775	695	RNVE	
VNID	MC-Group-IP					
102	0.0.0.0					
RVTEP-IP	L2-RTT	IFL-Idx	Interface	NH-Id	RVTEP-Mode	ELP-
IP	Flags					
10.1.255.2	VS_VLAN103	375	vtep.32777	714	RNVE	
VNID	MC-Group-IP					
103	0.0.0.0					
RVTEP-IP	L2-RTT	IFL-Idx	Interface	NH-Id	RVTEP-Mode	ELP-
IP	Flags					
10.1.255.111	VS_VLAN103	371	vtep.32773	689	RNVE	
10.1.255.112	VS_VLAN103	374	vtep.32776	692	RNVE	
VNID	MC-Group-IP					
103	0.0.0.0					



NOTE: On EX9200 switches, the show ethernet-switching command corresponds to the show l2-learning command show here for MX Series routers.

Verifying MAC Reachability to a Multihomed CE Device

Purpose

Verify MAC reachability to the multihomed Tenant_B CE device on Leaf-1 and Leaf-2.

Action

Verify that Leaf-1 and Leaf-2 are advertising both Type-1 and Type-2 reachability towards the multihomed CE device.

```
lab@leaf-1> show ethernet-switching table vlan-id 102
```

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 4 entries, 4 learned

Routing instance : default-switch

Vlan name	MAC address	MAC flags	Logical interface	SVLBNH/ VENH Index	Active source
v102	00:00:5e:00:01:01	DR	esi.1748		
05:00:00:fd:e8:00:00:00:66:00					
v102	2c:6b:f5:43:12:c0	DL	ae0.0		
v102	2c:6b:f5:54:95:f0	D	vtep.32770		
10.1.255.2					
v102	2c:6b:f5:ef:73:f0	D	vtep.32769		
10.1.255.1					

```
lab@leaf-2>
```

```
show ethernet-switching table vlan-id 102
```

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 4 entries, 4 learned

Routing instance : default-switch

Vlan name	MAC address	MAC flags	Logical interface	SVLBNH/ VENH Index	Active source
v102	00:00:5e:00:01:01	DR	esi.1749		
05:00:00:fd:e8:00:00:00:66:00					

v102	2c:6b:f5:43:12:c0	DR	ae0.0
v102	2c:6b:f5:54:95:f0	D	vtep.32769
10.1.255.2			
v102	2c:6b:f5:ef:73:f0	D	vtep.32770
10.1.255.1			

Meaning

The output shows that 2c:6b:f5:43:12:c0 represents the MAC of the Tenant_B attached to Leaf-1 and Leaf-2.

Verifying EVPN, Layer 2 Address Learning Daemon, and the Kernel-Forwarding Tables for Multihomed CE Device

Purpose

Verify the Tenant B's EVPN table, and Core-1's Layer 2 address learning daemon table and kernel-forwarding table.

Action

In Core-1, display the Tenant B's EVPN table.

```
lab@core-1> show route table VS_VLAN102.evpn.0
```

```
VS_VLAN102.evpn.0: 20 destinations, 29 routes (20 active, 0 holddown, 0 hidden)
```

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

```
1:10.1.255.2:0::050000fde80000006600::FFFF:FFFF/192 AD/ESI
```

```
*[BGP/170] 2d 23:43:32, localpref 100, from 10.1.255.2
```

```
AS path: I, validation-state: unverified
```

```
to 10.1.11.2 via xe-0/2/0.0
```

```
> to 10.1.12.2 via xe-0/2/1.0
```

```
1:10.1.255.111:0::0101010101010101::FFFF:FFFF/192 AD/ESI
```

```
*[BGP/170] 00:14:59, localpref 100, from 10.1.255.111
```

```
AS path: I, validation-state: unverified
```

```
> to 10.1.11.2 via xe-0/2/0.0
```

```
to 10.1.12.2 via xe-0/2/1.0
```

```
[BGP/170] 00:14:58, localpref 100, from 10.1.255.2
```

```
AS path: I, validation-state: unverified
```

```
> to 10.1.11.2 via xe-0/2/0.0
```

```

        to 10.1.12.2 via xe-0/2/1.0
1:10.1.255.111:1::0101010101010101::0/192 AD/EVI
    *[BGP/170] 00:15:00, localpref 100, from 10.1.255.111
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
    [BGP/170] 00:14:59, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
1:10.1.255.112:0::0101010101010101::FFFF:FFFF/192 AD/ESI
    *[BGP/170] 00:10:13, localpref 100, from 10.1.255.112
        AS path: I, validation-state: unverified
        to 10.1.11.2 via xe-0/2/0.0
    > to 10.1.12.2 via xe-0/2/1.0
    [BGP/170] 00:10:13, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
        to 10.1.11.2 via xe-0/2/0.0
    > to 10.1.12.2 via xe-0/2/1.0
1:10.1.255.112:1::0101010101010101::0/192 AD/EVI
    *[BGP/170] 00:10:14, localpref 100, from 10.1.255.112
        AS path: I, validation-state: unverified
        to 10.1.11.2 via xe-0/2/0.0
    > to 10.1.12.2 via xe-0/2/1.0
    [BGP/170] 00:10:14, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
        to 10.1.11.2 via xe-0/2/0.0
    > to 10.1.12.2 via xe-0/2/1.0
2:10.1.255.1:102::102::00:00:5e:00:01:01/304 MAC/IP
    *[EVPN/170] 2d 23:44:03
        Indirect
2:10.1.255.1:102::102::2c:6b:f5:ef:73:f0/304 MAC/IP
    *[EVPN/170] 2d 23:44:03
        Indirect
2:10.1.255.2:102::102::00:00:5e:00:01:01/304 MAC/IP
    *[BGP/170] 2d 23:43:32, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
        to 10.1.11.2 via xe-0/2/0.0
    > to 10.1.12.2 via xe-0/2/1.0
2:10.1.255.2:102::102::2c:6b:f5:54:95:f0/304 MAC/IP
    *[BGP/170] 2d 23:43:32, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0

```



```

        to 10.1.12.2 via xe-0/2/1.0
2:10.1.255.111:1::102::2c:6b:f5:43:12:c0/304 MAC/IP
    *[BGP/170] 00:14:49, localpref 100, from 10.1.255.111
        AS path: I, validation-state: unverified
        to 10.1.11.2 via xe-0/2/0.0
    > to 10.1.12.2 via xe-0/2/1.0
    [BGP/170] 00:14:49, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
        to 10.1.11.2 via xe-0/2/0.0
    > to 10.1.12.2 via xe-0/2/1.0
2:10.1.255.112:1::102::2c:6b:f5:43:12:c0/304 MAC/IP
    *[BGP/170] 00:09:24, localpref 100, from 10.1.255.112
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
    [BGP/170] 00:09:24, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
2:10.1.255.1:102::102::00:00:5e:00:01:01::10.1.102.254/304 MAC/IP
    *[EVPN/170] 2d 23:44:03
        Indirect
2:10.1.255.1:102::102::2c:6b:f5:ef:73:f0::10.1.102.1/304 MAC/IP
    *[EVPN/170] 2d 23:44:03
        Indirect
2:10.1.255.2:102::102::00:00:5e:00:01:01::10.1.102.254/304 MAC/IP
    *[BGP/170] 2d 23:43:32, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
2:10.1.255.2:102::102::2c:6b:f5:54:95:f0::10.1.102.2/304 MAC/IP
    *[BGP/170] 2d 23:43:32, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
2:10.1.255.112:1::102::2c:6b:f5:43:12:c0::10.1.102.101/304 MAC/IP
    *[BGP/170] 00:06:19, localpref 100, from 10.1.255.112
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
    [BGP/170] 00:06:18, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0

```

```

        to 10.1.12.2 via xe-0/2/1.0
3:10.1.255.1:102::102::10.1.255.1/248 IM
    *[EVPN/170] 2d 23:45:49
        Indirect
3:10.1.255.2:102::102::10.1.255.2/248 IM
    *[BGP/170] 2d 23:44:03, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
3:10.1.255.111:1::102::10.1.255.111/248 IM
    *[BGP/170] 00:14:58, localpref 100, from 10.1.255.111
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
    [BGP/170] 00:14:58, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
3:10.1.255.112:1::102::10.1.255.112/248 IM
    *[BGP/170] 00:10:17, localpref 100, from 10.1.255.112
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0
    [BGP/170] 00:10:17, localpref 100, from 10.1.255.2
        AS path: I, validation-state: unverified
    > to 10.1.11.2 via xe-0/2/0.0
        to 10.1.12.2 via xe-0/2/1.0

```

Display Core-1's Layer 2 address learning daemon table.

```
lab@core-1> show bridge mac-table instance VS_VLAN102
```

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, FU - Fast Update)

Routing instance : VS_VLAN102

Bridging domain : bd102, VLAN : 102

MAC address	MAC flags	Logical interface	Active source
00:00:5e:00:01:01	DRP	esi.708	05:00:00:fd:e8:00:00:00:66:00

2c:6b:f5:43:12:c0	DR	esi.719	00:01:01:01:01:01:01:01:01
2c:6b:f5:54:95:f0	DR	vtep.32772	10.1.255.2



NOTE: On EX9200 switches, the `show ethernet-switching table-instance instance-name` command corresponds to the `show bridge mac-table instance instance-name` command [show here](#) for MX Series routers

Display Core-1's kernel forwarding table.

```
lab@core-1> show route forwarding-table vpn VS_VLAN102
```

```
Routing table: VS_VLAN102.evpn-vxlan
```

```
VPLS:
```

Destination	Type	RtRef	Next hop	Type	Index	NhRef	Netif
default	perm	0		dscd	544	1	
vtep.32772	intf	0		comp	688	7	
vtep.32775	intf	0		comp	716	5	
vtep.32778	intf	0		comp	722	5	

```
Routing table: VS_VLAN102.evpn-vxlan
```

```
Bridging domain: bd102.evpn-vxlan
```

```
VPLS:
```

```
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,
```

Destination	Type	RtRef	Next hop	Type	Index	NhRef	Netif
00:00:5e:00:01:01/48	user	0		indr	1048574	2	
				comp	708	2	
2c:6b:f5:43:12:c0/48	user	0		indr	1048578	3	
				comp	719	2	
2c:6b:f5:54:95:f0/48	user	0		comp	688	7	
0x30004/51	user	0		comp	702	2	

Meaning

For the Tenant_B CE device, four different routes are listed for ESI 00:01:01:01:01:01:01:01:

- 1:10.1.255.111:0:0101010101010101::FFFF:FFFF/192 AD/ESI

This per-Ethernet Segment A-D Type-1 EVPN route originated from Leaf-1. The route distinguisher is obtained from global-level routing-options. Core-1 receives this Type-1 route, originated from Leaf-1, from both Leaf-1 and Leaf-2.

- 1:10.1.255.111:1:0101010101010101::0/192 AD/EVI

This is the per-EVI A-D Type-1 EVPN route. The route distinguisher is obtained from the routing instance, or in the case of QFX5100, the switch-options. Core-1 receives this Type-1 route, originated from Leaf-1, from both Leaf-1 and Leaf-2.

- 1:10.1.255.112:0:0101010101010101::FFFF:FFFF/192 AD/ESI

This is the per-Ethernet Segment A-D Type-1 EVPN route originated from Leaf-2. The route distinguisher is obtained from global-level routing-options. Core-1 receives this Type-1 route, originated from Leaf-2, from both Leaf-2 and Leaf-1.

- 1:10.1.255.112:1:0101010101010101::0/192 AD/EVI

This is the per-EVI A-D Type-1 EVPN route. The route distinguisher is obtained from the routing instance, or in the case of QFX5100, switch-options. Core-1 receives this Type-1 route, originated from Leaf-2, from both Leaf-2 and Leaf-1.

The Type-2 routes for the two physical and one virtual MAC associated with the Tenant_B multihomed CE device are originated as expected.

From the output we cannot yet determine what VTEPs are used to forward to ESI 00:01:01:01:01:01:01:01:01. To determine the VTEPS, display the VXLAN tunnel endpoint ESIs.


```
lab@core-1> show l2-learning vxlan-tunnel-end-point esi
```

ESI	RTT	VLNBH	INH	ESI-IFL	LOC-IFL	#RVTEPs
00:01:01:01:01:01:01:01	VS_VLAN101	718	1048577	esi.718		2

Aliasing

RVTEP-IP	RVTEP-IFL	VENH	MASK-ID	FLAGS	MAC-COUNT
10.1.255.112	vtep.32779	723	1	2	0
10.1.255.111	vtep.32774	714	0	2	0

...



NOTE: On EX9200 switches, the show ethernet-switching command corresponds to the show l2-learning command show here for MX Series routers.

The output shows active load-balancing on the VTEP interfaces to both Leaf-1 and Leaf-2 for the MAC addresses on this ESI, which validates the all-active configuration on Leaf-1 and Leaf-2.

RELATED DOCUMENTATION

Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric	771
Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway	845

[Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric With a Virtual Gateway | 866](#)

[Understanding EVPN with VXLAN Data Plane Encapsulation | 588](#)

[EVPN Primer](#)

Configuring an EVPN-VXLAN Edge-Routed Bridging Overlay

IN THIS CHAPTER

- [Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway | 845](#)
- [Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric With a Virtual Gateway | 866](#)
- [EVPN-VXLAN Pure Type 5 Host-Route Auto-Generated Community | 894](#)

Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway

IN THIS SECTION

- [Requirements | 846](#)
- [Overview and Topology | 847](#)
- [Configuration For Leaf1 | 850](#)
- [Verification | 855](#)
- [Quick Configuration For All Devices | 860](#)

Ethernet VPN (EVPN) is a BGP-based control plane technology that enables hosts (physical servers and virtual machines) to be placed anywhere in a network and remain connected to the same logical Layer 2 (L2) overlay network. Virtual Extensible LAN (VXLAN) is a tunneling protocol that creates the data plane for the L2 overlay network.

The physical underlay network over which EVPN-VXLAN is commonly deployed is a two-layer IP fabric, which includes spine and leaf devices as shown in [Figure 76 on page 846](#). A two-layer spine and leaf fabric is referred to as a 3-stage Clos.

This example details how to deploy an edge-routed bridging (ERB) architecture using a 3-stage Clos fabric. In this design, the spine devices (such as QFX10000 switches) provide only IP connectivity between the leaf devices. In this capacity we call the spine devices lean spines, as they require no VXLAN functionality. The leaf devices (such as QFX5100 switches) provide connectivity to attached workloads. In the ERB case, the leaf devices provide L2 and Layer 3 (L3) VXLAN functionality in the overlay network. L2 gateways provide bridging within the same VLAN. An L3 gateway handles traffic between VLANs (inter-VLAN), using integrated routing and bridging (IRB) interfaces.

In this example, the IRB interfaces are configured with an anycast IP address. For an ERB example that uses virtual gateway address (VGA) IP address, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric With a Virtual Gateway"](#) on page 866

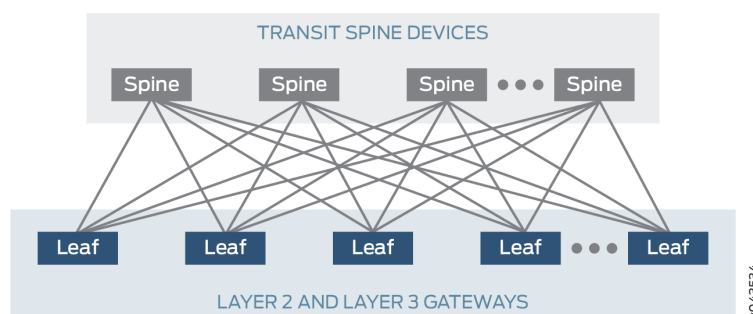


NOTE: We also call the ERB architecture a "collapsed" fabric. Compared with a CRB design, the L2 and L3 VXLAN gateway functions collapse into a single layer of the fabric (the leaves).

For background on EVPN-VXLAN technology and supported architectures, see [EVPN Primer](#).

For an example of how to configure an EVPN-VXLAN centrally-routed bridging (CRB) overlay, see ["Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric"](#) on page 771.

Figure 76: A 3-Stage (Leaf-And-Spine) Edge-Routing Bridging Architecture



This example describes how to configure an EVPN-VXLAN ERB overlay. As a result, you configure routing instances and IRB interfaces on the leaf devices only.

Requirements

This example uses the following hardware and software components:

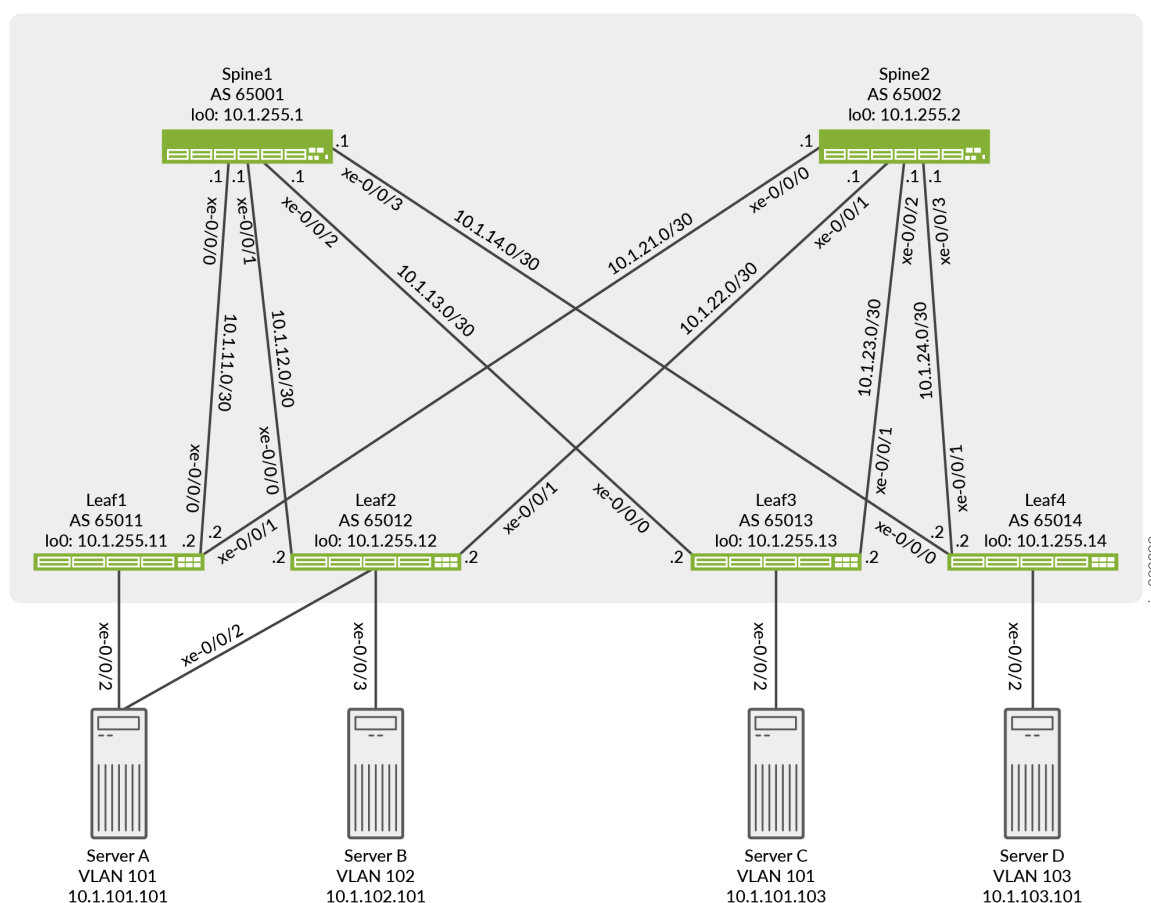
- Two devices that function as transit spine devices.
- Four devices running Junos OS Release 15.1X53-D60 or later software that serve as leaf devices and provide both L2 and L3 gateway functionality.

- Updated and re-validated using QFX10002 switches running Junos OS Release 21.3R1
- See the [hardware summary](#) for a list of supported platforms.

Overview and Topology

The ERB overlay shown in [Figure 77 on page 847](#) includes two transit spine devices and four leaf devices that function as both L2 and L3 gateways. Four servers are attached to the leaf devices. Server A is connected to Leaf1 and Leaf2 through a link aggregation group (LAG) interface. On both leaf devices, the interface is assigned the same Ethernet segment identifier (ESI) and set to multihoming all-active mode.

Figure 77: ERB Overlay within a Data Center



In this topology, Server A and Server C are in VLAN 101, Server B is in VLAN 102, and Server D is in VLAN 103. For communication between VLANs to occur, you must configure IRB interfaces for each VLAN on all leaf devices.

The most significant difference between the configuration of ERB compared with CRB is the configuration and location of the L3 gateway. Therefore, this example focuses on the EVPN-VXLAN configuration, in particular, the L3 gateway configuration, on the leaf devices.

For an ERB overlay, you can configure the IRB interfaces within an EVPN instance (EVI) using one of the following methods:

- **Method 1**—This method entails a unique IP address for each IRB interface, but uses the same MAC for each IRB interface. With this method a single MAC entry is installed for each IRB address on both the leaf devices and servers. For each IRB interface on a particular leaf device, for example, Leaf1, you specify the following:
 - A unique IP address for each IRB interface.
 - The *same* MAC address is used for each IRB interface.

For example:

Table 46: Unique IP Address with Same MAC per IRB Interface

irb.101	IP address: 10.1.101.254/24	MAC address: 00:00:5e:00:53:01
irb.102	IP address: 10.1.102.254/24	MAC address: 00:00:5e:00:53:01
irb.103	IP address: 10.1.103.254/24	MAC address: 00:00:5e:00:53:01

- **Method 2**—This method entails a unique IP address and MAC for each IRB interface. With this method, a MAC entry is installed for each IRB address on the leaf devices but only a single MAC on the servers. For each IRB interface on Leaf1, you specify the following:
 - A unique IP address for each IRB interface.
 - A *unique* MAC address is used for each IRB interface..

For example:

Table 47: Unique IP Address and MAC per IRB Interface

irb.101	IP address: 10.1.101.254/24	MAC address: 00:00:5e:00:53:01
irb.102	IP address: 10.1.102.254/24	MAC address: 00:00:5e:00:53:02

irb.103	IP address: 10.1.103.254/24	MAC address: 00:00:5e:00:53:03
---------	-----------------------------	--------------------------------

- **Method 3**—This method entails a unique IP address and VGA for each IRB interface. With this method a MAC entry is installed for each IRB address and the VGA on the leaf devices and servers. For each IRB interface on Leaf1, you specify the following:
 - A unique IP address for each IRB interface.
 - A *unique* VGA address for each IRB interface.

For example:

Table 48: Unique IP Address and Virtual Gateway Address per IRB Interface

irb.101	IP address: 10.1.101.1/24	VGA address: 10.1.101.254
irb.102	IP address: 10.1.102.1/24	VGA address: 10.1.102.254
irb.103	IP address: 10.1.103.1/24	VGA address: 10.1.103.254

For methods 1 and 2, the same IRB interface configuration is applied across all leaf devices. For method 3, a unique IRB interface address and the same VGA is applied across all leaf devices. In this example, method 1 is used to configure the IRB interfaces.

This example (with method 1) configures the same MAC address for each IRB interface on each leaf device. Each host uses the same MAC address when sending inter-VLAN traffic regardless of where the host is located or which leaf device receives the traffic. For example, in the topology shown in [Figure 77 on page 847](#), multi-homed Server A in VLAN 101 sends a packet to Server B in VLAN 102. If Leaf1 is down, Leaf2 continues to forward the inter-VLAN traffic even without the configuration of a redundant default gateway MAC address.



NOTE: The IRB interfaces configuration in this example doesn't include a virtual gateway address (VGA) and a corresponding V-MAC address that establishes redundant default gateway functionality, which is mentioned above. By configuring the same MAC address for each IRB interface on each leaf device, hosts use the local leaf device configured with the common MAC address as the default L3 gateway.

Therefore, you eliminate the need to advertise a redundant default gateway and dynamically synchronize the MAC addresses of the redundant default gateway

throughout the EVPN control plane. As a result, when configuring each leaf device, you must disable the advertisement of the redundant default gateway by including the `default-gateway do-not-advertise` configuration statement at the `[edit protocols evpn]` hierarchy level in your configuration.

Also, although the IRB interface configuration used in this example does not include a VGA, you can configure a VGA as needed to make EVPN-VXLAN work properly in your ERB overlay. If you configure a VGA for each IRB interface, you specify the same IP address for each VGA on each leaf device instead of configuring the same MAC address for each IRB interface on each leaf device as is shown in this example.

When it comes to handling the replication of broadcast, unknown unicast, and multicast (BUM) traffic, note that the configuration on Leaf1:

- includes the `set protocols evpn multicast-mode ingress-replication` command. This command causes Leaf1, which is a hardware VTEP, to handle replicating and sending BUM traffic instead of relying on a multicast-enabled underlay.

Configuration For Leaf1

IN THIS SECTION

- [CLI Quick Configuration | 850](#)
- [Configuring EVPN-VXLAN on Leaf1 | 852](#)

CLI Quick Configuration

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

Leaf1

```
set chassis aggregated-devices ethernet device-count 1
set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.21.2/30
set interfaces xe-0/0/2 ether-options 802.3ad ae0
set interfaces ae0 esi 00:01:01:01:01:01:01:01:01
set interfaces ae0 esi all-active
```

```

set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:01:01:01:01:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members v101
set interfaces irb unit 101 family inet address 10.1.101.254/24
set interfaces irb unit 101 mac 00:00:5e:00:53:01
set interfaces irb unit 102 family inet address 10.1.102.254/24
set interfaces irb unit 102 mac 00:00:5e:00:53:01
set interfaces irb unit 103 family inet address 10.1.103.254/24
set interfaces irb unit 103 mac 00:00:5e:00:53:01
set interfaces lo0 unit 0 family inet address 10.1.255.11/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.11/32 exact
set policy-options policy-statement send-direct term 1 then accept
set routing-instances vrf101 instance-type vrf
set routing-instances vrf101 interface irb.101
set routing-instances vrf101 interface irb.102
set routing-instances vrf101 interface irb.103
set routing-instances vrf101 route-distinguisher 10.1.255.11:101
set routing-instances vrf101 vrf-target target:1001:1
set routing-options router-id 10.1.255.11
set routing-options autonomous-system 65011
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002
set protocols bgp group overlay type external
set protocols bgp group overlay multihop
set protocols bgp group overlay local-address 10.1.255.11
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway do-not-advertise
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1
set protocols evpn vni-options vni 102 vrf-target target:102:1
set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all

```

```

set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.11:1
set switch-options vrf-target target:1:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Configuring EVPN-VXLAN on Leaf1

Step-by-Step Procedure

1. Configure the underlay configuration. In this example we use EBGp for the underlay routing protocol.

```

[edit]
user@leaf1# set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.2/30
user@leaf1# set interfaces xe-0/0/1 unit 0 family inet address 10.1.21.2/30
user@leaf1# set interfaces lo0 unit 0 family inet address 10.1.255.11/32
user@leaf1# set policy-options policy-statement load-balancing-policy then load-balance per-
packet
user@leaf1# set policy-options policy-statement send-direct term 1 from protocol direct
user@leaf1# set policy-options policy-statement send-direct term 1 from route-filter
10.1.255.11/32 exact
user@leaf1# set policy-options policy-statement send-direct term 1 then accept
user@leaf1# set routing-options router-id 10.1.255.11
user@leaf1# set routing-options autonomous-system 65011
user@leaf1# set routing-options forwarding-table export load-balancing-policy
user@leaf1# set protocols bgp group underlay type external
user@leaf1# set protocols bgp group underlay export send-direct
user@leaf1# set protocols bgp group underlay multipath multiple-as
user@leaf1# set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
user@leaf1# set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002

```

2. Configure Server A to be multihomed to Leaf1 and Leaf2 by configuring an aggregated Ethernet interface, specifying an ESI for the interface, and setting the mode so that the connections to both leaf devices are active. We show the applied VLAN configuration in a later step.



NOTE: When configuring the AE interface on Leaf2, you must specify the same ESI (00:01:01:01:01:01:01:01) as the ESI for the same interface on Leaf1.

```
[edit]
user@leaf1# set chassis aggregated-devices ethernet device-count 1
user@leaf1# set interfaces xe-0/0/2 ether-options 802.3ad ae0
user@leaf1# set interfaces ae0 esi 00:01:01:01:01:01:01:01
user@leaf1# set interfaces ae0 esi all-active
user@leaf1# set interfaces ae0 aggregated-ether-options lacp active
user@leaf1# set interfaces ae0 aggregated-ether-options lacp system-id 00:01:01:01:01:01
user@leaf1# set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
user@leaf1# set interfaces ae0 unit 0 family ethernet-switching vlan members v101
```

3. Configure the IRB interfaces, each with unique IP addresses and the same MAC address.



NOTE: Each leaf device should have the same IRB interface configuration.

```
[edit]
user@leaf1# set interfaces irb unit 101 family inet address 10.1.101.254/24
user@leaf1# set interfaces irb unit 101 mac 00:00:5e:00:53:01
user@leaf1# set interfaces irb unit 102 family inet address 10.1.102.254/24
user@leaf1# set interfaces irb unit 102 mac 00:00:5e:00:53:01
user@leaf1# set interfaces irb unit 103 family inet address 10.1.103.254/24
user@leaf1# set interfaces irb unit 103 mac 00:00:5e:00:53:01
```

4. Set up the EBGp-based overlay configuration. Make sure to include the multihop configuration option because we use loopback peering.

```
[edit]
user@leaf1# set protocols bgp group overlay type external
user@leaf1# set protocols bgp group overlay multihop
user@leaf1# set protocols bgp group overlay local-address 10.1.255.11
user@leaf1# set protocols bgp group overlay family evpn signaling
```

```

user@leaf1# set protocols bgp group overlay multipath multiple-as
user@leaf1# set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
user@leaf1# set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002

```



NOTE: Some IP fabrics use an IBGP based EVPN-VXLAN overlay. For an example of an IP fabric that uses IBGP for the overlay, see ["Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric" on page 771](#). Note that choosing EBGP or IBGP for the overlay does not impact the fabric architecture. Both CRB and ERB designs support either type of overlay.

5. Set up the EVPN-VXLAN domain, which entails determining which VNIs are included in the domain, specifying that Leaf1, which is a hardware VTEP, handles the replication and sending of BUM traffic, disabling the advertisement of the redundant default gateway throughout the EVPN control plane, and specifying a route target for each VNI.

```

[edit]
user@leaf1# set protocols evpn encapsulation vxlan
user@leaf1# set protocols evpn default-gateway do-not-advertise
user@leaf1# set protocols evpn multicast-mode ingress-replication
user@leaf1# set protocols evpn vni-options vni 101 vrf-target target:101:1
user@leaf1# set protocols evpn vni-options vni 102 vrf-target target:102:1
user@leaf1# set protocols evpn vni-options vni 103 vrf-target target:103:1
user@leaf1# set protocols evpn extended-vni-list all

```

6. Set up an EVPN routing instance.

```

[edit]
user@leaf1# set routing-instances vrf101 instance-type vrf
user@leaf1# set routing-instances vrf101 interface irb.101
user@leaf1# set routing-instances vrf101 interface irb.102
user@leaf1# set routing-instances vrf101 interface irb.103
user@leaf1# set routing-instances vrf101 route-distinguisher 10.1.255.11:101
user@leaf1# set routing-instances vrf101 vrf-target target:1001:1

```

7. Configure the switch options to use loopback interface lo0.0 as the source interface of the VTEP, set a route distinguisher, and set the vrf target.

```
[edit]
user@leaf1# set switch-options vtep-source-interface lo0.0
user@leaf1# set switch-options route-distinguisher 10.1.255.11:1
user@leaf1# set switch-options vrf-target target:1:1
user@leaf1# set switch-options vrf-target auto
```

8. Configure VLANs associated with IRB interfaces and VXLAN VNIs.

```
[edit]
user@leaf1# set vlans v101 vlan-id 101
user@leaf1# set vlans v101 l3-interface irb.101
user@leaf1# set vlans v101 vxlan vni 101
user@leaf1# set vlans v102 vlan-id 102
user@leaf1# set vlans v102 l3-interface irb.102
user@leaf1# set vlans v102 vxlan vni 102
user@leaf1# set vlans v103 vlan-id 103
user@leaf1# set vlans v103 l3-interface irb.103
user@leaf1# set vlans v103 vxlan vni 103
```

Verification

IN THIS SECTION

- [Verifying BGP | 856](#)
- [Verifying the ESI | 857](#)
- [Verifying the EVPN Database | 858](#)
- [Verifying Connectivity | 859](#)

The section describes the following verifications for this example:

Verifying BGP

Purpose

Verify that the spine devices have established BGP session connectivity.

Action

Display the BGP summary:

```

user@leaf1> show bgp summary
Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 2 Peers: 4 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State   Pending
bgp.evpn.0
              76         38         0         0         0         0
inet.0
              8         8         0         0         0         0
Peer          AS      InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.1.11.1      65001    3826    3867      0       0 1d 5:14:55 Establ
  inet.0: 4/4/4/0
10.1.21.1      65002    3829    3871      0       0 1d 5:14:55 Establ
  inet.0: 4/4/4/0
10.1.255.1     65001    4321    4228      0       0 1d 5:14:52 Establ
  bgp.evpn.0: 14/38/38/0
  default-switch.evpn.0: 13/37/37/0
  __default_evpn__.evpn.0: 1/1/1/0
10.1.255.2     65002    4385    4169      0       0 1d 5:14:53 Establ
  bgp.evpn.0: 24/38/38/0
  default-switch.evpn.0: 24/37/37/0
  __default_evpn__.evpn.0: 0/1/1/0

```

Meaning

Both underlay and overlay BGP sessions are established with the spine devices.

Verifying the ESI

Purpose

Verify the status of the ESI.

Action

Display the status of the ESI:

```

user@leaf1> show evpn instance esi 00:01:01:01:01:01:01:01:01 extensive
Instance: default-switch
Route Distinguisher: 10.1.255.11:1
Encapsulation type: VXLAN
Duplicate MAC detection threshold: 5
Duplicate MAC detection window: 180
MAC database status
MAC advertisements:
MAC+IP advertisements:
Default gateway MAC advertisements:
Number of local interfaces: 2 (2 up)
Interface name ESI Mode Status AC-Role
.local..5 00:00:00:00:00:00:00:00:00 single-homed Up Root
ae0.0 00:01:01:01:01:01:01:01:01 all-active Up Root
Number of IRB interfaces: 3 (3 up)
Interface name VLAN VNI Status L3 context
irb.101 101 Up vrf101
irb.102 102 Up vrf101
irb.103 103 Up vrf101
Number of protect interfaces: 0
Number of bridge domains: 3
VLAN Domain-ID Intfs/up IRB-intf Mode MAC-sync IM-label MAC-label v4-SG-
sync IM-core-NH v6-SG-sync IM-core-NH Trans-ID
101 101 1 1 irb.101 Extended Enabled 101
Disabled Disabled 101
102 102 0 0 irb.102 Extended Enabled 102
Disabled Disabled 102
103 103 0 0 irb.103 Extended Enabled 103
Disabled Disabled 103
Number of neighbors: 3
Address MAC MAC+IP AD IM ES Leaf-label Remote-DCI-Peer
10.1.255.12 2 2 2 3 0

```

```

10.1.255.13      1      1      0      3      0
10.1.255.14      1      1      0      3      0
Number of ethernet segments: 4
ESI: 00:01:01:01:01:01:01:01:01
Status: Resolved by IFL ae0.0
Local interface: ae0.0, Status: Up/Forwarding
Number of remote PEs connected: 1
  Remote-PE      MAC-label  Aliasing-label  Mode
  10.1.255.12    101          0              all-active
DF Election Algorithm: MOD based
Designated forwarder: 10.1.255.12
Backup forwarder: 10.1.255.11
Last designated forwarder update: May 12 10:33:14
Router-ID: 10.1.255.11
Source VTEP interface IP: 10.1.255.11
SMET Forwarding: Disabled

```

Meaning

The ESI is up and Leaf2 is the remote provider edge (PE) device and the designated forwarder.

Verifying the EVPN Database

Purpose

Verify the MAC addresses in the EVPN database.

Action

Verify the MAC addresses in the EVPN database for VLAN 101.

```

user@leaf1> show evpn database l2-domain-id 101
Instance: default-switch
VLAN  DomainId  MAC address      Active source      Timestamp      IP address
  101      00:00:5e:00:53:01  irb.101          May 12 16:40:47  10.1.101.254
  101      2c:6b:f5:1b:6e:c1  00:01:01:01:01:01:01:01:01  May 12 17:26:30  10.1.101.101
  101      56:04:15:00:af:fa  10.1.255.13      May 12 16:40:46  10.1.101.103

```

Meaning

The MAC and IP addresses for Server A are shown with an active source of the ESI, and the MAC and IP addresses for server C are shown with an active source from Leaf3.

Verifying Connectivity

Purpose

Verify ping works between servers.

Action

Ping from server A to the other servers.

```
user@serverA> ping 10.1.102.101 count 2
PING 10.1.102.101 (10.1.102.101): 56 data bytes
64 bytes from 10.1.102.101: icmp_seq=0 ttl=63 time=117.425 ms
64 bytes from 10.1.102.101: icmp_seq=1 ttl=63 time=109.663 ms

--- 10.1.102.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 109.663/113.544/117.425/3.881 ms

user@serverA> ping 10.1.101.103 count 2
PING 10.1.101.103 (10.1.101.103): 56 data bytes
64 bytes from 10.1.101.103: icmp_seq=0 ttl=64 time=311.050 ms
64 bytes from 10.1.101.103: icmp_seq=1 ttl=64 time=201.300 ms

--- 10.1.101.103 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 201.300/256.175/311.050/54.875 ms

user@serverA> ping 10.1.103.101 count 2
PING 10.1.103.101 (10.1.103.101): 56 data bytes
64 bytes from 10.1.103.101: icmp_seq=0 ttl=63 time=311.321 ms
64 bytes from 10.1.103.101: icmp_seq=1 ttl=63 time=367.343 ms

--- 10.1.103.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 311.321/339.332/367.343/28.011 ms
```

Meaning

End-to-end connectivity is working.

Quick Configuration For All Devices

IN THIS SECTION

- [CLI Quick Configuration | 860](#)

CLI Quick Configuration

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

Leaf2

```
set chassis aggregated-devices ethernet device-count 1
set interfaces xe-0/0/0 unit 0 family inet address 10.1.12.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.22.2/30
set interfaces xe-0/0/2 ether-options 802.3ad ae0
set interfaces xe-0/0/3 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/3 unit 0 family ethernet-switching vlan members v102
set interfaces ae0 esi 00:01:01:01:01:01:01:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:01:01:01:01:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members v101
set interfaces irb unit 101 family inet address 10.1.101.254/24
set interfaces irb unit 101 mac 00:00:5e:00:53:01
set interfaces irb unit 102 family inet address 10.1.102.254/24
set interfaces irb unit 102 mac 00:00:5e:00:53:01
set interfaces irb unit 103 family inet address 10.1.103.254/24
set interfaces irb unit 103 mac 00:00:5e:00:53:01
set interfaces lo0 unit 0 family inet address 10.1.255.12/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.12/32 exact
```

```

set policy-options policy-statement send-direct term 1 then accept
set routing-instances vrf101 instance-type vrf
set routing-instances vrf101 interface irb.101
set routing-instances vrf101 interface irb.102
set routing-instances vrf101 interface irb.103
set routing-instances vrf101 route-distinguisher 10.1.255.12:101
set routing-instances vrf101 vrf-target target:1001:1
set routing-options router-id 10.1.255.12
set routing-options autonomous-system 65012
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.12.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.22.1 peer-as 65002
set protocols bgp group overlay type external
set protocols bgp group overlay multihop
set protocols bgp group overlay local-address 10.1.255.12
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway do-not-advertise
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1
set protocols evpn vni-options vni 102 vrf-target target:102:1
set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.12:1
set switch-options vrf-target target:1:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Leaf3

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.13.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.23.2/30
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members v101
set interfaces irb unit 101 family inet address 10.1.101.254/24
set interfaces irb unit 101 mac 00:00:5e:00:53:01
set interfaces irb unit 102 family inet address 10.1.102.254/24
set interfaces irb unit 102 mac 00:00:5e:00:53:01
set interfaces irb unit 103 family inet address 10.1.103.254/24
set interfaces irb unit 103 mac 00:00:5e:00:53:01
set interfaces lo0 unit 0 family inet address 10.1.255.13/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.13/32 exact
set policy-options policy-statement send-direct term 1 then accept
set routing-instances vrf101 instance-type vrf
set routing-instances vrf101 interface irb.101
set routing-instances vrf101 interface irb.102
set routing-instances vrf101 interface irb.103
set routing-instances vrf101 route-distinguisher 10.1.255.13:101
set routing-instances vrf101 vrf-target target:1001:1
set routing-options router-id 10.1.255.13
set routing-options autonomous-system 65013
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.13.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.23.1 peer-as 65002
set protocols bgp group overlay type external
set protocols bgp group overlay multihop
set protocols bgp group overlay local-address 10.1.255.13
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway do-not-advertise
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1

```

```

set protocols evpn vni-options vni 102 vrf-target target:102:1
set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.13:1
set switch-options vrf-target target:1:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Leaf4

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.14.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.24.2/30
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members v103
set interfaces irb unit 101 family inet address 10.1.101.254/24
set interfaces irb unit 101 mac 00:00:5e:00:53:01
set interfaces irb unit 102 family inet address 10.1.102.254/24
set interfaces irb unit 102 mac 00:00:5e:00:53:01
set interfaces irb unit 103 family inet address 10.1.103.254/24
set interfaces irb unit 103 mac 00:00:5e:00:53:01
set interfaces lo0 unit 0 family inet address 10.1.255.14/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.14/32 exact
set policy-options policy-statement send-direct term 1 then accept
set routing-instances vrf103 instance-type vrf
set routing-instances vrf103 interface irb.101
set routing-instances vrf103 interface irb.102
set routing-instances vrf103 interface irb.103
set routing-instances vrf103 route-distinguisher 10.1.255.14:101
set routing-instances vrf103 vrf-target target:1001:1
set routing-options router-id 10.1.255.14
set routing-options autonomous-system 65014

```



```

set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.14.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.24.1 peer-as 65002
set protocols bgp group overlay type external
set protocols bgp group overlay multihop
set protocols bgp group overlay local-address 10.1.255.14
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway do-not-advertise
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1
set protocols evpn vni-options vni 102 vrf-target target:102:1
set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.14:1
set switch-options vrf-target target:1:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Spine 1

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.1/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.12.1/30
set interfaces xe-0/0/2 unit 0 family inet address 10.1.13.1/30
set interfaces xe-0/0/3 unit 0 family inet address 10.1.14.1/30
set interfaces lo0 unit 0 family inet address 10.1.255.1/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet

```

```

set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.1/32 exact
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.1
set routing-options autonomous-system 65001
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.2 peer-as 65011
set protocols bgp group underlay neighbor 10.1.12.2 peer-as 65012
set protocols bgp group underlay neighbor 10.1.13.2 peer-as 65013
set protocols bgp group underlay neighbor 10.1.14.2 peer-as 65014
set protocols bgp group overlay type external
set protocols bgp group overlay multihop no-nexthop-change
set protocols bgp group overlay local-address 10.1.255.1
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.11 peer-as 65011
set protocols bgp group overlay neighbor 10.1.255.12 peer-as 65012
set protocols bgp group overlay neighbor 10.1.255.13 peer-as 65013
set protocols bgp group overlay neighbor 10.1.255.14 peer-as 65014
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002

```

Spine 2

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.21.1/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.22.1/30
set interfaces xe-0/0/2 unit 0 family inet address 10.1.23.1/30
set interfaces xe-0/0/3 unit 0 family inet address 10.1.24.1/30
set interfaces lo0 unit 0 family inet address 10.1.255.2/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.2/32 exact
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.2
set routing-options autonomous-system 65002
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.21.2 peer-as 65011

```

```

set protocols bgp group underlay neighbor 10.1.22.2 peer-as 65012
set protocols bgp group underlay neighbor 10.1.23.2 peer-as 65013
set protocols bgp group underlay neighbor 10.1.24.2 peer-as 65014
set protocols bgp group overlay type external
set protocols bgp group overlay multihop no-nexthop-change
set protocols bgp group overlay local-address 10.1.255.2
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.11 peer-as 65011
set protocols bgp group overlay neighbor 10.1.255.12 peer-as 65012
set protocols bgp group overlay neighbor 10.1.255.13 peer-as 65013
set protocols bgp group overlay neighbor 10.1.255.14 peer-as 65014
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001

```

RELATED DOCUMENTATION

[Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric | 771](#)

[Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric Using MX Routers as Spines | 808](#)

[Understanding EVPN with VXLAN Data Plane Encapsulation | 588](#)

[EVPN Primer](#)

Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric With a Virtual Gateway

IN THIS SECTION

- [Requirements | 868](#)
- [Overview and Topology | 868](#)
- [Quick Configuration | 871](#)
- [Underlay Network Configuration | 873](#)
- [EVPN-VXLAN Overlay Network Configuration | 875](#)
- [Customer Profile Configuration | 877](#)
- [Route Leaking Configuration | 880](#)

- Verification | 882
- Quick Configuration For All Devices | 886

Ethernet VPN (EVPN) is a control plane technology that enables hosts (physical [bare-metal] servers and virtual machines [VMs]) to be placed anywhere in a network and remain connected to the same logical Layer 2 (L2) overlay network. Virtual Extensible LAN (VXLAN) is a tunneling protocol that creates the data plane for the L2 overlay network.

The physical underlay network over which EVPN-VXLAN is commonly deployed is a two-layer IP fabric, which includes spine and leaf devices as shown in [Figure 78 on page 868](#). A two-layer spine and leaf fabric is referred to as a 3-stage Clos.

This example details how to deploy an edge-routed bridging (ERB) architecture using a 3-stage Clos fabric. In this design the spine devices—for example, QFX10000 switches—provide only IP connectivity between the leaf devices. In this capacity they are referred to as lean spines, as they require no VXLAN functionality. The leaf devices—for example, QFX5100 switches—provide connectivity to attached workloads, and in the ERB case, provide L2 and Layer 3 (L3) VXLAN functionality in the overlay network. L2 gateways provide bridging within the same VLAN while a L3 gateway handles traffic between VLANs (inter-VLAN), through the use of integrated routing and bridging (IRB) interfaces.

In this example, we configure the IRB interfaces with a virtual gateway address (VGA). For an ERB example that uses an anycast IP address on the IRBs and more information about the different methods, see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway" on page 845](#).

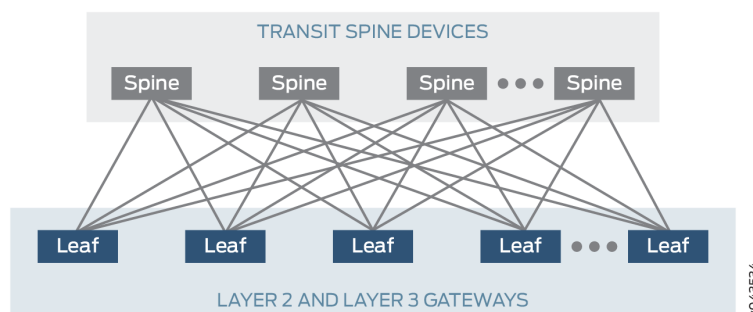


NOTE: The ERB architecture is sometimes called a "collapsed" fabric. This is because, when compared to a CEB design, the L2 and L3 VXLAN gateway functional is collapsed into a single layer of the fabric (the leaves).

For background information on EVPN-VXLAN technology and supported architectures, see [EVPN Primer](#).

For an example of how to configure an EVPN-VXLAN centrally-routed bridging (CRB) overlay see ["Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric" on page 771](#).

Figure 78: A 3-Stage (Leaf and Spine) Edge-Routing Bridging Architecture



Starting with Junos OS Release 17.3R1, the QFX5110 switch can function as a leaf device, which acts as L2 and L3 VXLAN gateways in an EVPN-VXLAN ERB overlay.

This topic provides a sample configuration of a QFX device that functions as a leaf in an ERB overlay.

Requirements

This example uses the following hardware and software components:

- Two devices that function as transit spine devices.
- Four devices running Junos OS Release 17.3R1 or later that serve as leaf devices and provide both L2 and L3 VXLAN gateway functionality.
 - Updated and re-validated using QFX10002 switches running Junos OS Release 21.3R1
- See the [hardware summary](#) for a list of supported platforms.

Overview and Topology

In this example, a service provider supports ABC Corporation, which has multiple servers. Server A and Server C communicate with each other using VLAN 101. Server B and Server D communicate using the L3 gateway. To enable this communication in the ERB overlay shown in [Figure 79 on page 869](#), you configure the key software entities in [Table 49 on page 869](#) on the switches that function as L2 and L3 VXLAN gateways, or leaf devices.

Figure 79: Sample Edge-Routed Bridging Overlay

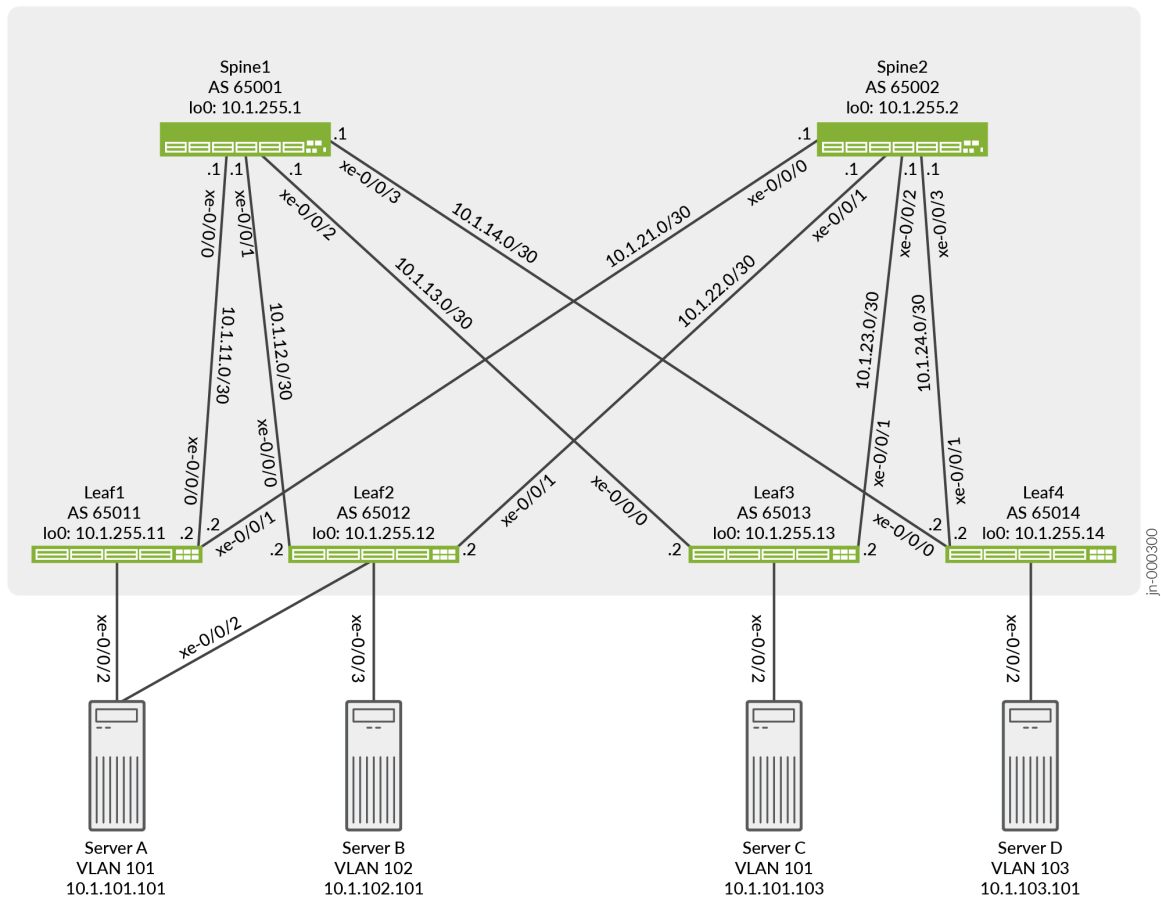


Table 49: Layer 3 Inter-VLAN Routing Entities Configured on Leaf1, Leaf2, Leaf3 and Leaf4

Entities	Configuration on Leaf1, Leaf2, Leaf3 and Leaf4
VLANs	v101 v102 v103
VRF instances	vrf101 vrf102_103

Table 49: Layer 3 Inter-VLAN Routing Entities Configured on Leaf1, Leaf2, Leaf3 and Leaf4 (Continued)

Entities	Configuration on Leaf1, Leaf2, Leaf3 and Leaf4
IRB interfaces	irb.101
	10.1.101.1/24 (IRB IP address)
	10.1.101.254 (virtual gateway address)
	irb.102
	10.1.102.1/24 (IRB IP address)
	10.1.102.254 (virtual gateway address)
	irb.103
	10.1.103.1/24 (IRB IP address)
	10.1.103.254 (virtual gateway address)

As outlined in [Table 49 on page 869](#), you configure VLAN v101 for Server A and Server C, VLAN v102 for Server B, and VLAN v103 for Server D on each leaf device. To segregate the L3 routes for VLANs v101, v102, and v103, you create VPN routing and forwarding (VRF) instances vrf101 and vrf102_103 on each leaf device. To route traffic between the VLANs, you configure IRB interfaces irb.101, irb.102, and irb.103. You also associate VRF instance vrf101 with IRB interface irb.101, and VRF instance vrf102_103 with IRB interfaces irb.102 and irb.103.

You configure IRB interfaces irb.101, irb.102, and irb.103 to function as default L3 gateways that handle server inter-VLAN traffic. To that end, in each IRB interface configuration, you also include a virtual gateway address (VGA), which configures an IRB interface as a default L3 gateway. In addition, this example assumes that each server is configured to use a particular default gateway. For more information about default gateways and how inter-VLAN traffic flows between a physical server to another physical server or VM in another VLAN in an ERB overlay, see ["Using a Default Layer 3 Gateway to Route Traffic in an EVPN-VXLAN Overlay Network" on page 741](#).



NOTE: When configuring a VGA for an IRB interface, keep in mind that the IRB IP address and VGA must be different.

Also, we require that you configure the no-gateway-community option at the [edit routing-instances *EVPN-instance-name* protocols evpn default-gateway] hierarchy level in each EVPN routing instance in which you configure an IRB interface with a virtual gateway address.

You can alternatively configure the `no-gateway-community` option globally at the `[edit protocols evpn default-gateway]` hierarchy level on platforms that support configuration at this level. See *default-gateway* for details on using the `no-gateway-community` option.

As outlined in [Table 49 on page 869](#), you configure a separate VRF routing instance for VLAN v101 and VLANs v102 and v103. To enable the communication between hosts in VLANs v101, v102, and v103, this example shows how to export unicast routes from the routing table for vrf101 and import the routes into the routing table for vrf102_103 and vice versa. This feature is also known as route leaking.

Quick Configuration

IN THIS SECTION

- [CLI Quick Configuration | 871](#)

CLI Quick Configuration

To quickly configure Leaf1, copy the following commands and paste them into a text file. Remove any line breaks and change any details necessary to match your network configuration. Then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

```
set chassis aggregated-devices ethernet device-count 1
set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.21.2/30
set interfaces xe-0/0/2 ether-options 802.3ad ae0
set interfaces ae0 esi 00:01:01:01:01:01:01:01:01:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:01:01:01:01:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members v101
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.1/24 virtual-gateway-address 10.1.101.254
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 family inet address 10.1.102.1/24 virtual-gateway-address 10.1.102.254
set interfaces irb unit 103 virtual-gateway-accept-data
```



```

set interfaces irb unit 103 family inet address 10.1.103.1/24 virtual-gateway-address
10.1.103.254
set interfaces lo0 unit 0 family inet address 10.1.255.11/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.11/32 exact
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement vrf101_vrf_imp term 1 from community vrf102_103
set policy-options policy-statement vrf101_vrf_imp term 1 then accept
set policy-options policy-statement vrf102_103_vrf_imp term 1 from community vrf101
set policy-options policy-statement vrf102_103_vrf_imp term 1 then accept
set policy-options community vrf101 members target:1001:1
set policy-options community vrf102_103 members target:1023:1
set routing-instances vrf101 instance-type vrf
set routing-instances vrf101 routing-options auto-export
set routing-instances vrf101 interface irb.101
set routing-instances vrf101 route-distinguisher 10.1.255.11:101
set routing-instances vrf101 vrf-import vrf101_vrf_imp
set routing-instances vrf101 vrf-target target:1001:1
set routing-instances vrf102_103 instance-type vrf
set routing-instances vrf102_103 routing-options auto-export
set routing-instances vrf102_103 interface irb.102
set routing-instances vrf102_103 interface irb.103
set routing-instances vrf102_103 route-distinguisher 10.1.255.11:123
set routing-instances vrf102_103 vrf-import vrf102_103_vrf_imp
set routing-instances vrf102_103 vrf-target target:1023:1
set routing-options router-id 10.1.255.11
set routing-options autonomous-system 65011
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002
set protocols bgp group overlay type external
set protocols bgp group overlay multihop
set protocols bgp group overlay local-address 10.1.255.11
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community

```

```

set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1
set protocols evpn vni-options vni 102 vrf-target target:102:1
set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.11:1
set switch-options vrf-target target:1:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Underlay Network Configuration

IN THIS SECTION

- [CLI Quick Configuration | 873](#)
- [Configuring the Underlay Network | 874](#)

CLI Quick Configuration

To quickly configure a underlay network on Leaf1, copy the following commands and paste them into a text file. Remove any line breaks and change any details necessary to match your network configuration. Then copy and paste the commands into the CLI at the [edit] hierarchy level.

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.21.2/30
set interfaces lo0 unit 0 family inet address 10.1.255.11/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.11/32 exact

```

```

set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.11
set routing-options autonomous-system 65011
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002

```

Configuring the Underlay Network

Step-by-Step Procedure

To configure a underlay network on Leaf1:

1. Configure the interfaces connected to the spine devices and the loopback interface on Leaf1.

```

user@leaf1# set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.2/30
user@leaf1# set interfaces xe-0/0/1 unit 0 family inet address 10.1.21.2/30
user@leaf1# set interfaces lo0 unit 0 family inet address 10.1.255.11/32

```

2. Configure the router ID, autonomous system number, and apply the load balancing policy for Leaf1. We show the policy configuration in a later step.

```

user@leaf1# set routing-options router-id 10.1.255.11
user@leaf1# set routing-options autonomous-system 65011
user@leaf1# set routing-options forwarding-table export load-balancing-policy

```

3. Configure an EBGp group that peers with both spine devices. We show the policy to advertise the loopback address for Leaf1 in a later step.

```

user@leaf1# set protocols bgp group underlay type external
user@leaf1# set protocols bgp group underlay export send-direct
user@leaf1# set protocols bgp group underlay multipath multiple-as
user@leaf1# set protocols bgp group underlay neighbor 10.1.11.1 peer-as 65001
user@leaf1# set protocols bgp group underlay neighbor 10.1.21.1 peer-as 65002

```

4. Configure policies to load balancing and advertise the loopback address of Leaf1.

```

user@leaf1# set policy-options policy-statement load-balancing-policy then load-balance per-
packet
user@leaf1# set policy-options policy-statement send-direct term 1 from protocol direct
user@leaf1# set policy-options policy-statement send-direct term 1 from route-filter
10.1.255.11/32 exact
user@leaf1# set policy-options policy-statement send-direct term 1 then accept

```

EVPN-VXLAN Overlay Network Configuration

IN THIS SECTION

- [CLI Quick Configuration | 875](#)
- [Configuring an EVPN-VXLAN Underlay Network | 876](#)

CLI Quick Configuration

To quickly configure an overlay network, copy the following commands and paste them into a text file. Remove any line breaks and change any details necessary to match your network configuration. Then copy and paste the commands into the CLI at the [edit] hierarchy level.

```

set protocols bgp group overlay type external
set protocols bgp group overlay multihop
set protocols bgp group overlay local-address 10.1.255.11
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1
set protocols evpn vni-options vni 102 vrf-target target:102:1
set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.11:1

```

```
set switch-options vrf-target target:1:1
set switch-options vrf-target auto
```

Configuring an EVPN-VXLAN Underlay Network

Step-by-Step Procedure

To configure a basic EVPN-VXLAN overlay network on Leaf1:

1. Configure an EBGp-based overlay between Leaf1 and the spine devices, specify a local IP address for Leaf1, and include the EVPN signaling Network Layer Reachability Information (NLRI) to the BGP group.

```
user@leaf1# set protocols bgp group overlay type external
user@leaf1# set protocols bgp group overlay multihop
user@leaf1# set protocols bgp group overlay local-address 10.1.255.11
user@leaf1# set protocols bgp group overlay family evpn signaling
user@leaf1# set protocols bgp group overlay multipath multiple-as
user@leaf1# set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
user@leaf1# set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
```



NOTE: Some IP fabrics use an IBGP based EVPN-VXLAN overlay. For an example of an IP fabric that uses IBGP for the overlay, see ["Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric" on page 771](#). Note that choosing either EBGp or IBGP for the overlay does not impact the fabric architecture. Both CRB and ERB designs support either type of overlay.

2. Configure VXLAN encapsulation for the data packets exchanged between the EVPN neighbors. Specify that all VXLAN network identifiers (VNIs) are part of the virtual routing and forwarding (VRF) instance. Also, specify that the MAC address of the IRB interface and the MAC address of the corresponding default gateway are advertised without the extended community option default-gateway.

```
user@leaf1# set protocols evpn encapsulation vxlan
user@leaf1# set protocols evpn default-gateway no-gateway-community
user@leaf1# set protocols evpn multicast-mode ingress-replication
user@leaf1# set protocols evpn vni-options vni 101 vrf-target target:101:1
user@leaf1# set protocols evpn vni-options vni 102 vrf-target target:102:1
```

```
user@leaf1# set protocols evpn vni-options vni 103 vrf-target target:103:1
user@leaf1# set protocols evpn extended-vni-list all
```

3. Configure switch options to set a route distinguisher and VRF target for the VRF routing instance, and associate interface lo0 with the virtual tunnel endpoint (VTEP).

```
user@leaf1# set switch-options vtep-source-interface lo0.0
user@leaf1# set switch-options route-distinguisher 10.1.255.11:1
user@leaf1# set switch-options vrf-target target:1:1
user@leaf1# set switch-options vrf-target auto
```

Customer Profile Configuration

IN THIS SECTION

- [CLI Quick Configuration | 877](#)
- [Configuring a Customer Profile | 878](#)

CLI Quick Configuration

To quickly configure a basic customer profile for Servers A, Server B, Server C and Server D, copy the following commands and paste them into a text file. Remove any line breaks and change any details necessary to match your network configuration. Then copy and paste the commands into the CLI at the [edit] hierarchy level.

```
set chassis aggregated-devices ethernet device-count 1
set interfaces xe-0/0/2 ether-options 802.3ad ae0
set interfaces ae0 esi 00:01:01:01:01:01:01:01:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:01:01:01:01:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members v101
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.1/24 virtual-gateway-address
10.1.101.254
set interfaces irb unit 102 virtual-gateway-accept-data
```

```

set interfaces irb unit 102 family inet address 10.1.102.1/24 virtual-gateway-address
10.1.102.254
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 family inet address 10.1.103.1/24 virtual-gateway-address
10.1.103.254
set routing-instances vrf101 instance-type vrf
set routing-instances vrf101 interface irb.101
set routing-instances vrf101 route-distinguisher 10.1.255.11:101
set routing-instances vrf101 vrf-target target:1001:1
set routing-instances vrf102_103 instance-type vrf
set routing-instances vrf102_103 interface irb.102
set routing-instances vrf102_103 interface irb.103
set routing-instances vrf102_103 route-distinguisher 10.1.255.11:123
set routing-instances vrf102_103 vrf-target target:1023:1
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Configuring a Customer Profile

Step-by-Step Procedure

To configure a basic customer profile on Leaf1:

1. Enable Server A to be multihomed to Leaf1 and Leaf2 by configuring an aggregated Ethernet interface, specifying an ESI for the interface, and setting the mode so that the connections to both leaf devices are active.

```

user@leaf1# set chassis aggregated-devices ethernet device-count 1
user@leaf1# set interfaces xe-0/0/2 ether-options 802.3ad ae0
user@leaf1# set interfaces ae0 esi 00:01:01:01:01:01:01:01:01
user@leaf1# set interfaces ae0 esi all-active
user@leaf1# set interfaces ae0 aggregated-ether-options lacp active
user@leaf1# set interfaces ae0 aggregated-ether-options lacp system-id 00:01:01:01:01:01

```

```
user@leaf1# set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
user@leaf1# set interfaces ae0 unit 0 family ethernet-switching vlan members v101
```



NOTE: When configuring the ae0 interface on Leaf2, you must specify the same ESI (00:01:01:01:01:01:01:01) that is specified for the same interface on Leaf1.

2. Configure IRB interfaces and associated VGAs (default L3 virtual gateways), which enable the communication between servers in different VLANs. Add the optional configuration virtual-gateway-accept-data to allow the VGA to respond to ping packets.

```
user@leaf1# set interfaces irb unit 101 virtual-gateway-accept-data
user@leaf1# set interfaces irb unit 101 family inet address 10.1.101.1/24 virtual-gateway-
address 10.1.101.254
user@leaf1# set interfaces irb unit 102 virtual-gateway-accept-data
user@leaf1# set interfaces irb unit 102 family inet address 10.1.102.1/24 virtual-gateway-
address 10.1.102.254
user@leaf1# set interfaces irb unit 103 virtual-gateway-accept-data
user@leaf1# set interfaces irb unit 103 family inet address 10.1.103.1/24 virtual-gateway-
address 10.1.103.254
```



NOTE: When configuring a VGA for an IRB interface, keep in mind that the IRB IP address and VGA must be different.

3. Configure a VRF routing instance for VLAN v101 and another VRF routing instance for VLAN v102 and v103. In each routing instance, associate the IRB interfaces, a route-distinguisher, and a vrf-target.

```
user@leaf1# set routing-instances vrf101 instance-type vrf
user@leaf1# set routing-instances vrf101 interface irb.101
user@leaf1# set routing-instances vrf101 route-distinguisher 10.1.255.11:101
user@leaf1# set routing-instances vrf101 vrf-target target:1001:1
user@leaf1# set routing-instances vrf102_103 instance-type vrf
user@leaf1# set routing-instances vrf102_103 interface irb.102
user@leaf1# set routing-instances vrf102_103 interface irb.103
user@leaf1# set routing-instances vrf102_103 route-distinguisher 10.1.255.11:123
user@leaf1# set routing-instances vrf102_103 vrf-target target:1023:1
```


4. Configure VLANs v101, v102, and v103 and associate an IRB interface and VNI with each VLAN.

```

user@leaf1# set vlans v101 vlan-id 101
user@leaf1# set vlans v101 l3-interface irb.101
user@leaf1# set vlans v101 vxlan vni 101
user@leaf1# set vlans v102 vlan-id 102
user@leaf1# set vlans v102 l3-interface irb.102
user@leaf1# set vlans v102 vxlan vni 102
user@leaf1# set vlans v103 vlan-id 103
user@leaf1# set vlans v103 l3-interface irb.103
user@leaf1# set vlans v103 vxlan vni 103

```

Route Leaking Configuration

IN THIS SECTION

- [CLI Quick Configuration | 880](#)
- [Configuring Route Leaking | 881](#)

At this point based on the configuration, Server A and Server C should be able to reach each other. Server B and Server D should be able to reach each other. To enable all servers to be able to reach each other, we leak routes between the routing instances.

CLI Quick Configuration

To quickly configure route leaking, copy the following commands and paste them into a text file. Remove any line breaks and change any details necessary to match your network configuration. Then copy and paste the commands into the CLI at the [edit] hierarchy level.

```

set policy-options policy-statement vrf101_vrf_imp term 1 from community vrf102_103
set policy-options policy-statement vrf101_vrf_imp term 1 then accept
set policy-options policy-statement vrf102_103_vrf_imp term 1 from community vrf101
set policy-options policy-statement vrf102_103_vrf_imp term 1 then accept
set policy-options community vrf101 members target:1001:1
set policy-options community vrf102_103 members target:1023:1
set routing-instances vrf101 routing-options auto-export
set routing-instances vrf101 vrf-import vrf101_vrf_imp

```

```
set routing-instances vrf102_103 routing-options auto-export
set routing-instances vrf102_103 vrf-import vrf102_103_vrf_imp
```

Configuring Route Leaking

Step-by-Step Procedure

To configure route leaking on Leaf1:

1. Configure the communities that match the targets you configured for the routing instances. Configure policies that are applied under each routing instance that matches on the vrf-target of the other routing-instance.

```
user@leaf1# set policy-options policy-statement vrf101_vrf_imp term 1 from community
vrf102_103
user@leaf1# set policy-options policy-statement vrf101_vrf_imp term 1 then accept
user@leaf1# set policy-options policy-statement vrf102_103_vrf_imp term 1 from community
vrf101
user@leaf1# set policy-options policy-statement vrf102_103_vrf_imp term 1 then accept
user@leaf1# set policy-options community vrf101 members target:1001:1
user@leaf1# set policy-options community vrf102_103 members target:1023:1
```

2. In the VRF routing instances, apply the routing policies configured in the previous step. This establishes a common route target between the routing instances.

```
user@leaf1# set routing-instances vrf101 vrf-import vrf101_vrf_imp
user@leaf1# set routing-instances vrf102_103 vrf-import vrf102_103_vrf_imp
```

3. Configure the auto-export option that allows sharing routes between instances with common route targets.

```
user@leaf1# set routing-instances vrf101 routing-options auto-export
user@leaf1# set routing-instances vrf102_103 routing-options auto-export
```

Verification

IN THIS SECTION

- [Verifying BGP | 882](#)
- [Verifying the ESI | 883](#)
- [Verifying the EVPN Database | 884](#)
- [Verifying Connectivity | 885](#)

The section describes the following verifications for this example:

Verifying BGP

Purpose

Verify that the spine devices have established BGP session connectivity.

Action

Display the BGP summary:

```
user@leaf1> show bgp summary
Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 2 Peers: 4 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.evpn.0
              130         65         0         0         0         0
inet.0
              8          8          0          0         0         0
Peer          AS      InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.1.11.1      65001    31797    32161      0       0 1w3d 3:51:49 Establ
  inet.0: 4/4/4/0
10.1.21.1      65002    31794    32164      0       0 1w3d 3:51:48 Establ
  inet.0: 4/4/4/0
10.1.255.1     65001    32506    32631      0       0 1w3d 3:51:47 Establ
  bgp.evpn.0: 40/65/65/0
```

```

default-switch.evpn.0: 44/64/64/0
__default_evpn__.evpn.0: 0/1/1/0
10.1.255.2          65002      32530      32624      0          0 1w3d 3:51:43 Establ
bgp.evpn.0: 25/65/65/0
default-switch.evpn.0: 20/64/64/0
__default_evpn__.evpn.0: 1/1/1/0

```

Meaning

Both underlay and overlay BGP sessions are established with the spine devices.

Verifying the ESI

Purpose

Verify the status of the ESI.

Action

Display the status of the ESI:

```

user@leaf1> show evpn instance esi 00:01:01:01:01:01:01:01:01 extensive
Instance: default-switch
Route Distinguisher: 10.1.255.11:1
Encapsulation type: VXLAN
Duplicate MAC detection threshold: 5
Duplicate MAC detection window: 180
MAC database status
MAC advertisements:
MAC+IP advertisements:
Default gateway MAC advertisements:
Number of local interfaces: 2 (2 up)
Interface name  ESI
.local..6      00:00:00:00:00:00:00:00:00
ae0.0          00:01:01:01:01:01:01:01:01
Mode           Status  AC-Role
single-homed   Up      Root
all-active     Up      Root
Number of IRB interfaces: 3 (3 up)
Interface name  VLAN  VNI  Status  L3 context
irb.101         101   Up    vrf101
irb.102         102   Up    vrf102_103
irb.103         103   Up    vrf102_103
Number of protect interfaces: 0

```

```

Number of bridge domains: 3
  VLAN  Domain-ID Intfs/up  IRB-intf  Mode          MAC-sync IM-label  MAC-label  v4-SG-
sync IM-core-NH v6-SG-sync IM-core-NH Trans-ID
  101   101        1 1     irb.101   Extended    Enabled  101
Disabled
  102   102        0 0     irb.102   Extended    Enabled  102
Disabled
  103   103        0 0     irb.103   Extended    Enabled  103
Disabled
Number of neighbors: 3
  Address          MAC      MAC+IP      AD      IM      ES Leaf-label Remote-DCI-Peer
  10.1.255.12      8        8          5        3        0
  10.1.255.13      7        7          3        3        0
  10.1.255.14      7        7          3        3        0
Number of ethernet segments: 13
ESI: 00:01:01:01:01:01:01:01:01
Status: Resolved by IFL ae0.0
Local interface: ae0.0, Status: Up/Forwarding
Number of remote PEs connected: 1
  Remote-PE      MAC-label  Aliasing-label  Mode
  10.1.255.12    101        0               all-active
DF Election Algorithm: MOD based
Designated forwarder: 10.1.255.12
Backup forwarder: 10.1.255.11
Last designated forwarder update: Jun 02 12:46:14
Router-ID: 10.1.255.11
Source VTEP interface IP: 10.1.255.11
SMET Forwarding: Disabled

```

Meaning

The ESI is up and Leaf2 is the remote provider edge (PE) device and the designated forwarder.

Verifying the EVPN Database

Purpose

Verify the MAC addresses in the EVPN database.

Action

Verify the MAC addresses in the EVPN database for VLAN 101.

```
user@leaf1> show evpn database l2-domain-id 101
Instance: default-switch
```

VLAN	DomainId	MAC address	Active source	Timestamp	IP address
101		00:00:5e:00:01:01	05:00:00:fd:f3:00:00:00:65:00	May 23 11:58:57	10.1.101.254
101		02:05:86:11:af:00	10.1.255.14	May 26 12:49:49	10.1.101.4
101		02:05:86:54:14:00	10.1.255.13	May 23 12:04:41	10.1.101.3
101		02:05:86:8d:d9:00	irb.101	May 23 11:58:57	10.1.101.1
101		02:05:86:c2:39:00	10.1.255.12	May 23 11:58:58	10.1.101.2
101		2c:6b:f5:01:b1:c0	00:01:01:01:01:01:01:01:01:01	Jun 02 12:46:21	10.1.101.101
101		56:04:15:00:dc:95	10.1.255.13	May 23 12:07:33	10.1.101.103

Meaning

The MAC and IP addresses for Server A are shown with an active source of the ESI, and the MAC and IP addresses for Server C are shown with an active source from Leaf3. Also, the MAC for the VGA and each leaf IRB interface are shown.

Verifying Connectivity

Purpose

Verify ping works between servers.

Action

Ping from Server A to the other servers.

```
user@serverA> ping 10.1.102.101 count 2
PING 10.1.102.101 (10.1.102.101): 56 data bytes
64 bytes from 10.1.102.101: icmp_seq=0 ttl=63 time=117.425 ms
64 bytes from 10.1.102.101: icmp_seq=1 ttl=63 time=109.663 ms

--- 10.1.102.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 109.663/113.544/117.425/3.881 ms
```

```

user@serverA> ping 10.1.101.103 count 2
PING 10.1.101.103 (10.1.101.103): 56 data bytes
64 bytes from 10.1.101.103: icmp_seq=0 ttl=64 time=311.050 ms
64 bytes from 10.1.101.103: icmp_seq=1 ttl=64 time=201.300 ms

--- 10.1.101.103 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 201.300/256.175/311.050/54.875 ms

user@serverA> ping 10.1.103.101 count 2
PING 10.1.103.101 (10.1.103.101): 56 data bytes
64 bytes from 10.1.103.101: icmp_seq=0 ttl=63 time=311.321 ms
64 bytes from 10.1.103.101: icmp_seq=1 ttl=63 time=367.343 ms

--- 10.1.103.101 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 311.321/339.332/367.343/28.011 ms

```

Meaning

End-to-end connectivity is working.

Quick Configuration For All Devices

IN THIS SECTION

- [CLI Quick Configuration | 886](#)

CLI Quick Configuration

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

Leaf2

```

set chassis aggregated-devices ethernet device-count 1
set interfaces xe-0/0/0 unit 0 family inet address 10.1.12.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.22.2/30

```

```

set interfaces xe-0/0/2 ether-options 802.3ad ae0
set interfaces xe-0/0/3 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/3 unit 0 family ethernet-switching vlan members v102
set interfaces ae0 esi 00:01:01:01:01:01:01:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:01:01:01:01:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members v101
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.2/24 virtual-gateway-address
10.1.101.254
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 family inet address 10.1.102.2/24 virtual-gateway-address
10.1.102.254
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 family inet address 10.1.103.2/24 virtual-gateway-address
10.1.103.254
set interfaces lo0 unit 0 family inet address 10.1.255.12/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.12/32 exact
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.11/32 exact
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement vrf101_vrf_imp term 1 from community vrf102_103
set policy-options policy-statement vrf101_vrf_imp term 1 then accept
set policy-options policy-statement vrf102_103_vrf_imp term 1 from community vrf101
set policy-options policy-statement vrf102_103_vrf_imp term 1 then accept
set policy-options community vrf101 members target:1001:1
set policy-options community vrf102_103 members target:1023:1
set routing-instances vrf101 instance-type vrf
set routing-instances vrf101 routing-options auto-export
set routing-instances vrf101 interface irb.101
set routing-instances vrf101 route-distinguisher 10.1.255.12:101
set routing-instances vrf101 vrf-import vrf101_vrf_imp
set routing-instances vrf101 vrf-target target:1001:1
set routing-instances vrf102_103 instance-type vrf
set routing-instances vrf102_103 routing-options auto-export
set routing-instances vrf102_103 interface irb.102
set routing-instances vrf102_103 interface irb.103
set routing-instances vrf102_103 route-distinguisher 10.1.255.12:123
set routing-instances vrf102_103 vrf-import vrf102_103_vrf_imp
set routing-instances vrf102_103 vrf-target target:1023:1

```



```

set routing-options router-id 10.1.255.12
set routing-options autonomous-system 65012
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.12.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.22.1 peer-as 65002
set protocols bgp group overlay type external
set protocols bgp group overlay multihop
set protocols bgp group overlay local-address 10.1.255.12
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1
set protocols evpn vni-options vni 102 vrf-target target:102:1
set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.12:1
set switch-options vrf-target target:1:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Leaf3

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.13.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.23.2/30
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members v101

```

```

set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.3/24 virtual-gateway-address
10.1.101.254
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 family inet address 10.1.102.3/24 virtual-gateway-address
10.1.102.254
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 family inet address 10.1.103.3/24 virtual-gateway-address
10.1.103.254
set interfaces lo0 unit 0 family inet address 10.1.255.13/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.13/32 exact
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement vrf101_vrf_imp term 1 from community vrf102_103
set policy-options policy-statement vrf101_vrf_imp term 1 then accept
set policy-options policy-statement vrf102_103_vrf_imp term 1 from community vrf101
set policy-options policy-statement vrf102_103_vrf_imp term 1 then accept
set policy-options community vrf101 members target:1001:1
set policy-options community vrf102_103 members target:1023:1
set routing-instances vrf101 instance-type vrf
set routing-instances vrf101 routing-options auto-export
set routing-instances vrf101 interface irb.101
set routing-instances vrf101 route-distinguisher 10.1.255.13:101
set routing-instances vrf101 vrf-import vrf101_vrf_imp
set routing-instances vrf101 vrf-target target:1001:1
set routing-instances vrf102_103 instance-type vrf
set routing-instances vrf102_103 routing-options auto-export
set routing-instances vrf102_103 interface irb.102
set routing-instances vrf102_103 interface irb.103
set routing-instances vrf102_103 route-distinguisher 10.1.255.13:123
set routing-instances vrf102_103 vrf-import vrf102_103_vrf_imp
set routing-instances vrf102_103 vrf-target target:1023:1
set routing-options router-id 10.1.255.13
set routing-options autonomous-system 65013
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.13.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.23.1 peer-as 65002
set protocols bgp group overlay type external
set protocols bgp group overlay multihop

```

```

set protocols bgp group overlay local-address 10.1.255.13
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1
set protocols evpn vni-options vni 102 vrf-target target:102:1
set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.13:1
set switch-options vrf-target target:1:1
set switch-options vrf-target auto
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Leaf4

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.14.2/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.24.2/30
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members v103
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 family inet address 10.1.101.4/24 virtual-gateway-address
10.1.101.254
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 family inet address 10.1.102.4/24 virtual-gateway-address
10.1.102.254
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 family inet address 10.1.103.4/24 virtual-gateway-address
10.1.103.254
set interfaces lo0 unit 0 family inet address 10.1.255.14/32

```

```

set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.14/32 exact
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement vrf101_vrf_imp term 1 from community vrf102_103
set policy-options policy-statement vrf101_vrf_imp term 1 then accept
set policy-options policy-statement vrf102_103_vrf_imp term 1 from community vrf101
set policy-options policy-statement vrf102_103_vrf_imp term 1 then accept
set policy-options community vrf101 members target:1001:1
set policy-options community vrf102_103 members target:1023:1
set routing-instances vrf101 instance-type vrf
set routing-instances vrf101 routing-options auto-export
set routing-instances vrf101 interface irb.101
set routing-instances vrf101 route-distinguisher 10.1.255.14:101
set routing-instances vrf101 vrf-import vrf101_vrf_imp
set routing-instances vrf101 vrf-target target:1001:1
set routing-instances vrf102_103 instance-type vrf
set routing-instances vrf102_103 routing-options auto-export
set routing-instances vrf102_103 interface irb.102
set routing-instances vrf102_103 interface irb.103
set routing-instances vrf102_103 route-distinguisher 10.1.255.14:123
set routing-instances vrf102_103 vrf-import vrf102_103_vrf_imp
set routing-instances vrf102_103 vrf-target target:1023:1
set routing-options router-id 10.1.255.14
set routing-options autonomous-system 65014
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.14.1 peer-as 65001
set protocols bgp group underlay neighbor 10.1.24.1 peer-as 65002
set protocols bgp group overlay type external
set protocols bgp group overlay multihop
set protocols bgp group overlay local-address 10.1.255.14
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn multicast-mode ingress-replication
set protocols evpn vni-options vni 101 vrf-target target:101:1
set protocols evpn vni-options vni 102 vrf-target target:102:1

```

```

set protocols evpn vni-options vni 103 vrf-target target:103:1
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.1.255.14:1
set switch-options vrf-target target:1:1
set vlans v101 vlan-id 101
set vlans v101 l3-interface irb.101
set vlans v101 vxlan vni 101
set vlans v102 vlan-id 102
set vlans v102 l3-interface irb.102
set vlans v102 vxlan vni 102
set vlans v103 vlan-id 103
set vlans v103 l3-interface irb.103
set vlans v103 vxlan vni 103

```

Spine 1

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.11.1/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.12.1/30
set interfaces xe-0/0/2 unit 0 family inet address 10.1.13.1/30
set interfaces xe-0/0/3 unit 0 family inet address 10.1.14.1/30
set interfaces lo0 unit 0 family inet address 10.1.255.1/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.1/32 exact
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.1
set routing-options autonomous-system 65001
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.11.2 peer-as 65011
set protocols bgp group underlay neighbor 10.1.12.2 peer-as 65012
set protocols bgp group underlay neighbor 10.1.13.2 peer-as 65013
set protocols bgp group underlay neighbor 10.1.14.2 peer-as 65014
set protocols bgp group overlay type external
set protocols bgp group overlay multihop no-nexthop-change
set protocols bgp group overlay local-address 10.1.255.1
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.11 peer-as 65011

```

```

set protocols bgp group overlay neighbor 10.1.255.12 peer-as 65012
set protocols bgp group overlay neighbor 10.1.255.13 peer-as 65013
set protocols bgp group overlay neighbor 10.1.255.14 peer-as 65014
set protocols bgp group overlay neighbor 10.1.255.2 peer-as 65002

```

Spine 2

```

set interfaces xe-0/0/0 unit 0 family inet address 10.1.21.1/30
set interfaces xe-0/0/1 unit 0 family inet address 10.1.22.1/30
set interfaces xe-0/0/2 unit 0 family inet address 10.1.23.1/30
set interfaces xe-0/0/3 unit 0 family inet address 10.1.24.1/30
set interfaces lo0 unit 0 family inet address 10.1.255.2/32
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.1.255.2/32 exact
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 10.1.255.2
set routing-options autonomous-system 65002
set routing-options forwarding-table export load-balancing-policy
set protocols bgp group underlay type external
set protocols bgp group underlay export send-direct
set protocols bgp group underlay multipath multiple-as
set protocols bgp group underlay neighbor 10.1.21.2 peer-as 65011
set protocols bgp group underlay neighbor 10.1.22.2 peer-as 65012
set protocols bgp group underlay neighbor 10.1.23.2 peer-as 65013
set protocols bgp group underlay neighbor 10.1.24.2 peer-as 65014
set protocols bgp group overlay type external
set protocols bgp group overlay multihop no-nexthop-change
set protocols bgp group overlay local-address 10.1.255.2
set protocols bgp group overlay family evpn signaling
set protocols bgp group overlay multipath multiple-as
set protocols bgp group overlay neighbor 10.1.255.11 peer-as 65011
set protocols bgp group overlay neighbor 10.1.255.12 peer-as 65012
set protocols bgp group overlay neighbor 10.1.255.13 peer-as 65013
set protocols bgp group overlay neighbor 10.1.255.14 peer-as 65014
set protocols bgp group overlay neighbor 10.1.255.1 peer-as 65001

```

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
17.3R1	Starting with Junos OS Release 17.3R1, the QFX5110 switch can function as a leaf device, which acts as an L2 and an L3 VXLAN gateway in an EVPN-VXLAN ERB overlay.

RELATED DOCUMENTATION

Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric 771
Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric Using MX Routers as Spines 808
Understanding EVPN with VXLAN Data Plane Encapsulation 588
EVPN Primer

EVPN-VXLAN Pure Type 5 Host-Route Auto-Generated Community

IN THIS SECTION

- [Benefits | 895](#)
- [Behavior and Limitations | 895](#)
- [Configure Type 5 Host-Route Auto-Generated Community | 896](#)

Border leaf devices in edge-routed bridging (ERB) EVPN topologies with Type 5 connectivity to external EVPN networks need to advertise aggregate routes to the external networks instead of individual Type 5 host routes. Leaf devices configured with the auto-generated community add a community to MAC-IP ARP/NDP based Type 5 routes and Type 2 MAC-IP routes. When the remote PE generates a Type 5 route from the received Type 2 route, it inherits the community. Border leaf devices can use this community to identify these routes and create an aggregate route to advertise to external EVPN networks.

Benefits

- Provides a mechanism to automatically add a community to locally learned MAC-IP ARP/NDP based pure Type 5 host routes and Type 2 MAC-IP routes.

Behavior and Limitations

Note the following runtime behaviors and limitations with this feature:

- This feature applies to the EVPN locally learned host routes within a Layer 2 instance. The policy only needs a single action to add a community. It is not required to match anything in the route. But you can add a matching qualifier [route-filter, route-filter-list, prefix-filter, prefix-filter-list] to the policy to limit which routes are matched, if necessary.
- The IP host route inherits the community from the remotely learned Type 2 MAC-IP route when added to the `L3VRF.inet` or `inet6.0` table. The Type 5 IP prefix routes generated from those Type 2 routes also inherit that community. However, you can prevent that inheritance by configuring the export-action skip option under [routing-instances *name* protocols evpn ip-prefix-routes route-attributes community].
- The export policy does not modify any other route parameters. It only adds a community to the locally learned EVPN host routes.
- The export policy does not add a community to Type 2 MAC-IP routes generated for IRB physical or virtual gateway IP Addresses.
- Applying this policy to an EVPN Layer 2 instance does not change the existing behavior for advertising or receiving Type 5 routes.
- An ip-prefix-route export policy configured to delete or override communities on IP Host Routes in the VRF instance while generating EVPN Type 5 routes will also drop the community that is added to the IP Host Route by the export policy under EVPN Layer 2 instance.
- The export policy does not reject learning of any local host routes in EVPN.
- The export policy does not reject advertising Type 2 MAC-IP routes. Existing VRF export policies under the EVPN instance or under [protocols bgp] continue to be used to reject advertising Type 2 MAC-IP routes.
- The export policy does not reject a locally learned EVPN route from being added to `L3VRF.inet` or `inet6.0` table and therefore does not prevent generating a T-5 route for locally learned host routes. The existing export policies under [ip-prefix-route export] or [protocols bgp] are used to reject advertising Type 5 routes.

Configure Type 5 Host-Route Auto-Generated Community

You enable the Type 5 Host-Route Auto-Generated Community by configuring the *mark-local-ip-host-routes* statement under the EVPN Layer 2 Instance. The statement invokes a policy that adds a community to the EVPN locally learned host routes for that instance.

1. Enable the auto-generated community in the routing instance.

```
set routing-instances instance-name protocols evpn mark-local-ip-host-routes export policy-name;
```

2. Configure the community to identify the routes.

```
set policy-options community community-name members community;
```

3. Configure the policy to add the community to the routes.

- a. Add the community without any filters.

```
set policy-options policy-statement policy-name term 1 then community add community-name;  
set policy-options policy-statement policy-name term 1 then accept;
```

- b. Add the community using a matching qualifier [route-filter, route-filter-list, prefix-filter, prefix-filter-list].

```
set policy-options policy-statement policy-name term 1 from route-filter address orlonger;  
set policy-options policy-statement policy-name term 1 then community add community-name;  
set policy-options policy-statement policy-name term 1 then accept;
```

RELATED DOCUMENTATION

[Understanding EVPN Pure Type 5 Routes | 50](#)

[ip-prefix-routes](#)

[EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN | 65](#)

IPv6 Underlay for VXLAN Overlays

IN THIS CHAPTER

- [EVPN-VXLAN with an IPv6 Underlay | 897](#)
- [Example: Configure an IPv6 Underlay for Layer 2 VXLAN Gateway Leaf Devices | 905](#)

EVPN-VXLAN with an IPv6 Underlay

SUMMARY

This topic describes how to set up an IPv6 underlay for the VXLAN overlay tunneling in an EVPN-VXLAN fabric.

IN THIS SECTION

- [IPv6 Underlay Support in EVPN-VXLAN Fabrics | 897](#)
- [Configure an IPv6 Underlay with EVPN-VXLAN | 903](#)

IPv6 Underlay Support in EVPN-VXLAN Fabrics

IN THIS SECTION

- [Benefits of Using an IPv6 Underlay with a VXLAN Overlay | 898](#)
- [Platform Support | 898](#)
- [Overview | 898](#)
- [Underlay Routing Protocols with an IPv6 Underlay | 900](#)
- [EVPN-VXLAN Features Supported with an IPv6 Underlay | 900](#)
- [Limitations in IPv6 Underlay Support | 903](#)

Ethernet VPNs (EVPNs) connect devices with Layer 2 (L2) virtual bridges. Virtual Extensible LANs (VXLANs) establish overlay tunnels that stretch the L2 connections over a Layer 3 (L3) network. In EVPN-VXLAN network configurations, a leaf or spine device can function as a VXLAN gateway at L2, L3, or both layers. The underlay network for the VXLAN overlay can be an IPv4 or an IPv6 network. This topic describes using an IPv6 underlay instead of an IPv4 underlay.

Benefits of Using an IPv6 Underlay with a VXLAN Overlay

- With an IPv6 underlay VXLAN tunnel configuration, you can take advantage of the expanded addressing capabilities and efficient packet processing that the IPv6 protocol offers.

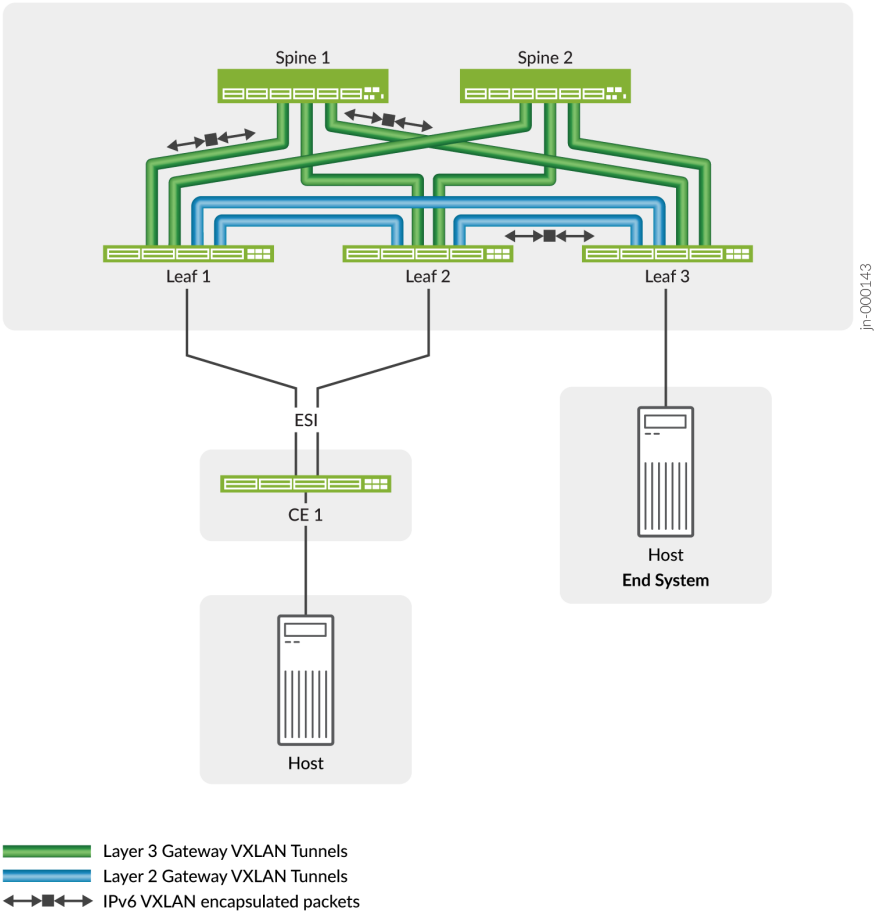
Platform Support

For information on supported platforms and Junos releases, see [Feature Explorer](#).

Overview

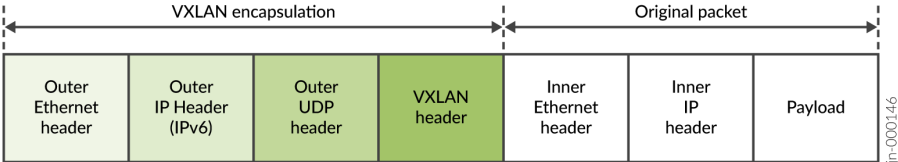
In EVPN-VXLAN installations, you configure a VXLAN overlay on L2 or L3 VXLAN gateway devices called virtual tunnel endpoints (VTEPs). The VXLAN overlay extends virtual tunnels between VTEPs over the underlying IP fabric. On supporting platforms, you can configure the IP underlay with IPv6 addressing to support the VXLAN overlay tunnels. For example:

Figure 80: EVPN-VXLAN Fabric with an IPv6 Underlay



When you use an IPv6 underlay, the VTEPs encapsulate VXLAN packets with an IPv6 outer header and tunnel the packets through an IPv6 underlay network.

Figure 81: IPv6 Underlay VXLAN Packet Encapsulation



IPv6 underlay configurations are similar to IPv4 underlay configurations, except you set the VTEP source addresses as IPv6 addresses. You also assign IPv6 addresses in the underlay and establish reachability using the IPv6 protocol.

Underlay Routing Protocols with an IPv6 Underlay

We've qualified an IPv6 underlay with the following routing protocols in the underlay configuration:

- BGP—Internal BGP (IBGP) and external BGP (eBGP)
- OSPFv3—Open Shortest Path First (OSPF) routing protocol for IPv6

EVPN-VXLAN Features Supported with an IPv6 Underlay

We support the following EVPN-VXLAN features with an IPv6 underlay:



NOTE: We support the items below on all platforms that support EVPN-VXLAN with an IPv6 underlay unless the item lists particular platforms. Use [Feature Explorer](#) to confirm platform and release support for specific features.

- EVPN Type 1, Type 2, Type 3, and Type 4 routes.

(QFX Series, ACX Series, PTX Series, and MX Series devices) EVPN Type 5 routes.

See ["EVPN Type 5 Route with VXLAN Encapsulation for EVPN-VXLAN" on page 47](#) for more about these EVPN route types.

- (QFX5130-32CD, QFX5700, ACX Series, and PTX Series devices) EVPN Type 2 and Type 5 route coexistence.

See ["EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN" on page 65](#).

- (QFX Series and MX Series devices) Shared VTEP tunnels.

On QFX series switches, we support an EVPN-VXLAN IPv6 underlay only with MAC-VRF EVPN routing instances. The MAC-VRF implementation relies on the shared VTEP tunnels feature to avoid VTEP scaling issues on some devices. On the devices that require shared tunnels but don't have the feature enabled by default, when you configure an IPv6 underlay, you must enable the `shared-tunnels` option at the `[edit forwarding-options evpn-vxlan]` hierarchy level. See ["MAC-VRF Routing Instance Type Overview" on page 38](#) for more about MAC-VRF instances.



NOTE: After you configure the `shared-tunnels` option, you must reboot the device for the setting to take effect.

- (All devices) VLAN-aware bundle, VLAN-based, and VLAN bundle Ethernet service types.

(MX Series routers, additionally) Port-based service, a VLAN bundle service where all VLANs for a port are part of the same VLAN bundle.

See ["Understanding VLAN-Aware Bundle and VLAN-Based Service for EVPN" on page 710](#) and ["MAC-VRF Routing Instance Type Overview" on page 38](#) for more about MAC-VRF instances and these service types.

- All-active multihoming.

See ["EVPN Multihoming Overview" on page 162](#).

- Underlay and overlay load balancing over multiple IPv6 VXLAN tunnels with multihoming.

See ["Load Balancing VXLAN Traffic" on page 614](#) and ["Dynamic Load Balancing in an EVPN-VXLAN Network" on page 723](#).

- Loop prevention on link recovery.

When multihoming interfaces on an Ethernet segment ID (ESI) flap, the device drops broadcast, unknown unicast, and multicast (BUM) traffic that comes into the ESI interface for a configured hold timer interval.

See ["Understanding EVPN with VXLAN Data Plane Encapsulation" on page 588](#).

- EVPN core isolation.

See ["Understanding When to Disable EVPN-VXLAN Core Isolation" on page 514](#).

- EVPN-VXLAN L2 and L3 gateway functions in ERB and CRB overlays with IPv4 or IPv6 traffic.

- Bridged overlays.

See [Bridged Overlay Design and Implementation](#).

- (EX Series and QFX Series switches) IPv4 and IPv6 multicast data traffic with the following multicast modes in an EVPN-VXLAN network:

- Centrally-routed multicast with local-remote forwarding mode—see ["Multicast Support in EVPN-VXLAN Overlay Networks" on page 925](#).
- Enhanced optimized intersubnet multicast (OISM) mode—see ["Overview of Enhanced OISM" on page 1057](#).

- Flood policers in EVPN-VXLAN bridge domains (VLANs) with IPv6 VXLAN tunnels using filters.

- (PTX Series routers) Overlay ping and overlay traceroute, and CE-IP ping and CE-IP traceroute.

See [ping overlay](#), [traceroute overlay](#), and [Pinging Customer Edge Device IP Address](#).

- (EX Series, QFX Series, and PTX Series devices) Quality of service (QoS) and Class of Service (CoS) classification, including:
 - Classification using Differentiated Services Code Point (DSCP) or IEEE 802.1p code points.



NOTE: (PTX Series routers) No support for DSCP copy or IEEE 802.1p rewrite operations.

- (PTX Series routers) Explicit Congestion Notification (ECN) field copy:
 - During VXLAN encapsulation, the device copies the ECN field from the inner header to the outer header.
 - During VXLAN decapsulation, the device interprets the ECN field from the outer header and propagates it to the inner header if the inner header ECN field value is non-zero (the inner header is ECN-capable).

See *CoS Support on EVPN VXLANs* for details.

- L3 protocols over IRB interfaces with BFD—Static routing, BGP, IS-IS, OSPF and OSPFv3.

See ["Supported Protocols on an IRB Interface in EVPN-VXLAN "](#) on page 760.

- Data center interconnect (DCI) over-the-top (OTT) full mesh.

See [Over-the-Top Data Center Interconnect in an EVPN Network](#) for details on the OTT DCI method.

- DCI seamless stitching, and optimized multicast traffic over DCI seamless stitching using enhanced OISM.

See ["EVPN-VXLAN DCI Multicast with Enhanced OISM"](#) on page 1188 for details on how these features work together in EVPN fabrics with IPv4 or IPv6 underlay peering.

- EVPN proxy ARP and ARP suppression, as well as EVPN proxy NDP and NDP suppression.

See ["EVPN Proxy ARP and ARP Suppression, and Proxy NDP and NDP Suppression"](#) on page 549 for more information on these features.

- (EX Series and QFX Series switches) Remote port mirroring and analyzers.

See ["MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment"](#) on page 687.

- (EX Series, QFX Series, and PTX Series devices) DHCP relay with DHCPv4 and DHCPv6.

See ["DHCP Relay Agent over EVPN-VXLAN "](#) on page 561.

Limitations in IPv6 Underlay Support

Note the following limitations in IPv6 underlay support:

- You can't mix IPv4 and IPv6 underlay configurations for the VXLAN overlays across the EVPN instances in the same fabric.
- We don't support the Open vSwitch database (OVSDb) management protocol for IPv6 underlays.
- (EX9200 switches) We don't support EVPN Type 5 routes with an IPv6 underlay.
- (QFX10002-60C switches) You can only use enterprise style interface configuration; we don't support service provider style interface configuration and Q-in-Q tunneling with IPv4 or IPv6 underlays on these switches.
- You must use MAC-VRF routing instances with EVPN protocol and VXLAN encapsulation. We don't support IPv6 underlays with other instance types such as evpn, evpn-vpws, virtual-switch or the default switching instance.

Configure an IPv6 Underlay with EVPN-VXLAN

This section describes the key steps to configure the IP underlay for the VXLAN tunnels in an EVPN-VXLAN fabric to use the IPv6 protocol (instead of an IPv4 underlay). You can use an IPv6 underlay in many different EVPN-VXLAN configurations and use cases. See these detailed configuration examples we provide for a few common use cases:

- ["Example: Configure an IPv6 Underlay for Layer 2 VXLAN Gateway Leaf Devices" on page 905](#)—OSPFv3 in the underlay and IBGP for the overlay connectivity
- [IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGp](#)—EBGP with LACP for load-balancing in the underlay and EBGp for the overlay connectivity
- [BGP Unnumbered IPv6 Underlay in an EVPN-VXLAN Data Center](#)—BGP autodiscovery with ECMP for load-balancing in the underlay that works with a variety of protocols for the overlay connectivity

Keep the following configuration options and requirements in mind when you plan to configure EVPN-VXLAN with an IPv6 underlay:

- You can configure your EVPN-VXLAN fabric with any of the underlay routing protocols that support IPv6 underlays. See ["Underlay Routing Protocols with an IPv6 Underlay" on page 900](#).
- We support IPv6 underlays only with MAC-VRF routing instances in EVPN-VXLAN fabrics. You must configure your EVPN instances with VXLAN encapsulation in all MAC-VRF routing instances. The routing instances mentioned in the steps here are always EVPN-VXLAN MAC-VRF instances. See ["MAC-VRF Routing Instance Type Overview" on page 38](#) for more about MAC-VRF instances.

- If the network uses an IPv4 underlay and you're switching the configuration to an IPv6 underlay, you must:
 - Remove any existing VXLAN IPv4 underlay configuration items.
 - (ACX7100-32C, AX7100-48L, and ACX7024 devices only) To enable an IPv6 underlay, you must configure the `vlan-extended` system profile option on the device, as follows:

```
set system packet-forwarding-options system-profile vlan-extended
```

When you change the system profile, the Packet Forwarding Engine reboots. After the Packet Forwarding Engine comes back up, you can continue with the IPv6 VXLAN underlay configuration.



NOTE: If you switch a configuration from an IPv6 underlay to an IPv4 underlay, be sure to delete the `vlan-extended` option configuration item to restore the device to the default system profile as part of setting up the IPv4 underlay.

- Configure the IPv6 underlay statements, and commit that configuration.
- Reboot the device.
- (PTX Series routers only) In any EVPN-VXLAN network, whether the configuration uses an IPv4 underlay or an IPv6 underlay, you must enable the *tunnel-termination* option at the [edit forwarding-options] global hierarchy level, or at the [edit interfaces *name* unit *unit-number* family inet6] physical interface hierarchy level. A more specific interface level setting supersedes a global setting.

To enable an IPv6 underlay for VXLAN tunneling, include these items in your EVPN-VXLAN fabric configuration:

1. Assign an IPv6 address to the loopback interface (lo0) on the devices that serve as L2 or L3 VXLAN gateway VTEPs.
2. Configure the underlay and EVPN instance device connectivity for the overlay using any of the supported routing protocols with IPv6 addressing. (See ["Underlay Routing Protocols with an IPv6 Underlay" on page 900.](#))
3. In the EVPN routing instance, configure the underlay VTEP source interface as the device's loopback IPv6 address. For example:

```
set routing-instances instance-name vtep-source-interface lo0.0 inet6
```

4. The underlay uses the IPv6 address family. IPv6 protocols also need a 32-bit router ID to function properly. The router ID must be a 4 octet unsigned nonzero integer that is unique in the routing

domain. Configure the router ID using dotted quad notation. The IPv6-based overlay uses the IPv6 loopback address for the VTEP local address.

```
set routing-options router-id router-id
```

5. (QFX Series Broadcom-based switches in Junos OS Release 21.2R2 only) Enable the Broadcom VXLAN flexible flow feature. You don't need this step in later releases, which have this option enabled in the default configuration on any affected Junos devices. After you set this option, you must reboot the device for the change to take effect.

```
set forwarding-options vxlan-flexflow
```

SEE ALSO

[Understanding EVPN with VXLAN Data Plane Encapsulation | 588](#)

[Understanding VXLANs | 606](#)

[Example: Configure an IPv6 Underlay for Layer 2 VXLAN Gateway Leaf Devices | 905](#)

Example: Configure an IPv6 Underlay for Layer 2 VXLAN Gateway Leaf Devices

IN THIS SECTION

- [Overview | 906](#)
- [Requirements | 909](#)
- [Topology | 909](#)
- [Configure Leaf 1 | 910](#)
- [Configure Leaf 3 | 916](#)
- [Verify the IPv6 Underlay on Leaf 3 | 920](#)

Overview

IN THIS SECTION

- [Overview of Configuration Differences with an IPv6 Underlay Compared to an IPv4 Underlay | 907](#)
- [References to Other Example Configurations for EVPN-VXLAN with an IPv6 Underlay | 908](#)
- [Considerations Before You Configure EVPN-VXLAN with an IPv6 Underlay | 908](#)

Ethernet VPNs (EVPNs) enable you to connect customer sites using Layer 2 virtual bridges. Virtual Extensible LANs (VXLANs) establish overlay tunnels that stretch the Layer 2 connection over an intervening Layer 3 network. Like VLANs, VXLANs help provide network segmentation, but without the scaling limitation of traditional VLANs. EVPN with VXLAN encapsulation enables Layer 2 connectivity at scale.

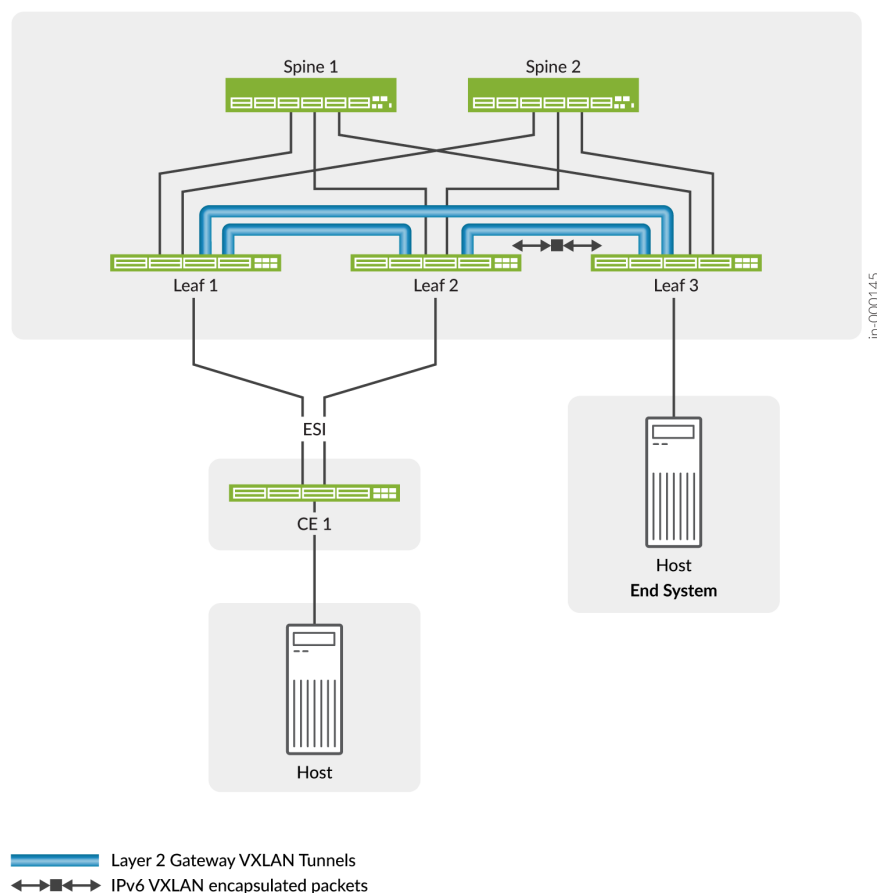
The physical underlay network in EVPN-VXLAN installations is often a two-layer IP fabric that includes spine and leaf devices. In this design, the spine devices provide connectivity between the leaf devices and the leaf devices provide connectivity to attached hosts. In different overlay network configurations, the leaf or spine devices might function as either or both of the following:

- Layer 2 gateways that handle traffic within a VXLAN.
- Layer 3 gateways that handle traffic between VXLANs using integrated routing and bridging (IRB) interfaces.

On supported platforms, in either case, the underlay network for the VXLAN overlay can use the IPv6 protocol to take advantage of the extended addressing and other capabilities of IPv6.

This example shows a use case to configure an IPv6 underlay for the Layer 2 VXLAN gateway leaf devices in a simple EVPN-VXLAN fabric. In this use case, the EVPN-VXLAN fabric supports a bridged overlay with VXLAN tunnels between the leaf devices. The leaf devices connect to end systems that might be single homed or include EVPN multihoming for redundancy. The following figure shows a high-level view of the topology in this example:

Figure 82: EVPN-VXLAN Fabric with an IPv6 Underlay for Layer 2 VXLAN Gateway Devices



Overview of Configuration Differences with an IPv6 Underlay Compared to an IPv4 Underlay

The following list describes the main differences in how you set up an IPv6 underlay compared to setting up an IPv4 underlay:

- You assign an IPv6 address to the loopback interface on the devices that serve as the Layer 2 or Layer 3 VXLAN gateway VTEPs.
- We support an IPv6 VXLAN underlay only with MAC-VRF routing instances. (See ["MAC-VRF Routing Instance Type Overview" on page 38](#) for more information about using MAC-VRF routing instances.) As a result, you configure the EVPN instance as a MAC-VRF instance.
- You set the VTEP source interface as an IPv6 address. You also must assign IPv6 addresses to the EVPN core-facing interfaces for IP reachability with IPv6.
- IPv6 protocols need a 32-bit router ID to function properly. The router ID you assign must be:

- A 32-bit value as a 4-octet, unsigned, non-zero integer.
- Unique within the routing domain.

References to Other Example Configurations for EVPN-VXLAN with an IPv6 Underlay

The use case in this configuration example uses OSPFv3 for IPv6 connectivity and internal BGP (IBGP) with IPv6 neighbor addressing in one autonomous system (AS). We also provide configuration examples for other tested IPv6 underlay use cases in the [Data Center EVPN-VXLAN Fabric Architecture Guide](#), as follows:

- [IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGp](#)
- [BGP Unnumbered IPv6 Underlay in an EVPN-VXLAN Data Center](#)

Considerations Before You Configure EVPN-VXLAN with an IPv6 Underlay

Before you start to configure an EVPN-VXLAN fabric with an IPv6 underlay, note the following:

- You can't mix IPv4 and IPv6 underlays in the same fabric, so you must configure an IPv6 underlay across all EVPN instances in the fabric.
- If the network uses an IPv4 underlay and you're switching the configuration to an IPv6 underlay, you must do the following:
 - Remove any existing VXLAN IPv4 underlay configuration items.
 - (ACX7100-32C, AX7100-48L, and ACX7024 devices only) To enable the IPv6 underlay, you must set the system profile to the `vxlan-extended` option, as follows:

```
set system packet-forwarding-options system-profile vxlan-extended
```

When you change the system profile, the Packet Forwarding Engine reboots. After the Packet Forwarding Engine comes back up, you can continue with the IPv6 VXLAN underlay configuration.



NOTE: If you switch a configuration from an IPv6 underlay to an IPv4 underlay, be sure to delete the `vxlan-extended` option configuration item to restore the device to the default system profile as part of setting up the IPv4 underlay.

- Configure the IPv6 underlay statements, and commit that configuration.
- Reboot the device.

Requirements

This example consists of a full mesh two-layer spine-and-leaf EVPN-VXLAN fabric with two spine devices and three leaf devices. You can configure the IPv6 underlay in this example using:

- QFX Series switches that support this feature.
- Junos OS Release 21.4R1 or later for QFX5120 switches and switches in the QFX10000 line, Junos OS Evolved 22.3R1 or later for QFX5130-32CD and QFX5700 switches, and Junos OS Evolved Release 23.4R1 on ACX7100-32C and ACX7100-48L devices.



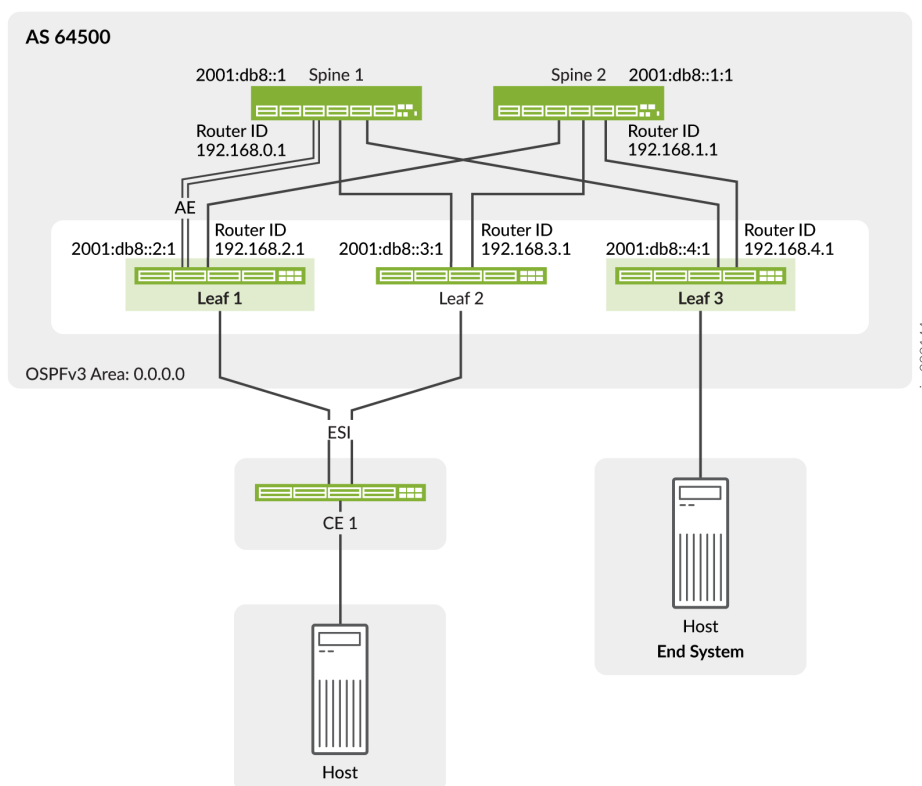
NOTE: We also support this feature in Junos OS Release 21.2R2 on QFX5120 switches and switches in the QFX10000 line.

The leaf devices can host multihomed or single homed end devices on the access side. This example illustrates configuring an Ethernet segment for EVPN multihoming on one leaf and a single homed end system interface on another leaf. However, the elements you configure for the IPv6 underlay are independent of the access-side configuration.

Topology

This example shows how to configure an IPv6 underlay on Leaf 1 and Leaf 3 for VXLAN overlay tunnels like those in [Figure 82 on page 907](#). The configuration uses OSPFv3 for IPv6 connectivity and IBGP with IPv6 neighbor addressing in a single AS in the following topology:

Figure 83: Example Topology



Leaf 1 serves a customer edge switch that is multihomed to Leaf 1 and Leaf 2, so you would use a similar configuration on Leaf 2 to reach devices on that Ethernet segment.

In the example topology, Leaf 1 includes an aggregated Ethernet interface bundle for the connection to Spine 1. You configure the remaining spine and leaf connections on Leaf 1 and Leaf 3 as single interfaces. Leaf 3 includes an access-side interface configuration to a single-homed end system.

This example includes show commands you can run to verify IPv6 underlay operation. For simplicity, we show these verification commands and output only for Leaf 3. You see similar results from the same commands on the other leaf devices.

Configure Leaf 1

IN THIS SECTION

- [CLI Quick Configuration on Leaf 1 | 911](#)
- [Step-by-Step Procedure on Leaf 1 | 912](#)

CLI Quick Configuration on Leaf 1

To quickly configure Leaf 1 with an IPv6 underlay according to [Figure 83 on page 910](#), 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-0/0/18:0 description "LINK-0 to SPINE-1"
set interfaces xe-0/0/18:0 ether-options 802.3ad ae0
set interfaces xe-0/0/18:1 description "LINK-1 to SPINE-1"
set interfaces xe-0/0/18:1 ether-options 802.3ad ae0
set interfaces ae0 aggregated-ether-options link-speed mixed
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 unit 0 family inet6 address 2001:db8::02:1:2/112
set interfaces xe-0/0/34:0 description "LINK to SPINE-2"
set interfaces xe-0/0/34:0 unit 0 family inet6 address 2001:db8::12:1:2/112
set interfaces xe-0/0/0:0 description "NETWORK LINK"
set interfaces xe-0/0/0:0 flexible-vlan-tagging
set interfaces xe-0/0/0:0 encapsulation extended-vlan-bridge
set interfaces xe-0/0/0:0 unit 100 vlan-id 100
set interfaces xe-0/0/0:0 unit 110 vlan-id 110

set interfaces lo0 unit 0 family inet6 address 2001:db8::2:1/128 primary
set forwarding-options evpn-vxlan shared-tunnels
set routing-instances USER_MVS1 instance-type mac-vrf
set routing-instances USER_MVS1 protocols evpn encapsulation vxlan
set routing-instances USER_MVS1 vtep-source-interface lo0.0 inet6
set routing-instances USER_MVS1 service-type vlan-aware
set routing-instances USER_MVS1 route-distinguisher 64500:11000002
set routing-instances USER_MVS1 vrf-target target:64500:1110
set routing-instances USER_MVS1 vrf-target auto
set routing-instances USER_MVS1 vlans V100 interface xe-0/0/0:0.100
set routing-instances USER_MVS1 vlans V100 vxlan vni 1100
set routing-instances USER_MVS1 vlans V110 interface xe-0/0/0:0.110
set routing-instances USER_MVS1 vlans V110 interface ae10.110
set routing-instances USER_MVS1 vlans V110 vxlan vni 1110
set protocols ospf3 area 0.0.0.0 interface xe-0/0/34:0.0
set protocols ospf3 area 0.0.0.0 interface ae0.0
set protocols ospf3 area 0.0.0.0 interface lo0.0 passive
set routing-options router-id 192.168.2.1
set routing-options autonomous-system 64500
set protocols bgp group vteps type internal

```



```

set protocols bgp group vteps local-address 2001:db8::2:1
set protocols bgp group vteps family evpn signaling
set protocols bgp group vteps neighbor 2001:db8::1
set protocols bgp group vteps neighbor 2001:db8::1:1
set protocols bgp group vteps neighbor 2001:db8::3:1
set protocols bgp group vteps neighbor 2001:db8::4:1
set interfaces xe-0/0/26:0 description "TO CE-1"
set interfaces xe-0/0/26:0 ether-options 802.3ad ae10
set interfaces ae10 flexible-vlan-tagging
set interfaces ae10 encapsulation extended-vlan-bridge
set interfaces ae10 esi 00:10:11:12:13:14:15:16:17:01
set interfaces ae10 esi all-active
set interfaces ae10 aggregated-ether-options link-speed mixed
set interfaces ae10 aggregated-ether-options lacp active
set interfaces ae10 aggregated-ether-options lacp periodic fast
set interfaces ae10 aggregated-ether-options lacp system-id 00:11:11:03:04:01
set interfaces ae10 unit 110 vlan-id 110

```

Step-by-Step Procedure on Leaf 1

1. Configure the interfaces for the EVPN fabric device connections. For illustrative purposes, in this example Leaf 1 connects to Spine 1 with an aggregated Ethernet (AE) interface bundle and to Spine 2 with a single interface.

```

set interfaces xe-0/0/18:0 description "LINK-0 to SPINE-1"
set interfaces xe-0/0/18:0 ether-options 802.3ad ae0
set interfaces xe-0/0/18:1 description "LINK-1 to SPINE-1"
set interfaces xe-0/0/18:1 ether-options 802.3ad ae0
set interfaces ae0 aggregated-ether-options link-speed mixed
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 unit 0 family inet6 address 2001:db8::02:1:2/112

set interfaces xe-0/0/34:0 description "LINK to SPINE-2"
set interfaces xe-0/0/34:0 unit 0 family inet6 address 2001:db8::12:1:2/112

```

2. Configure an interface for network traffic and the associated VLANs. This example uses a service provider style interface configuration.

```
set interfaces xe-0/0/0:0 description "NETWORK LINK"
set interfaces xe-0/0/0:0 flexible-vlan-tagging
set interfaces xe-0/0/0:0 encapsulation extended-vlan-bridge
set interfaces xe-0/0/0:0 unit 100 vlan-id 100
set interfaces xe-0/0/0:0 unit 110 vlan-id 110
```

3. Assign an IPv6 address to the loopback interface on this device.

```
set interfaces lo0 unit 0 family inet6 address 2001:db8::2:1/128 primary
```

4. (QFX5120 switches only) A device might have problems with VTEP scaling when the configuration uses multiple MAC-VRF instances. As a result, to avoid this problem, we require that you enable the shared tunnels feature on QFX5120 switches when setting up an IPv6 underlay. When you configure the shared-tunnels option, the device minimizes the number of next-hop entries to reach remote VTEPs.

Include the following statement to globally enable shared VXLAN tunnels on the device:

```
set forwarding-options evpn-vxlan shared-tunnels
```



NOTE: After you configure the shared-tunnels option, you must reboot the device for the setting to take effect.

The shared tunnels feature is:

- Enabled by default on QFX5130-32CD and QFX5700 switches, so you don't need to explicitly set this option on those switches.
 - Optional on the QFX10000 line of switches, because those switches can handle higher VTEP scaling.
5. (Required with EVPN-VXLAN on PTX Series routers only) Enable tunnel termination on the device. Set the *tunnel-termination* option either globally (for all interfaces) at the [edit forwarding-options]

hierarchy level, or for an particular physical interface at the [edit interfaces *name* unit *unit-number* family inet6] hierarchy level. Here we set tunnel-termination globally:

```
set forwarding-options tunnel-termination
```

6. Create an EVPN-VXLAN MAC-VRF instance. To use an IPv6 underlay, you also configure the device loopback interface as an IPv6 VTEP source interface.

In this step you also configure the following elements in the MAC-VRF instance:

- Set the VLAN-aware Ethernet service type so you can associate multiple VLANs with the instance.
- Assign a route distinguisher for the instance.
- Assign the route target.

We also set the `auto route target` option here, which uses one target for both import and export and helps to simplify the configuration.

```
set routing-instances USER_MVS1 instance-type mac-vrf
set routing-instances USER_MVS1 protocols evpn encapsulation vxlan
set routing-instances USER_MVS1 vtep-source-interface lo0.0 inet6
set routing-instances USER_MVS1 service-type vlan-aware
set routing-instances USER_MVS1 route-distinguisher 64500:11000002
set routing-instances USER_MVS1 vrf-target target:64500:1110
set routing-instances USER_MVS1 vrf-target auto
```

7. Configure the VLANs associated with the MAC-VRF instance and VLAN to VNI mappings—in this example, VLAN 100 (VNI 1100) and VLAN 110 (VNI 1110). This step includes the access-side ESI interface in the instance as well (ae10, which you configure in the last step).

```
set routing-instances USER_MVS1 vlans V100 interface xe-0/0/0:0.100
set routing-instances USER_MVS1 vlans V100 vxlan vni 1100
set routing-instances USER_MVS1 vlans V110 interface xe-0/0/0:0.110
set routing-instances USER_MVS1 vlans V110 interface ae10.110
set routing-instances USER_MVS1 vlans V110 vxlan vni 1110
```

8. Set up the IPv6 underlay. This example uses OSPFv3 for the IPv6 underlay connectivity.



NOTE: You might alternatively use BGP (for example, external BGP [eBGP]) as the IPv6 underlay routing protocol.

```
set protocols ospf3 area 0.0.0.0 interface xe-0/0/34:0.0
set protocols ospf3 area 0.0.0.0 interface ae0.0
set protocols ospf3 area 0.0.0.0 interface lo0.0 passive
```

9. Set up the IPv6 overlay. This example uses IBGP as the overlay routing protocol for EVPN with VXLAN tunneling. Our example is based on a pure IPv6 overlay. We must explicitly configure a router ID because we have no IPv4 addresses assigned to any interface. In this example, we configure the router ID as an arbitrary 32-bit integer in dotted quad notation.



NOTE: When running IPv6 routing protocols, you must configure a router ID for proper operation. The router ID must be a 4-octet unsigned non-zero integer that is unique in the routing domain.

For simplicity, the router ID is often set to match an IPv4 address on the router, typically a loopback address. While the router ID looks like an IPv4 address, there is no need for it to be routable, or that it be assigned to any interfaces on the device. In an IPv6-based overlay, we use the IPv6 loopback address for the VTEP local address.

```
set routing-options router-id 192.168.2.1
set routing-options autonomous-system 64500
set protocols bgp group vteps type internal
set protocols bgp group vteps local-address 2001:db8::2:1
set protocols bgp group vteps family evpn signaling
set protocols bgp group vteps neighbor 2001:db8::1
set protocols bgp group vteps neighbor 2001:db8::1:1
set protocols bgp group vteps neighbor 2001:db8::3:1
set protocols bgp group vteps neighbor 2001:db8::4:1
```

10. Set up an Ethernet segment (ESI) from Leaf 1 to CE 1, which is multihomed to Leaf 1 and Leaf 2. You would configure the ESI on Leaf 2 similarly. For simplicity, this example doesn't show the Leaf 2 configuration.

```
set interfaces xe-0/0/26:0 description "TO CE-1"
set interfaces xe-0/0/26:0 ether-options 802.3ad ae10
set interfaces ae10 flexible-vlan-tagging
```

```

set interfaces ae10 encapsulation extended-vlan-bridge
set interfaces ae10 esi 00:10:11:12:13:14:15:16:17:01
set interfaces ae10 esi all-active
set interfaces ae10 aggregated-ether-options link-speed mixed
set interfaces ae10 aggregated-ether-options lacp active
set interfaces ae10 aggregated-ether-options lacp periodic fast
set interfaces ae10 aggregated-ether-options lacp system-id 00:11:11:03:04:01
set interfaces ae10 unit 110 vlan-id 110

```

Configure Leaf 3

IN THIS SECTION

- [CLI Quick Configuration on Leaf 3 | 916](#)
- [Step-by-Step Procedure on Leaf 3 | 917](#)

CLI Quick Configuration on Leaf 3

To quickly configure Leaf 3 with an IPv6 underlay according to [Figure 83 on page 910](#), 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-0/0/18:0 description "LINK TO SPINE-1"
set interfaces xe-0/0/18:0 unit 0 family inet6 address 2001:db8::04:1:2/112
set interfaces xe-0/0/70:0 description "LINK TO SPINE-2"
set interfaces xe-0/0/70:0 unit 0 family inet6 address 2001:db8::14:1:2/112
set interfaces xe-0/0/35:3 description "NETWORK LINK"
set interfaces xe-0/0/35:3 flexible-vlan-tagging
set interfaces xe-0/0/35:3 native-vlan-id 110
set interfaces xe-0/0/35:3 encapsulation extended-vlan-bridge
set interfaces xe-0/0/35:3 unit 100 vlan-id 100
set interfaces xe-0/0/35:3 unit 110 vlan-id 110

set interfaces lo0 unit 0 family inet6 address 2001:db8::4:1/128 primary
set forwarding-options evpn-vxlan shared-tunnels
set routing-instances USER_MVS1 instance-type mac-vrf
set routing-instances USER_MVS1 protocols evpn encapsulation vxlan

```

```

set routing-instances USER_MVS1 vtep-source-interface lo0.0 inet6
set routing-instances USER_MVS1 service-type vlan-aware
set routing-instances USER_MVS1 route-distinguisher 64500:11000004
set routing-instances USER_MVS1 vrf-target target:64500:1110
set routing-instances USER_MVS1 vrf-target auto
set routing-instances USER_MVS1 vlans V100 interface xe-0/0/35:3.100
set routing-instances USER_MVS1 vlans V100 vxlan vni 1100
set routing-instances USER_MVS1 vlans V110 interface xe-0/0/35:3.110
set routing-instances USER_MVS1 vlans V110 vxlan vni 1110
set protocols ospf3 area 0.0.0.0 interface xe-0/0/18:0.0
set protocols ospf3 area 0.0.0.0 interface xe-0/0/70:0.0
set protocols ospf3 area 0.0.0.0 interface lo0.0 passive
set routing-options router-id 192.168.4.1
set routing-options autonomous-system 64500
set protocols bgp group vteps type internal
set protocols bgp group vteps local-address 2001:db8::4:1
set protocols bgp group vteps family evpn signaling
set protocols bgp group vteps neighbor 2001:db8::1
set protocols bgp group vteps neighbor 2001:db8::1:1
set protocols bgp group vteps neighbor 2001:db8::2:1
set protocols bgp group vteps neighbor 2001:db8::3:1

```

Step-by-Step Procedure on Leaf 3

1. Configure the interfaces for the EVPN fabric device connections from Leaf 3 to Spine 1 and Spine 2.

```

set interfaces xe-0/0/18:0 description "LINK TO SPINE-1"
set interfaces xe-0/0/18:0 unit 0 family inet6 address 2001:db8::04:1:2/112
set interfaces xe-0/0/70:0 description "LINK TO SPINE-2"
set interfaces xe-0/0/70:0 unit 0 family inet6 address 2001:db8::14:1:2/112

```

2. Configure an interface for network traffic and the associated VLANs. This example uses a service provider style interface configuration.

```

set interfaces xe-0/0/35:3 description "NETWORK LINK"
set interfaces xe-0/0/35:3 flexible-vlan-tagging
set interfaces xe-0/0/35:3 native-vlan-id 110
set interfaces xe-0/0/35:3 encapsulation extended-vlan-bridge
set interfaces xe-0/0/35:3 unit 100 vlan-id 100
set interfaces xe-0/0/35:3 unit 110 vlan-id 110

```

3. Assign an IPv6 address to the loopback interface on this device.

```
set interfaces lo0 unit 0 family inet6 address 2001:db8::4:1/128 primary
```

4. (QFX5120 switches only) A device might have problems with VTEP scaling when the configuration uses multiple MAC-VRF instances. As a result, to avoid this problem, we require that you enable the shared tunnels feature on QFX5120 switches when setting up an IPv6 underlay. When you configure the shared-tunnels option, the device minimizes the number of next-hop entries to reach remote VTEPs.

Include the following statement to globally enable shared VXLAN tunnels on the device:

```
set forwarding-options evpn-vxlan shared-tunnels
```



NOTE: The shared tunnels feature is enabled by default on QFX5130-32CD and QFX5700 switches, so you don't need to explicitly set this option on those switches. This statement is optional on the QFX10000 line of switches, which can handle higher VTEP scaling.

5. Create an EVPN-VXLAN MAC-VRF instance. To use an IPv6 underlay, you configure the device loopback interface as an IPv6 VTEP source interface in this step (although you configure the IPv6 underlay itself in a later step.)

In this step you also configure the following elements in the MAC-VRF instance:

- Set the VLAN-aware Ethernet service type so you can associate multiple VLANs with the instance.
- Assign a route distinguisher for the instance.
- Assign the route target.

We also set the auto route target option here, which uses one target for both import and export and helps to simplify the configuration.

```
set routing-instances USER_MVS1 instance-type mac-vrf
set routing-instances USER_MVS1 protocols evpn encapsulation vxlan
set routing-instances USER_MVS1 vtep-source-interface lo0.0 inet6
set routing-instances USER_MVS1 service-type vlan-aware
set routing-instances USER_MVS1 route-distinguisher 64500:11000004
```

```
set routing-instances USER_MVS1 vrf-target target:64500:1110
set routing-instances USER_MVS1 vrf-target auto
```

6. Configure the VLANs associated with the MAC-VRF instance and VLAN to VNI mappings—in this example, VLAN 100 (VNI 1100) and VLAN 110 (VNI 1110).

```
set routing-instances USER_MVS1 vlans V100 interface xe-0/0/35:3.100
set routing-instances USER_MVS1 vlans V100 vxlan vni 1100
set routing-instances USER_MVS1 vlans V110 interface xe-0/0/35:3.110
set routing-instances USER_MVS1 vlans V110 vxlan vni 1110
```

7. Set up the IPv6 underlay. This example uses OSPFv3 for the IPv6 underlay connectivity.



NOTE: You might alternatively use BGP (for example, external BGP [eBGP]) as the IPv6 underlay routing protocol.

```
set protocols ospf3 area 0.0.0.0 interface xe-0/0/18:0.0
set protocols ospf3 area 0.0.0.0 interface xe-0/0/70:0.0
set protocols ospf3 area 0.0.0.0 interface lo0.0 passive
```

8. Set up the IPv6 overlay. This example uses using IBGP as the overlay routing protocol for EVPN with VXLAN tunneling.



NOTE: Even though we use the IPv6 address family, you must configure a router ID for proper operation. The router ID must be a 4-octet unsigned non zero integer that is unique in the routing domain. You configure the router ID using dotted quad notation. In an IPv6 based overlay, we use the IPv6 loopback address for the VTEP local address.

```
set routing-options router-id 192.168.4.1
set routing-options autonomous-system 64500
set protocols bgp group vteps type internal
set protocols bgp group vteps local-address 2001:db8::4:1
set protocols bgp group vteps family evpn signaling
set protocols bgp group vteps neighbor 2001:db8::1
set protocols bgp group vteps neighbor 2001:db8::1:1
```



```
set protocols bgp group vteps neighbor 2001:db8::2:1
set protocols bgp group vteps neighbor 2001:db8::3:1
```

Verify the IPv6 Underlay on Leaf 3

IN THIS SECTION

- [Verify Peer Device Connectivity | 921](#)
- [Verify VTEP Source Parameters | 922](#)
- [Verify Remote VTEPs | 922](#)
- [Verify MAC-VRF EVPN Instance Forwarding | 924](#)

Use the CLI commands in this section to verify the IPV6 underlay configuration is operational on the leaf devices in this example. This section shows the results from running these commands on Leaf 3.

This example includes show mac-vrf forwarding *command-name* commands that display information for MAC-VRF instance configurations. Most show mac-vrf forwarding commands are aliases for the same command in the following command hierarchies that you might use for the default switching instance or other instance types:

- QFX Series switches—show ethernet-switching *command-name*
- MX Series routers and EX9200 line of switches—show l2-learning *command-name* or show bridge *command-name*

See "[MAC-VRF Routing Instance Type Overview](#)" on page 38 for a full list of the MAC-VRF instance show commands and their mappings to the commands that display equivalent results for other instances.

On devices with multiple MAC-VRF EVPN instances, to avoid VTEP scaling issues, we might require or recommend that you enable the shared tunnels feature. On some platforms, shared tunnels are enabled by default. In this example, we enable shared tunnels on the leaf devices using the set forwarding-options evpn-vxlan shared-tunnels configuration statement. MAC-VRF show commands display shared tunnel VTEP interfaces as vtep-*index* *shared-tunnel-unit* , where:

- *index* is the index associated with the MAC-VRF routing instance.
- *shared-tunnel-unit* is the unit number associated with the shared tunnel remote VTEP logical interface.

For example:

```
vtep-7.32772
```



NOTE: After you configure the `shared-tunnels` option, you must reboot the device for the setting to take effect.

Verify Peer Device Connectivity

Purpose

Check that the leaf device established BGP IPv6 connectivity to its peer spine and leaf devices in the fabric.

Action

Run the `show bgp summary` command on the leaf device:

```
user@leaf-3> show bgp summary
```

```
Threading mode: BGP I/O
```

```
Default eBGP mode: advertise - accept, receive - accept
```

```
Groups: 1 Peers: 5 Down peers: 0
```

Table	Tot Paths	Act Paths	Suppressed	History	Damp	State	Pending
bgp.evpn.0	33	33	0	0	0	0	0
Peer	AS	InPkt	OutPkt	OutQ	Flaps	Last Up/Dwn	State #Active/Received/Accepted/Damped...
2001:db8::1	64500	1428	1406	0	0	10:38:00	Establ
bgp.evpn.0: 10/10/10/0							
USER_MVS1.evpn.0: 10/10/10/0							
__default_evpn__.evpn.0: 0/0/0/0							
2001:db8::1:1	64500	1435	1406	0	0	10:38:01	Establ
bgp.evpn.0: 10/10/10/0							
USER_MVS1.evpn.0: 10/10/10/0							
__default_evpn__.evpn.0: 0/0/0/0							
2001:db8::2:1	64500	1578	1415	0	0	10:37:58	Establ
bgp.evpn.0: 5/5/5/0							
USER_MVS1.evpn.0: 5/5/5/0							

```

__default_evpn__.evpn.0: 0/0/0/0
2001:db8::3:1      64500      1483      1406      0      0      10:37:55 Establ
bgp.evpn.0: 3/3/3/0
USER_MVS1.evpn.0: 3/3/3/0
__default_evpn__.evpn.0: 0/0/0/0

```

Meaning

Leaf 3 (IPv6 address 2001:db8::3:1 in [Figure 83 on page 910](#)) sees its eBGP peer devices Spine 1 (2001:db8::1), Spine 2(2001:db8::1:1), Leaf 1(2001:db8::2:1) and Leaf 2 (2001:db8::3:1).

Verify VTEP Source Parameters

Purpose

View the configured IPv6 VTEP source interface(s).

Action

Run the `show mac-vrf forwarding vxlan-tunnel-end-point source` command:

```

user@leaf-3> show mac-vrf forwarding vxlan-tunnel-end-point source
Logical System Name      Id  SVTEP-IP      IFL  L3-Idx  SVTEP-Mode  ELP-SVTEP-IP
<default>                0   2001:db8::4:1 lo0.0 0
  L2-RTT                  Bridge Domain          VNID   Translation-VNID  MC-Group-IP
  USER_MVS1              V110                  1110          ::

```

Meaning

The output shows you configured Leaf 3 with IPv6 VTEP source address 2001:db8::4:1 on the loopback port in MAC-VRF instance USER-MVS1 for VLAN V110, which you mapped to VNI 1110.

Verify Remote VTEPs

Purpose

Verify the device has forwarding information for the remote VTEPs.

Action

Run the show mac-vrf forwarding vxlan-tunnel-end-point remote command:

```
user@leaf-3> show mac-vrf forwarding vxlan-tunnel-end-point remote
Logical System Name      Id  SVTEP-IP      IFL  L3-Idx  SVTEP-Mode  ELP-SVTEP-IP
<default>                0   2001:db8::4:1  lo0.0  0
RVTEP-IP      IFL-Idx  Interface  NH-Id  RVTEP-Mode  ELP-IP      Flags
2001:db8::1    612    vtep.32771  1758   RNVE
2001:db8::1:1  611    vtep.32770  1753   RNVE
2001:db8::2:1  613    vtep.32772  1759   RNVE
2001:db8::3:1  614    vtep.32773  1761   RNVE
RVTEP-IP      L2-RTT                IFL-Idx  Interface  NH-Id  RVTEP-Mode  ELP-IP
Flags
2001:db8::1    USER_MVS1                671145987 vtep-7.32771 1758   RNVE
  VNID          MC-Group-IP
  1110          ::
RVTEP-IP      L2-RTT                IFL-Idx  Interface  NH-Id  RVTEP-Mode  ELP-IP
Flags
2001:db8::1:1  USER_MVS1                671145986 vtep-7.32770 1753   RNVE
  VNID          MC-Group-IP
  1110          ::
RVTEP-IP      L2-RTT                IFL-Idx  Interface  NH-Id  RVTEP-Mode  ELP-IP
Flags
2001:db8::2:1  USER_MVS1                671145988 vtep-7.32772 1759   RNVE
  VNID          MC-Group-IP
  1110          ::
RVTEP-IP      L2-RTT                IFL-Idx  Interface  NH-Id  RVTEP-Mode  ELP-IP
Flags
2001:db8::3:1  USER_MVS1                671145989 vtep-7.32773 1761   RNVE
  VNID          MC-Group-IP
  1110          ::
```

Meaning

The output shows that Leaf 3 has forwarding information for remote IPv6 VTEPs on Leaf 1 (2001:db8::2:1) and Leaf 2 (2001:db8::3:1).

Verify MAC-VRF EVPN Instance Forwarding

Purpose

View the forwarding table for the configured MAC-VRF instance to see the interfaces for the remote VTEPs associated with the instance.

Action

Run the `show mac-vrf forwarding mac-table instance name` command for the MAC-VRF instance in this example, `USER_MVS1`:

```
user@leaf-3> show mac-vrf forwarding mac-table instance USER_MVS1

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 3 entries, 3 learned
Routing instance : USER_MVS1

Ethernet switching table : 3 entries, 3 learned
Routing instance : USER_MVS1
```

Vlan name	MAC address	MAC flags	Logical interface	SVLBNH/ VENH Index	Active source
V110	00:01:01:10:00:fe	DRP	esi.1754		
05:00:00:02:9a:00:00:04:56:00					
V110	54:4b:8c:d3:40:32	DRP	vtep-7.32771		
2001:db8::1					
V110	84:03:28:50:3c:e0	DRP	vtep-7.32770		
2001:db8::1:1					

Meaning

The output for this command shows the MAC addresses that were populated in the MAC table.

Multicast Features with EVPN-VXLAN

IN THIS CHAPTER

- [Multicast Support in EVPN-VXLAN Overlay Networks | 925](#)
- [Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment | 934](#)
- [Example: Preserving Bandwidth with IGMP Snooping in an EVPN-VXLAN Environment | 953](#)
- [Overview of Selective Multicast Forwarding | 1005](#)
- [Configuring the Number of SMET Nexthops | 1008](#)
- [Assisted Replication Multicast Optimization in EVPN Networks | 1010](#)
- [Optimized Intersubnet Multicast in EVPN Networks | 1049](#)
- [EVPN-VXLAN DCI Multicast with Enhanced OISM | 1188](#)

Multicast Support in EVPN-VXLAN Overlay Networks

IN THIS SECTION

- [Inter-VLAN Multicast Forwarding Modes for EVPN-VXLAN Overlay Networks | 925](#)

This topic describes the following multicast feature, which is supported in an EVPN-VXLAN overlay network:

Inter-VLAN Multicast Forwarding Modes for EVPN-VXLAN Overlay Networks

IN THIS SECTION

- [Centrally-routed Multicast Mode Support | 926](#)

- [Edge-routed Multicast Mode and OISM Support | 927](#)
- [Benefits of Inter-VLAN Multicast Forwarding Modes | 927](#)
- [Supported EVPN-VXLAN Architectures and Inter-VLAN Multicast Forwarding Modes | 927](#)
- [Understanding Multicast Traffic Flows in a Centrally-Routed Bridging Overlay | 928](#)
- [Understanding Multicast Traffic Flows in an Edge-Routed Bridging Overlay | 930](#)
- [Differences Between Inter-VLAN Multicast Forwarding Modes | 932](#)

We support a few different multicast forwarding modes based on the type of EVPN-VXLAN architecture you have in your network:

- Centrally-routed multicast mode (local-remote forwarding mode)
- Edge-routed multicast mode (local-only forwarding mode)
- Optimized intersubnet multicast (OISM) mode (combines aspects of local-remote and local-only forwarding modes)

This topic introduces the centrally-routed (local-remote) and edge-routed (local-only) multicast forwarding modes and how they work. For full details on OISM operation and configuration, including regular OISM and enhanced OISM modes, see ["Optimized Intersubnet Multicast in EVPN Networks" on page 1049](#).

Centrally-routed Multicast Mode Support

Starting with Junos OS Release 17.3R3, the QFX10000 line of switches support the centrally-routed mode for inter-VLAN multicast forwarding of IPv4 traffic in an EVPN-VXLAN network. You use this mode with a centrally-routed bridging (CRB) overlay network.

QFX5120 switches support IGMP snooping as leaf devices in multihomed EVPN-VXLAN fabrics in centrally-routed multicast mode starting with the indicated Junos OS releases below. QFX10000 switches serve as spine devices that perform the Layer 3 multicast routing in the centrally-routed bridging (CRB) model.

- QFX5120-48Y—Junos OS Release 18.4R2 (although not supported in releases 19.1R1 and 19.1R2)
- QFX5120-32C—Junos OS Release 19.1R1
- QFX5120-48T—Junos OS Release 20.2R1
- QFX5120-48YM—Junos OS Release 20.4R1

Starting with Junos OS Release 20.4R1, QFX5110, QFX5120, and the QFX10000 line of switches support the centrally-routed forwarding mode with MLDv1, MLDv2, and MLD snooping for IPv6 intra-VLAN and IPv6 inter-VLAN multicast traffic in an EVPN-VXLAN overlay network.

Edge-routed Multicast Mode and OISM Support

Starting with Junos OS Release 17.3R3, the QFX10000 line of switches support the edge-routed mode for inter-VLAN multicast forwarding of IPv4 traffic in an EVPN-VXLAN overlay network. You use this mode with an edge-routed bridging (ERB) overlay fabric.

Starting with Junos OS Release 21.2R1, we introduce the OISM routing and forwarding mode for ERB overlay EVPN-VXLAN fabrics on QFX5110, QFX5120, and QFX10002 switches support. The OISM mode enables the leaf devices in the network to efficiently handle multicast traffic routing with an ERB overlay for sources and receivers that are outside of the EVPN network. OISM also scales much better than the original edge-routed multicast mode. As a result, we recommend using OISM for multicast in ERB overlay networks.

Benefits of Inter-VLAN Multicast Forwarding Modes

- The configuration statement enables you to choose the inter-VLAN multicast forwarding mode that suits the architecture of your EVPN-VXLAN overlay network.

Supported EVPN-VXLAN Architectures and Inter-VLAN Multicast Forwarding Modes

We support the following modes of inter-VLAN multicast forwarding:

- EVPN-VXLAN CRB overlays (EVPN-VXLAN networks with a two-layer IP fabric) support central routing and forwarding of multicast traffic at the spine layer using a local-remote model. CRB overlay networks have a layer of Layer 3 spine devices and another layer of Layer 2 leaf devices. You configure all spine devices with IRB interfaces that route the multicast packets from one VLAN to another. One spine device is elected to handle the inter-VLAN routing for the network.

In this mode, you configure the devices in the fabric with the `irb local-remote` option at the `[edit forwarding-options multicast-replication evpn]` hierarchy. (See *multicast-replication*.)

- EVPN-VXLAN ERB overlays (EVPN-VXLAN networks with a collapsed IP fabric) support multicast traffic routing and forwarding at the leaf layer using either a local-only model or the OISM model. ERB overlay networks have leaf devices with IRB interfaces that handle multicast routing at Layer 3 and multicast forwarding at Layer 2. The leaf devices essentially act as spine-leaf devices. This fabric model usually has a layer of spine devices acting only as IP transit devices in the fabric, which we commonly refer to as *lean spines*. All of the leaf devices handle routing multicast packets from one VLAN to another.

You configure supported leaf devices in the ERB fabric with either the `irb local-only` option or `irb oism` option at the `[edit forwarding-options multicast-replication evpn]` hierarchy. (See *multicast-replication*.)



NOTE: This topic describes the local-only model.

See "[Optimized Intersubnet Multicast in EVPN Networks](#)" on page 1049 for full details on OISM configuration and operation.

For centrally-routed bridging overlays, you can simply retain the default setting, `irb local-remote`, at the `[edit forwarding-options multicast-replication evpn]` hierarchy level on all spine devices. For edge-routed bridging overlays, you must explicitly specify the `irb local-only` or `irb oism` option on all leaf devices.



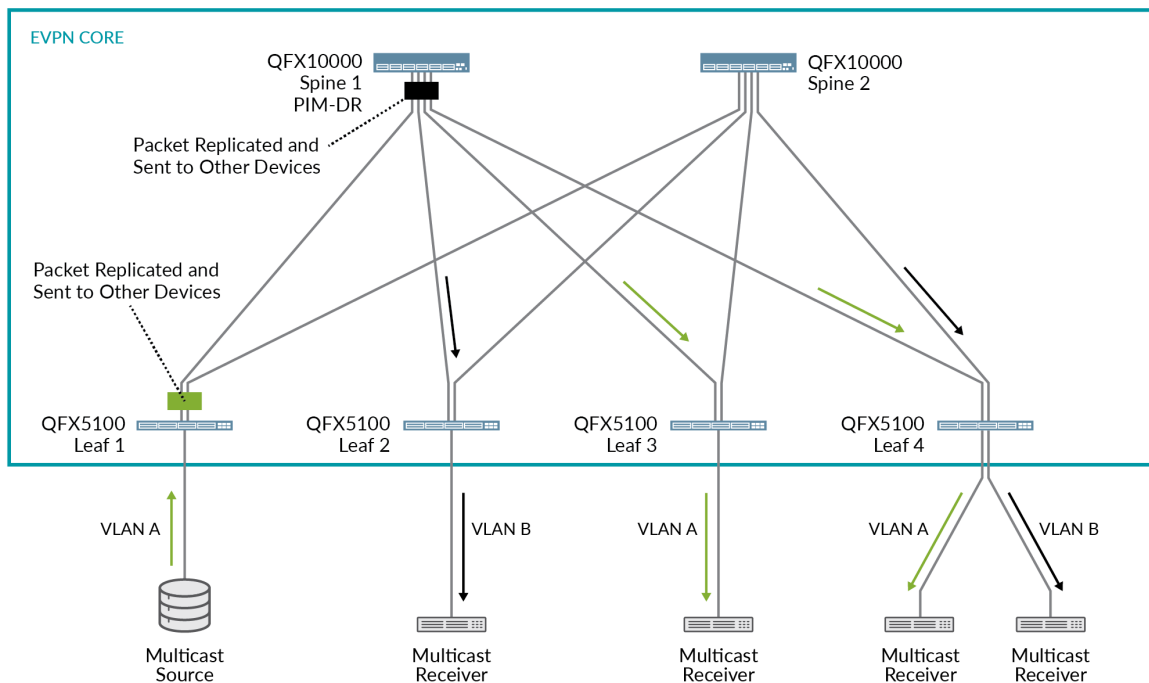
NOTE: We do not recommend specifying the `local-remote` option on some QFX10000 switches and the `local-only` option on the other QFX10000 switches in either of the overlay networks. Doing so might cause the QFX10000 switches to forward the inter-VLAN multicast traffic inconsistently.

Understanding Multicast Traffic Flows in a Centrally-Routed Bridging Overlay

This section describes the multicast traffic flows in a centrally-routed bridging overlay.

The network shown in [Figure 84 on page 929](#) includes the following devices.

Figure 84: Centrally-Routed Bridging Overlay



- Two QFX10000 switches that function as Layer 3 spine devices, on which the following key features are configured:
 - Centrally-routed (local-remote) multicast forwarding mode.
 - Protocol Independent Multicast (PIM). By virtue of PIM hello messages, Spine 1 is elected as the PIM designated router (PIM DR).
 - VLANs A and B.



NOTE: For inter-VLAN multicast forwarding to work properly in this scenario, you must configure all VLANs on each spine device.

- IRB interfaces associated with VLANs A and B.
- Four QFX5100 switches that function as Layer 2 leaf devices, on which VLANs A and B are configured are follows:
 - Leafs 1 and 3 are configured with VLAN A only.
 - Leaf 2 is configured with VLAN B only.
 - Leaf 4 is configured with VLANs A and B.

- A multicast source and various receivers.

When the multicast source shown in [Figure 84 on page 929](#) sends a packet from VLAN A, the following flows occur:

- **Flow 1: Intra-VLAN traffic**—Based on the ingress replication mechanism, Leaf 1 replicates and switches the packet to all spine and other leaf devices. Leafs 3 and 4, on which VLAN A is configured, receive and forward the packet to the connected multicast receivers.
- **Flow 2: Inter-VLAN traffic**—Upon receipt of the packet from Leaf 1 as described in flow 1, Spine 1, which is the PIM DR, takes the following action:
 - Routes the packet over the IRB interface associated with VLAN B.
 - Based on the ingress replication mechanism, replicates and forwards the packet to the other spine and the leaf devices.

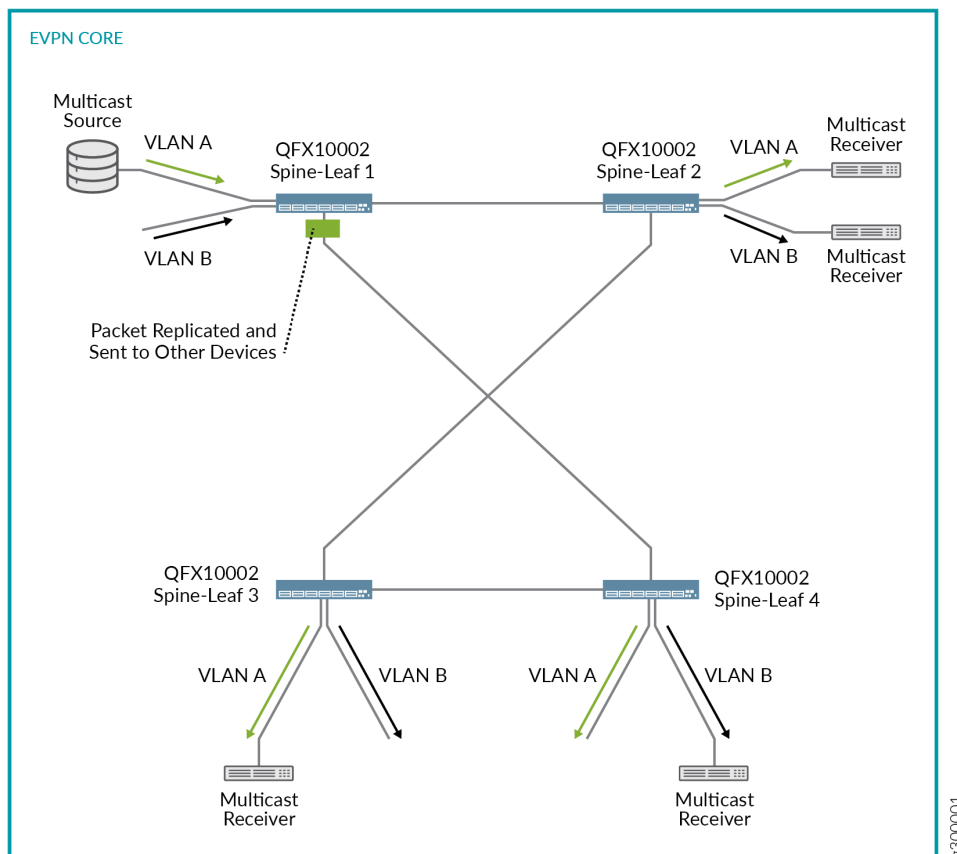
Leafs 2 and 4, on which VLAN B is configured, receive and forward the packet to the connected multicast receivers.

Understanding Multicast Traffic Flows in an Edge-Routed Bridging Overlay

This section describes the multicast traffic flow in an edge-routed bridging overlay.

The network shown in [Figure 85 on page 931](#) includes the following devices.

Figure 85: Edge-Routed Bridging Overlay



- Four QFX10002 switches that function as Layer 3 and Layer 2 spine-leaf devices, on which the following key features are configured:
 - Edge-routed (local-only) multicast forwarding mode.
 - PIM. To support the edge-routed mode of multicast forwarding, each spine-leaf device must act as the PIM DR for each VLAN. To enable a spine-leaf device to elect itself as the PIM DR, for each IRB interface, specify the `distributed-dr` configuration statement at the `[edit protocols pim interface interface-name]` hierarchy.
 - VLANs A and B.



NOTE: For inter-VLAN multicast forwarding to work properly in this scenario, you must configure all VLANs on each spine-leaf device.

- IRB interfaces associated with VLANs A and B.

- A multicast source and various receivers.

When the multicast source shown in [Figure 85 on page 931](#) sends a packet from VLAN A, the following flows occur:

- **Flow 1: Intra-VLAN traffic**—Based on the ingress replication mechanism, Spine-Leaf 1 replicates and switches the packet to the other spine-leaf devices. The spine-leaf devices forward the packet to VLAN A. Further, Spine-Leafs 2 and 3 forward the packet to VLAN A receivers.
- **Flow 2: Inter-VLAN traffic**—Upon receipt of the packet from Spine-Leaf 1 as described in flow 1, the spine-leaf devices route the packet over the IRB interface associated with VLAN B. Further, Spine-Leafs 2 and 4 forward the packet to the VLAN B receivers.

Differences Between Inter-VLAN Multicast Forwarding Modes

There is an important difference between the centrally-routed and edge-routed modes.

With centrally-routed mode, for each VLAN, the spine device elected as the PIM-DR must replicate and send the packet to the other devices in the network. Keep in mind that the additional instances of replication consume bandwidth, and if many VLANs are configured, can potentially flood the EVPN core with multicast packets.

With edge-routed mode with the local-only model, the first spine-leaf device to receive a multicast packet replicates and sends the packet to the other spine-leaf devices. Upon receipt of the packet, the other spine-leaf devices route the packet to each VLAN and send the packet to access-side interfaces that handle traffic for the multicast receivers. In other words, the spine-leaf devices do not replicate the packet and send the packet out of the EVPN core-facing interfaces, which prevents excessive bandwidth consumption and congestion in the EVPN core.

The OISM model uses local routing, similar to the local-only model, to minimize multicast traffic flow in the EVPN core. However, with OISM, the leaf devices can also efficiently handle multicast traffic flow from sources outside the fabric to receivers within the fabric. OISM similarly enables multicast sources inside the fabric to send traffic to multicast receivers outside the fabric. See "[Optimized Intersubnet Multicast in EVPN Networks](#)" on page 1049 for details on OISM configuration and operation.

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
21.2R1	Starting with Junos OS Release 21.2R1, QFX5110, QFX5120, and QFX10002 switches support the optimized intersubnet multicast (OISM) forwarding mode for edge-routed bridging overlay EVPN-VXLAN fabrics.

20.4R1	Starting with Junos OS Release 20.4R1, QFX5120-48YM switches support IGMP snooping as leaf devices in multihomed EVPN-VXLAN fabrics in centrally-routed multicast mode. (QFX10000 switches are the spine devices that perform the multicast routing.)
20.4R1	Starting with Junos OS Release 20.4R1, QFX5110, QFX5120, and the QFX10000 line of switches support centrally-routed forwarding mode with MLDv1, MLDv2, and MLD snooping for IPv6 intra-VLAN and IPv6 inter-VLAN multicast traffic in an EVPN-VXLAN overlay network.
20.2R1	Starting with Junos OS Release 20.2R1, QFX5120-48T switches support IGMP snooping as leaf devices in multihomed EVPN-VXLAN fabrics in centrally-routed multicast mode. (QFX10000 switches are the spine devices that perform the multicast routing.)
19.1R1	Starting with Junos OS Release 19.1R1, QFX5120-32C switches support IGMP snooping as leaf devices in multihomed EVPN-VXLAN fabrics in centrally-routed multicast mode. (QFX10000 switches are the spine devices that perform the multicast routing.)
18.4R2	Starting with Junos OS Release 18.4R2, QFX5120-48Y switches support IGMP snooping as leaf devices in multihomed EVPN-VXLAN fabrics in centrally-routed multicast mode. (QFX10000 switches are the spine devices that perform the multicast routing.)
17.3R3	Starting with Junos OS Release 17.3R3, the QFX10000 line of switches support two modes of inter-VLAN multicast forwarding—centrally-routed mode and edge-routed mode—in an EVPN-VXLAN overlay network. The IP fabric architecture of your EVPN-VXLAN overlay network determines the mode that you must use.

RELATED DOCUMENTATION

[Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment | 934](#)

[Optimized Intersubnet Multicast in EVPN Networks | 1049](#)

Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment

IN THIS SECTION

- Benefits of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment | 936
- Supported IGMP or MLD Versions and Group Membership Report Modes | 936
- Summary of Multicast Traffic Forwarding and Routing Use Cases | 937
- Use Case 1: Intra-VLAN Multicast Traffic Forwarding | 938
- Use Case 2: Inter-VLAN Multicast Routing and Forwarding—IRB Interfaces with PIM | 941
- Use Case 3: Inter-VLAN Multicast Routing and Forwarding—PIM Gateway with Layer 2 Connectivity | 945
- Use Case 4: Inter-VLAN Multicast Routing and Forwarding—PIM Gateway with Layer 3 Connectivity | 947
- Use Case 5: Inter-VLAN Multicast Routing and Forwarding—External Multicast Router | 949
- EVPN Multicast Flags Extended Community | 949
- Platform-Specific EVPN-VXLAN Multicast Behavior with IGMP Snooping and MLD Snooping | 950

Internet Group Management Protocol (IGMP) snooping and Multicast Listener Discovery (MLD) snooping constrain multicast traffic in a broadcast domain to interested receivers and multicast devices. In an environment with a significant volume of multicast traffic, using IGMP or MLD snooping preserves bandwidth because multicast traffic is forwarded only on those interfaces that have multicast listeners. IGMP snooping optimizes IPv4 multicast traffic flow. MLD snooping optimizes IPv6 multicast traffic flow.

We support IGMP snooping and MLD snooping in Ethernet VPN-Virtual extensible LAN (EVPN-VXLAN) networks as a way to help optimize multicast traffic flow.

Starting in Junos OS Release 21.2R1, on some platforms we also support optimized intersubnet multicast (OISM) routing and forwarding in EVPN-VXLAN edge-routed bridging (ERB) overlay networks. See ["Optimized Intersubnet Multicast in EVPN Networks" on page 1049](#) for full details on OISM configuration and operation. OISM enables efficient multicast routing and forwarding for both internal and external multicast sources and receivers in ERB overlay fabrics. You must configure IGMP snooping (or, if supported, MLD snooping) on the fabric leaf devices as part of OISM configuration. On some platforms, we support EVPN-VXLAN multicast traffic only with OISM.

This topic provides an introduction to available multicast optimization methods without OISM in EVPN-VXLAN CRB overlays or ERB overlays.

EVPN-VXLAN devices might support IGMPv2 and IGMPv3 with IGMP snooping for IPv4 multicast traffic. The devices might also support MLDv1 and MLDv2 with MLD snooping for IPv6 multicast traffic.

- With IGMPv2 and MLDv1, the device processes only any-source multicast (ASM) reports.
- If the device supports IGMPv3 and MLDv2, ASM mode is the default behavior, but you can configure the device to process source-specific multicast (SSM) reports with IGMPv3 and MLDv2 instead.

However, the device can't process both SSM reports and ASM reports at the same time. When you configure a device to operate in SSM mode with IGMPv3 and MLDv2, the device drops any ASM reports. When you don't configure the device to operate in SSM mode, a device processes any ASM reports but drops IGMPv3 and MLDv2 SSM reports.

Unless mentioned explicitly, the information in this topic applies to IGMPv2, IGMPv3, MLDv1, and MLDv2 on the devices that support these protocols in the following IP fabric architectures:

- EVPN-VXLAN ERB overlay
- EVPN-VXLAN CRB overlay



NOTE: On a switching devices, you can configure a *VLAN*. On routing devices, you can configure the same entity called a *bridge domain*. To keep things simple, this topic uses the term *VLAN* when we refer to the same entity configured on either switching or routing devices.

Here are a few important general notes about IGMP snooping and MLD snooping behavior with EVPN-VXLAN:



NOTE:

- On switches in the QFX5000 line, in releases up until Junos OS Releases 18.4R2 and 19.1R2, with IGMP snooping enabled the switches only constrain flooding for multicast traffic coming in on the VXLAN tunnel network ports. The switches would still flood multicast traffic coming in from an access interface to all other access and network interfaces.
- Starting with Junos release 22.3R1, on some platforms we support IPv4 and IPv6 multicast traffic in an IPv6 EVPN-VXLAN overlay with IPv6 underlay peering.

Benefits of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment

- In an environment with a significant volume of multicast traffic, using IGMP snooping or MLD snooping constrains multicast traffic in a VLAN to interested receivers and multicast devices, which conserves network bandwidth.
- Synchronizing the IGMP or MLD state among all EVPN devices for multihomed receivers ensures that all subscribed listeners receive multicast traffic, even in cases such as the following:
 - IGMP or MLD membership reports for a multicast group might arrive on an EVPN device that is not the Ethernet segment's designated forwarder (DF).
 - An IGMP or MLD message to leave a multicast group arrives at a different EVPN device than the EVPN device where the corresponding join message for the group was received.
- Selective multicast forwarding conserves bandwidth usage in the EVPN core and reduces the load on egress EVPN devices that do not have listeners.
- The support of external PIM gateways enables the exchange of multicast traffic between sources and listeners in an EVPN-VXLAN network and sources and listeners in an external PIM domain. Without this support, the sources and listeners in these two domains would not be able to communicate.

Supported IGMP or MLD Versions and Group Membership Report Modes

Table 50 on page 936 outlines the supported IGMP versions and the membership report modes supported for each version.

Table 50: Supported IGMP and MLD Versions and Group Membership Report Modes

IGMP Versions	Any-Source Multicast (ASM) (*,G) Only	Source-Specific Multicast (SSM) (S,G) Only	ASM (*,G) + SSM (S,G)
IGMPv2	Yes (default)	No	No
IGMPv3	Yes (default)	Yes (if configured)	No
MLDv1	Yes (default)	No	No
MLDv2	Yes (default)	Yes (if configured)	No

To explicitly configure EVPN devices to process only SSM (S,G) membership reports for IGMPv3 or MLDv2, set the `evpn-ssm-reports-only` configuration option at the `[edit protocols igmp-snooping vlan vlan-name]` hierarchy level.

You can enable SSM-only processing for one or more VLANs in an EVPN routing instance (EVI). When enabling this option for a routing instance of type `virtual switch`, the behavior applies to all VLANs in the virtual switch instance. When you enable this option, the device doesn't process ASM reports and drops them.

If you don't configure the `evpn-ssm-reports-only` option, by default, EVPN devices process IGMPv2, IGMPv3, MLDv1, or MLDv2 ASM reports and drop IGMPv3 or MLDv2 SSM reports.

Summary of Multicast Traffic Forwarding and Routing Use Cases

[Table 51 on page 937](#) provides a summary of the multicast traffic forwarding and routing use cases that we support in EVPN-VXLAN networks and our recommendation for when you should apply a use case to your EVPN-VXLAN network.

Table 51: Supported Multicast Traffic Forwarding and Routing Use Cases and Recommended Usage

Use Case Number	Use Case Name	Summary	Recommended Usage
1	Intra-VLAN multicast traffic forwarding	Forwarding of multicast traffic to hosts within the same VLAN.	We recommend implementing this basic use case in all EVPN-VXLAN networks.
2	Inter-VLAN multicast routing and forwarding—IRB interfaces with PIM	IRB interfaces using PIM on Layer 3 EVPN devices. These interfaces route multicast traffic between source and receiver VLANs.	We recommend implementing this basic use case in all EVPN-VXLAN networks except when you prefer to use an external multicast router to handle inter-VLAN routing (see use case 5).
3	Inter-VLAN multicast routing and forwarding—PIM gateway with Layer 2 connectivity	A Layer 2 mechanism for a data center, which uses IGMP and PIM, to exchange multicast traffic with an external PIM domain.	We recommend this use case in either EVPN-VXLAN ERB overlays or EVPN-VXLAN CRB overlays.

Table 51: Supported Multicast Traffic Forwarding and Routing Use Cases and Recommended Usage
(Continued)

Use Case Number	Use Case Name	Summary	Recommended Usage
4	Inter-VLAN multicast routing and forwarding—PIM gateway with Layer 3 connectivity	A Layer 3 mechanism for a data center, which uses IGMP (or MLD) and PIM, to exchange multicast traffic with an external PIM domain.	We recommend this use case in EVPN-VXLAN CRB overlays only.
5	Inter-VLAN multicast routing and forwarding—external multicast router	Instead of IRB interfaces on Layer 3 EVPN devices, an external multicast router handles inter-VLAN routing.	We recommend this use case when you prefer to use an external multicast router instead of IRB interfaces on Layer 3 EVPN devices to handle inter-VLAN routing.

For example, in a typical EVPN-VXLAN ERB overlay, you can implement use case 1 for intra-VLAN forwarding and use case 2 for inter-VLAN routing and forwarding. Or, if you want an external multicast router to handle inter-VLAN routing in your EVPN-VXLAN network instead of EVPN devices with IRB interfaces running PIM, you can implement use case 5 instead of use case 2. If there are hosts in an existing external PIM domain that you want hosts in your EVPN-VXLAN network to communicate with, you can also implement use case 3.

When implementing any of the use cases in an EVPN-VXLAN CRB overlay, you can use a mix of spine devices—for example, MX Series routers, EX9200 switches, and QFX10000 switches. However, if you do this, keep in mind that the functionality of all spine devices is determined by the limitations of each spine device. For example, QFX10000 switches support a single routing instance of type `virtual-switch`. Although MX Series routers and EX9200 switches support multiple routing instances of type `evpn` or `virtual-switch`, on each of these devices, you would have to configure a single routing instance of type `virtual-switch` to interoperate with the QFX10000 switches.

Use Case 1: Intra-VLAN Multicast Traffic Forwarding

We recommend this basic use case for all EVPN-VXLAN networks.

This use case supports the forwarding of multicast traffic to hosts within the same VLAN and includes the following key features:

- Hosts that are single-homed to an EVPN device or multihomed to more than one EVPN device in all-active mode.



NOTE: EVPN-VXLAN multicast uses special IGMP and MLD group leave processing to handle multihomed sources and receivers, so we don't support the `immediate-leave` configuration option in the `[edit protocols igmp-snooping]` or `[edit protocols mld-snooping]` hierarchies in EVPN-VXLAN networks.

- Routing instances:
 - The default switch instance if all of the devices in the network support EVPN using the default switch instance.
 - If all devices in the network support EVPN using the `virtual-switch` instance type:
 - (QFX Series switches) A single routing instance of type `virtual-switch`.
 - (MX Series routers, vMX virtual routers, and EX9200 switches) Multiple routing instances of type `evpn` or `virtual-switch`.
 - (Preferred) One or more routing instances of type `mac-vrf` if all devices in the network support EVPN using MAC-VRF instances. (See ["MAC-VRF Routing Instance Type Overview" on page 38.](#))
- EVI route target extended community attributes associated with multihomed EVIs.

BGP EVPN Type 7 (Join Sync Route) and Type 8 (Leave Synch Route) routes carry these attributes to enable the simultaneous support of multiple EVPN routing instances. For information about the multicast flags extended community, see ["EVPN Multicast Flags Extended Community" on page 949.](#)
- IGMPv2, IGMPv3, MLDv1 or MLDv2. For information about the membership report modes supported for each IGMP or MLD version, see [Table 50 on page 936.](#) For information about IGMP or MLD route synchronization between multihomed EVPN devices, see ["Overview of Multicast Forwarding with IGMP or MLD Snooping in an EVPN-MPLS Environment" on page 1565.](#)
- IGMP snooping or MLD snooping.

Hosts in a network send IGMP reports (for IPv4 traffic) or MLD reports (for IPv6 traffic) expressing interest in particular multicast groups from multicast sources. EVPN devices with IGMP snooping or MLD snooping enabled listen to the IGMP or MLD reports, and use the snooped information on the access side to establish multicast routes that only forward traffic for a multicast group to interested receivers.

IGMP snooping or MLD snooping supports multicast senders and receivers in the same or different sites. A site can have either receivers only, sources only, or both senders and receivers attached to it.
- Selective multicast forwarding (advertising EVPN Type 6 Selective Multicast Ethernet Tag (SMET) routes for forwarding only to interested receivers).

This feature enables EVPN devices to selectively forward multicast traffic to only the devices in the EVPN core that have expressed interest in that multicast group.



NOTE: We support selective multicast forwarding to devices in the EVPN core only in EVPN-VXLAN CRB overlays.

When you enable IGMP snooping or MLD snooping, selective multicast forwarding is enabled by default.

- EVPN devices that do not support IGMP snooping, MLD snooping, and selective multicast forwarding.

Although you can implement this use case in an EVPN single-homed environment, this use case is particularly effective in an EVPN multihomed environment with a high volume of multicast traffic.

All multihomed interfaces must have the same configuration, and all multihomed peer EVPN devices must be in active mode (not standby or passive mode).

An EVPN device that initially receives traffic from a multicast source is known as the *ingress device*. The ingress device handles the forwarding of intra-VLAN multicast traffic as follows:

- With IGMP snooping or MLD snooping enabled (which also enable selective multicast forwarding on supporting devices):
 - As shown in [Figure 86 on page 941](#), the ingress device (leaf 1) selectively forwards the traffic to other EVPN devices with access interfaces where there are interested receivers for the same multicast group.
 - The traffic is then selectively forwarded to egress devices in the EVPN core that have advertised the EVPN Type 6 SMET routes.
- If any EVPN devices do not support IGMP snooping or MLD snooping, or the ability to originate EVPN Type 6 SMET routes, the ingress device floods multicast traffic to these devices.
- If a host is multihomed to more than one EVPN device, the EVPN devices exchange EVPN Type 7 and Type 8 routes as shown in [Figure 86 on page 941](#). This exchange synchronizes IGMP or MLD membership reports received on multihomed interfaces to coordinate status from messages that go to different EVPN devices or in case one of the EVPN devices fails.

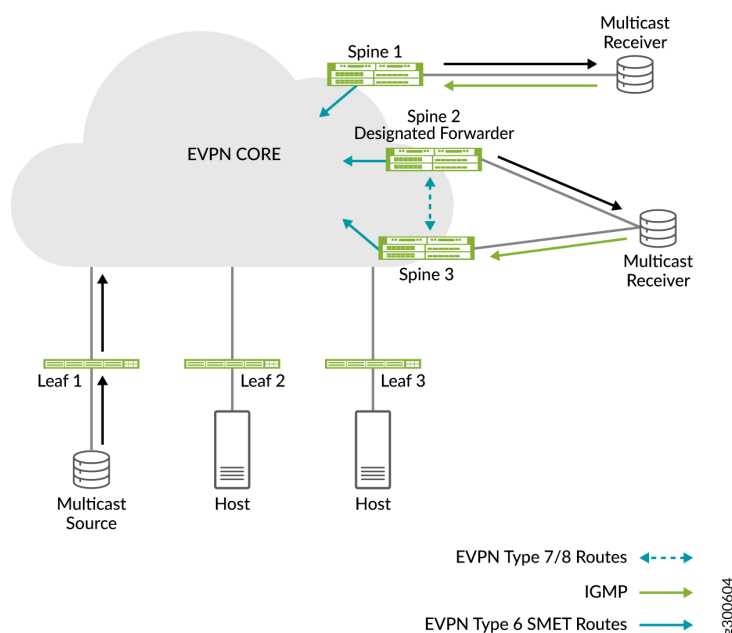


NOTE: The EVPN Type 7 and Type 8 routes carry EVI route extended community attributes to ensure the right EVPN instance gets the IGMP state information on devices with multiple routing instances. QFX Series switches support IGMP snooping only in the default EVPN routing instance (default-switch). In Junos OS releases before 17.4R2, 17.3R3, or 18.1R1, these switches did not include EVI route extended

community attributes in Type 7 and Type 8 routes, so they don't properly synchronize the IGMP state if you also have other routing instances configured. Starting in Junos OS releases 17.4R2, 17.3R3, and 18.1R1, QFX10000 switches include the EVI route extended community attributes that identify the target routing instance, and can synchronize IGMP state if IGMP snooping is enabled in the default EVPN routing instance when other routing instances are configured.

In releases that support MLD and MLD snooping in EVPN-VXLAN fabrics with multihoming, the same behavior applies to synchronizing the MLD state.

Figure 86: Intra-VLAN Multicast Traffic Flow with IGMP Snooping and Selective Multicast Forwarding



If you have configured IRB interfaces with PIM on one or more of the Layer 3 devices in your EVPN-VXLAN network (use case 2), note that the ingress device forwards the multicast traffic to the Layer 3 devices. The ingress device takes this action to register itself with the Layer 3 device that acts as the PIM rendezvous point (RP).

Use Case 2: Inter-VLAN Multicast Routing and Forwarding—IRB Interfaces with PIM

We recommend this basic use case for all EVPN-VXLAN networks except when you prefer to use an external multicast router to handle inter-VLAN routing (see Use Case 5: Inter-VLAN Multicast Routing and Forwarding—External Multicast Router).

For this use case, IRB interfaces using Protocol Independent Multicast (PIM) route multicast traffic between source and receiver VLANs. The EVPN devices on which the IRB interfaces reside then forward the routed traffic using these key features:

- Inclusive multicast forwarding with ingress replication
- IGMP snooping or MLD snooping (if supported)
- Selective multicast forwarding

The default behavior of inclusive multicast forwarding is to replicate multicast traffic and flood the traffic to all devices. For this use case, however, we support inclusive multicast forwarding coupled with IGMP snooping (or MLD snooping) and selective multicast forwarding. As a result, the multicast traffic is replicated but selectively forwarded to access interfaces and devices in the EVPN core that have interested receivers.

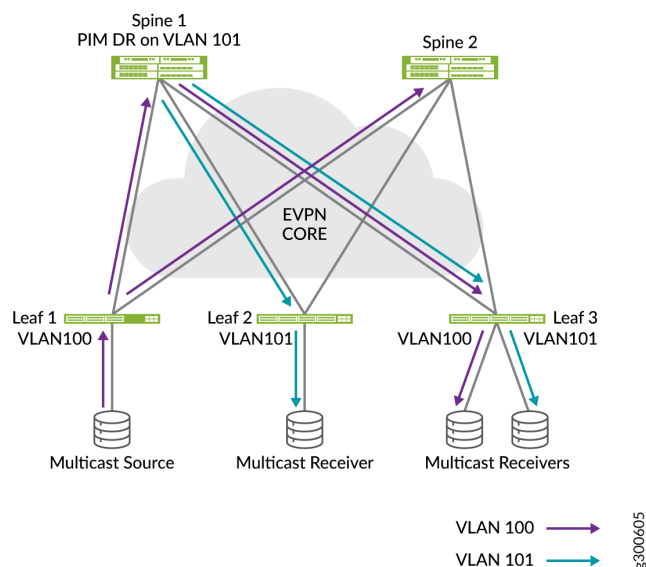
For information about the EVPN multicast flags extended community, which Juniper Networks devices that support EVPN and IGMP snooping (or MLD snooping) include in EVPN Type 3 (Inclusive Multicast Ethernet Tag) routes, see ["EVPN Multicast Flags Extended Community" on page 949](#).

In an EVPN-VXLAN CRB overlay, you can configure the spine devices so that some of them perform inter-VLAN routing and forwarding of multicast traffic and some do not. At a minimum, we recommend that you configure two spine devices to perform inter-VLAN routing and forwarding.

When there are multiple devices that can perform the inter-VLAN routing and forwarding of multicast traffic, one device is elected as the designated router (DR) for each VLAN.

In the sample EVPN-VXLAN CRB overlay shown in [Figure 87 on page 943](#), assume that multicast traffic needs to be routed from source VLAN 100 to receiver VLAN 101. Receiver VLAN 101 is configured on spine 1, which is designated as the DR for that VLAN.

Figure 87: Inter-VLAN Multicast Traffic Flow with IRB Interface and PIM



After the inter-VLAN routing occurs, the EVPN device forwards the routed traffic to:

- Access interfaces that have multicast listeners (IGMP snooping or MLD snooping).
- Egress devices in the EVPN core that have sent EVPN Type 6 SMET routes for the multicast group members in receiver VLAN 2 (selective multicast forwarding).

To understand how IGMP snooping (or MLD snooping) and selective multicast forwarding reduce the impact of the replicating and flooding behavior of inclusive multicast forwarding, assume that an EVPN-VXLAN CRB overlay includes the following elements:

- 100 IRB interfaces using PIM starting with irb.1 and going up to irb.100
- 100 VLANs
- 20 EVPN devices

For the sample EVPN-VXLAN CRB overlay, m represents the number of VLANs, and n represents the number of EVPN devices. Assuming that IGMP snooping (or MLD snooping) and selective multicast forwarding are disabled, when multicast traffic arrives on irb.1, the EVPN device replicates the traffic $m * n$ times or $100 * 20$ times, which equals a rate of 20,000 packets. If the incoming traffic rate for a particular multicast group is 100 packets per second (pps), the EVPN device would have to replicate 200,000 pps for that multicast group.

If IGMP snooping (or MLD snooping) and selective multicast forwarding are enabled in the sample EVPN-VXLAN CRB overlay, assume that there are interested receivers for a particular multicast group on only 4 VLANs and 3 EVPN devices. In this case, the EVPN device replicates the traffic at a rate of

$100 * m * n$ times ($100 * 4 * 3$), which equals 1200 pps. Note the significant reduction in the replication rate and the amount of traffic that must be forwarded.

When implementing this use case, keep in mind that there are important differences for EVPN-VXLAN CRB overlays and EVPN-VXLAN ERB overlays. [Table 52 on page 944](#) outlines these differences

Table 52: Use Case 2: Important Differences for EVPN-VXLAN Edge-routed and Centrally-routed Bridging Overlays

EVPN VXLAN IP Fabric Architectures	Support Mix of Juniper Networks Devices?	All EVPN Devices Required to Host All VLANs In EVPN-VXLAN Network?	All EVPN Devices Required to Host All VLANs that Include Multicast Listeners?	Required PIM Configuration
EVPN-VXLAN ERB overlay	No. We support only QFX10000 switches for all EVPN devices.	Yes	Yes	Configure PIM distributed designated router (DDR) functionality on the IRB interfaces of the EVPN devices.
EVPN-VXLAN CRB overlay	<p>Yes.</p> <p>Spine devices: We support mix of MX Series routers, EX9200 switches, and QFX10000 switches.</p> <p>Leaf devices: We support mix of MX Series routers and QFX5110 switches.</p> <p>NOTE: If you deploy a mix of spine devices, keep in mind that the functionality of all spine devices is determined by the limitations of each spine device.</p>	No	No. However, you must configure all VLANs that include multicast listeners on each spine device that performs inter-VLAN routing. You don't need to configure all VLANs that include multicast listeners on each leaf device.	Do not configure DDR functionality on the IRB interfaces of the spine devices. By not enabling DDR on an IRB interface, PIM remains in a default mode on the interface, which means that the interface acts the designated router for the VLANs.

In addition to the differences described in [Table 52 on page 944](#), a hair pinning issue exists with an EVPN-VXLAN CRB overlay. Multicast traffic typically flows from a source host to a leaf device to a spine

device, which handles the inter-VLAN routing. The spine device then replicates and forwards the traffic to VLANs and EVPN devices with multicast listeners. When forwarding the traffic in this type of EVPN-VXLAN overlay, be aware that the spine device returns the traffic to the leaf device from which the traffic originated (hair-pinning). This issue is inherent with the design of the EVPN-VXLAN CRB overlay. When designing your EVPN-VXLAN overlay, keep this issue in mind especially if you expect the volume of multicast traffic in your overlay to be high and the replication rate of traffic ($m * n$ times) to be large.

Use Case 3: Inter-VLAN Multicast Routing and Forwarding—PIM Gateway with Layer 2 Connectivity

We recommend the PIM gateway with Layer 2 connectivity use case for both EVPN-VXLAN ERB overlays and EVPN-VXLAN CRB overlays.

For this use case, we assume the following:

- You have deployed a EVPN-VXLAN network to support a data center.
- In this network, you have already set up:
 - Intra-VLAN multicast traffic forwarding as described in use case 1.
 - Inter-VLAN multicast traffic routing and forwarding as described in use case 2.
- There are multicast sources and receivers within the data center that you want to communicate with multicast sources and receivers in an external PIM domain.



NOTE: We support this use case with both EVPN-VXLAN ERB overlays and EVPN-VXLAN CRB overlays.

The use case provides a mechanism for the data center, which uses IGMP (or MLD) and PIM, to exchange multicast traffic with the external PIM domain. Using a Layer 2 multicast VLAN (MVLAN) and associated IRB interfaces on the EVPN devices in the data center to connect to the PIM domain, you can enable the forwarding of multicast traffic from:

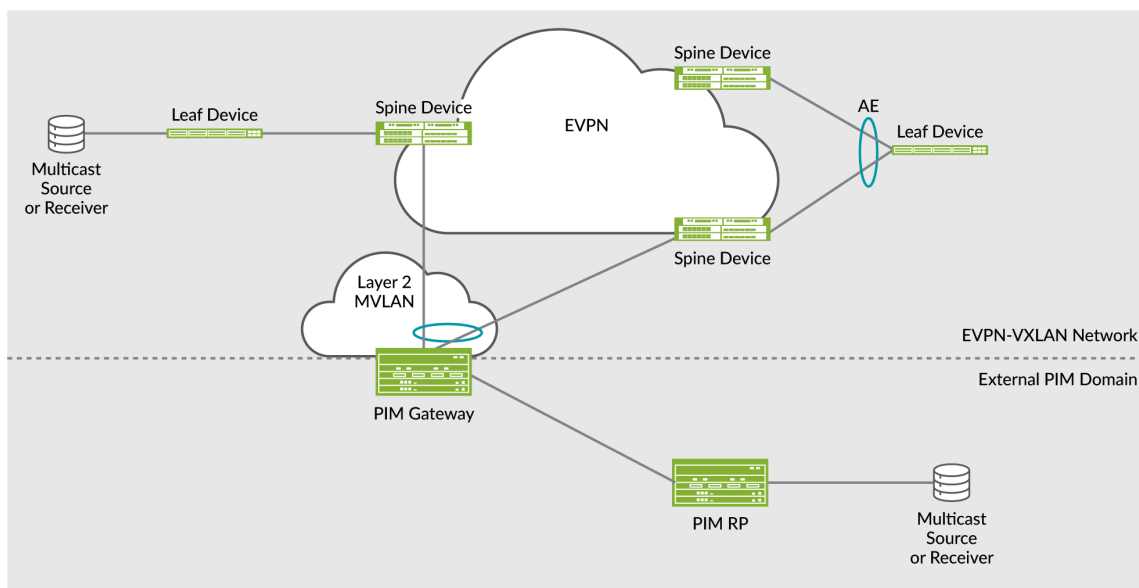
- An external multicast source to internal multicast destinations
- An internal multicast source to external multicast destinations



NOTE: In this section, *external* refers to components in the PIM domain. *Internal* refers to components in your EVPN-VXLAN network that supports a data center.

Figure 88 on page 946 shows the required key components for this use case in a sample EVPN-VXLAN CRB overlay.

Figure 88: Use Case 3: PIM Gateway with Layer 2 Connectivity—Key Components



- Components in the PIM domain:
 - A PIM gateway that acts as an interface between an existing PIM domain and the EVPN-VXLAN network. The PIM gateway is a Juniper Networks or third-party Layer 3 device on which PIM and a routing protocol such as OSPF are configured. The PIM gateway does not run EVPN. You can connect the PIM gateway to one, some, or all EVPN devices.
 - A PIM rendezvous point (RP) is a Juniper Networks or third-party Layer 3 device on which PIM and a routing protocol such as OSPF are configured. You must also configure the PIM RP to translate PIM join or prune messages into corresponding IGMP (or MLD) report or leave messages then forward the reports and messages to the PIM gateway.
- Components in the EVPN-VXLAN network:



NOTE: These components are in addition to the components already configured for use cases 1 and 2.

- EVPN devices. For redundancy, we recommend multihoming the EVPN devices to the PIM gateway through an aggregated Ethernet interface on which you configure an Ethernet segment identifier (ESI). On each EVPN device, you must also configure the following for this use case:
 - A Layer 2 multicast VLAN (MVLAN). The MVLAN is a VLAN that is used to connect the PIM gateway. In the MVLAN, PIM is enabled.

- An MVLAN IRB interface on which you configure PIM, IGMP snooping (or MLD snooping), and a routing protocol such as OSPF. To reach the PIM gateway, the EVPN device forwards multicast traffic out of this interface.
- To enable the EVPN devices to forward multicast traffic to the external PIM domain, configure:
 - PIM-to-IGMP translation:

For EVPN-VXLAN **ERB overlays**, configure PIM-to-IGMP translation by including `pim-to-igmp-proxy upstream-interface irb-interface-name` configuration statements at the `[edit routing-options multicast]` hierarchy level. Specify the MVLAN IRB interface for the IRB interface parameter. You also must set IGMP passive mode using `igmp interface irb-interface-name passive` configuration statements at the `[edit protocols]` hierarchy level on the upstream interfaces where you set `pim-to-igmp-proxy`.

For EVPN-VXLAN **CRB overlays**, you do not need to include the `pim-to-igmp-proxy upstream-interface irb-interface-name` or `pim-to-ml-d-proxy upstream-interface irb-interface-name` configuration statements. In this type of overlay, the PIM protocol handles the routing of multicast traffic from the PIM domain to the EVPN-VXLAN network and vice versa.

- Multicast router interface:

Configure the multicast router interface by including the `multicast-router-interface` configuration statement at the `[edit routing-instances routing-instance-name bridge-domains bridge-domain-name protocols (igmp-snooping | mld-snooping) interface interface-name]` hierarchy level. For the interface name, specify the MVLAN IRB interface.

- PIM passive mode. For EVPN-VXLAN ERB overlays only, you must ensure that the PIM gateway views the data center as only a Layer 2 multicast domain. To do so, include the `passive` configuration statement at the `[edit protocols pim]` hierarchy level.

Use Case 4: Inter-VLAN Multicast Routing and Forwarding—PIM Gateway with Layer 3 Connectivity

We recommend the PIM gateway with Layer 3 connectivity use case for EVPN-VXLAN CRB overlays only.

For this use case, we assume the following:

- You have deployed a EVPN-VXLAN network to support a data center.
- In this network, you have already set up:
 - Intra-VLAN multicast traffic forwarding as described in use case 1.
 - Inter-VLAN multicast traffic routing and forwarding as described in use case 2.

- There are multicast sources and receivers within the data center that you want to communicate with multicast sources and receivers in an external PIM domain.



NOTE: We recommend the PIM gateway with Layer 3 connectivity use case for EVPN-VXLAN CRB overlays only.

This use case provides a mechanism for the data center, which uses IGMP (or MLD) and PIM, to exchange multicast traffic with the external PIM domain. Using Layer 3 interfaces on the EVPN devices in the data center to connect to the PIM domain, you can enable the forwarding of multicast traffic from:

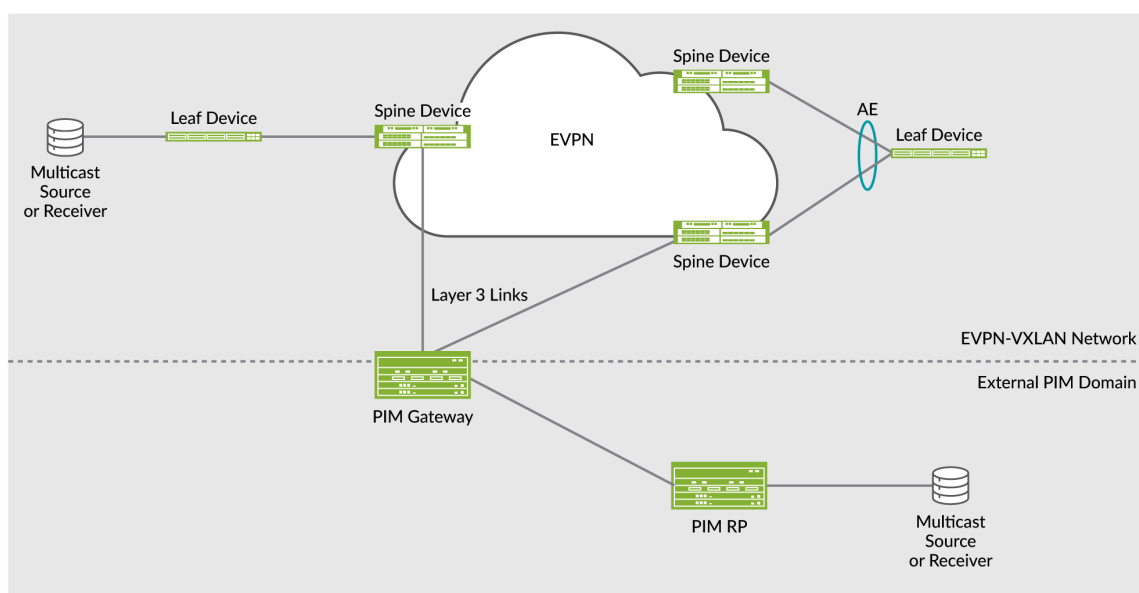
- An external multicast source to internal multicast destinations
- An internal multicast source to external multicast destinations



NOTE: In this section, *external* refers to components in the PIM domains. *Internal* refers to components in your EVPN-VXLAN network that supports a data center.

Figure 89 on page 948 shows the required key components for this use case in a sample EVPN-VXLAN CRB overlay.

Figure 89: Use Case 4: PIM Gateway with Layer 3 Connectivity—Key Components



- Components in the PIM domain:

- A PIM gateway that acts as an interface between an existing PIM domain and the EVPN-VXLAN network. The PIM gateway is a Juniper Networks or third-party Layer 3 device on which PIM and a routing protocol such as OSPF are configured. The PIM gateway does not run EVPN. You can connect the PIM gateway to one, some, or all EVPN devices.
- A PIM rendezvous point (RP) is a Juniper Networks or third-party Layer 3 device on which PIM and a routing protocol such as OSPF are configured. You must also configure the PIM RP to translate PIM join or prune messages into corresponding IGMP or MLD report or leave messages then forward the reports and messages to the PIM gateway.
- Components in the EVPN-VXLAN network:



NOTE: These components are in addition to the components already configured for use cases 1 and 2.

- EVPN devices. You can connect one, some, or all EVPN devices to a PIM gateway. You must make each connection through a Layer 3 interface on which PIM is configured. Other than the Layer 3 interface with PIM, this use case does not require additional configuration on the EVPN devices.

Use Case 5: Inter-VLAN Multicast Routing and Forwarding—External Multicast Router

Starting with Junos OS Release 17.3R1, you can configure an EVPN device to perform inter-VLAN forwarding of multicast traffic without having to configure IRB interfaces on the EVPN device. In such a scenario, an external multicast router is used to send IGMP or MLD queries to solicit reports and to forward VLAN traffic through a Layer 3 multicast protocol such as PIM. IRB interfaces are not supported with the use of an external multicast router.

For this use case, you must include the `igmp-snooping proxy` or `mld-snooping proxy` configuration statements at the `[edit routing-instances routing-instance-name protocols vlan vlan-name]` hierarchy level.

EVPN Multicast Flags Extended Community

Juniper Networks devices that support EVPN-VXLAN and IGMP snooping also support the EVPN multicast flags extended community. When you have enabled IGMP snooping on one of these devices, the device adds the community to EVPN Type 3 (Inclusive Multicast Ethernet Tag) routes.

The absence of this community in an EVPN Type 3 route can indicate the following about the device that advertises the route:

- The device does not support IGMP snooping.
- The device does not have IGMP snooping enabled on it.
- The device is running a Junos OS software release that doesn't support the community.

- The device does not support the advertising of EVPN Type 6 SMET routes.
- The device has IGMP snooping and a Layer 3 interface with PIM enabled on it. Although the Layer 3 interface with PIM performs snooping on the access side and selective multicast forwarding on the EVPN core, the device needs to attract all traffic to perform source registration to the PIM RP and inter-VLAN routing.

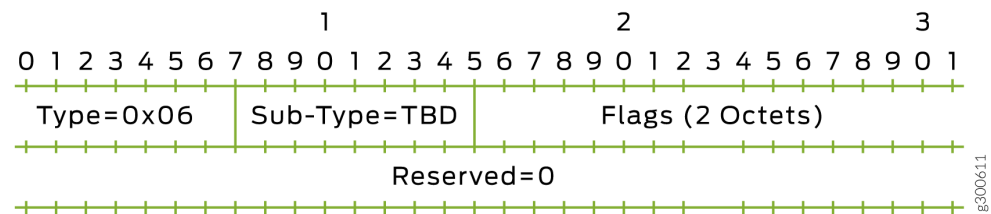
The behavior described above also applies to devices that support EVPN-VXLAN with MLD and MLD snooping.

Figure 90 on page 950 shows the EVPN multicast flag extended community, which has the following characteristics:

- The community is encoded as an 8-bit value.
- The Type field has a value of 6.
- The IGMP Proxy Support flag is set to 1, which means that the device supports IGMP proxy.

The same applies to the MLD Proxy Support flag; if that flag is set to 1, the device supports MLD proxy. Either or both flags might be set.

Figure 90: EVPN Multicast Flag Extended Community



Platform-Specific EVPN-VXLAN Multicast Behavior with IGMP Snooping and MLD Snooping

Use the following table to review platform-specific behaviors for your platforms when running EVPN-VXLAN with IGMP snooping and MLD snooping.

Table 53: Platform-Specific Behavior

Platform	Difference
PTX10008 with the PTX10K-LC1301-36DD (Express 5) and PTX10K-LC1201-36CD (Express 4) line cards	<p>The device has a limitation in reporting multicast packet statistics when the device operates in interop chassis mode (both Express 5 and Express 4 line cards interoperating on the device), because Express 5 line cards don't support multicast snooping route counters.</p> <p>The multicast snooping route counters record the number of packets that use a snooping route toward the destination. Express 4 line cards update the snooping route counters, but Express 5 line cards don't update those counters. As a result, on devices running in interop mode, the multicast packet statistics reported in the <code>show multicast snooping route extensive</code> CLI command won't match the actual packet count for snooping routes.</p>

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
22.3R1	Starting with Junos OS Release 22.3R1, we support IGMP snooping on EX4400 switches in EVPN-VXLAN networks.
22.3R1	Starting with Junos release 22.3R1, on some platforms we support IPv4 and IPv6 multicast traffic in an IPv6 EVPN-VXLAN overlay with IPv6 underlay peering.
21.2R1	Starting with Junos OS Release 21.2R1, we support optimized inter-subnet multicast (OISM) routing and forwarding with IGMP snooping on QFX5110, QFX5120, and QFX10002 switches in EVPN-VXLAN ERB overlay networks.
20.4R1	Starting with Junos OS Release 20.4R1, QFX5120-48YM switches support IGMP snooping as leaf devices in a multihomed EVPN-VXLAN CRB overlay fabric.

20.4R1	Starting in Junos OS Release 20.4R1, in EVPN-VXLAN CRB overlay fabrics, QFX5110, QFX5120 and the QFX10000 line of switches support IGMPv3 with IGMP snooping for IPv4 multicast traffic, and MLDv1 and MLDv2 with MLD snooping for IPv6 multicast traffic. Supported switches in the QFX5000 line only support multicast forwarding as leaf devices in CRB fabrics, and don't support multicast routing. The spine devices handle Layer 3 multicast routing. You can configure these switches to process IGMPv3 and MLDv2 source-specific multicast (SSM) reports, but these devices can't process both SSM reports and any-source multicast (ASM) reports at the same time.
20.2R1	Starting with Junos OS Release 20.2R1, QFX5120-48T switches support IGMP snooping as leaf devices in a multihomed EVPN-VXLAN CRB overlay fabric.
19.3R1	Starting with Junos OS Release 19.3R1, EX9200 switches, MX Series routers, and vMX virtual routers support IGMPv2, IGMPv3, IGMP snooping, selective multicast forwarding, external PIM gateways, and external multicast routers with an EVPN-VXLAN CRB overlay.
19.1R1	Starting with Junos OS Release 19.1R1, QFX5120-32C switches support IGMP snooping as leaf devices in an EVPN-VXLAN CRB overlay fabric.
18.4R2	Starting with Junos OS Release 18.4R2 (but not Junos OS Releases 19.1R1 and 19.2R1), QFX5120-48Y switches support IGMP snooping as leaf devices in an EVPN-VXLAN CRB overlay fabric.
18.4R2	Starting in Junos OS Releases 18.4R2 and 19.1R2, selective multicast forwarding is enabled by default on QFX5110 and QFX5120 switches when you configure IGMP snooping in EVPN-VXLAN networks, further constraining multicast traffic flooding. With IGMP snooping and selective multicast forwarding, these switches send the multicast traffic only to interested receivers in both the EVPN core and on the access side for multicast traffic coming in either from an access interface or an EVPN network interface.
18.1R1	Starting with Junos OS Release 18.1R1, QFX5110 switches support IGMP snooping as leaf devices in an EVPN-VXLAN CRB overlay (EVPN-VXLAN topology with a two-layer IP fabric) for forwarding multicast traffic within VLANs. You can't configure IRB interfaces on a VXLAN with IGMP snooping for forwarding multicast traffic between VLANs. You can only configure and use IRB interfaces for unicast traffic.
17.3R1	Starting with Junos OS Release 17.3R1, QFX10000 switches support exchanging traffic between multicast sources and receivers in an EVPN-VXLAN ERB overlay using IGMP. These switches also support multicast traffic flow from sources and to receivers in an external Protocol Independent Multicast (PIM) domain. A Layer 2 multicast VLAN (M-VLAN) and associated IRB interfaces enable the exchange of multicast traffic between these two domains.
17.3R1	Starting with Junos OS Release 17.3R1, you can configure an EVPN device to perform multicast traffic inter-VLAN forwarding without needing to configure IRB interfaces on the EVPN device.
17.2R1	Starting with Junos OS Release 17.2R1, QFX10000 switches support IGMP snooping in an EVPN-VXLAN ERB overlay.

RELATED DOCUMENTATION

[distributed-dr](#)[igmp-snooping](#)[mld-snooping](#)[multicast-router-interface](#)[Example: Preserving Bandwidth with IGMP Snooping in an EVPN-VXLAN Environment | 953](#)

Example: Preserving Bandwidth with IGMP Snooping in an EVPN-VXLAN Environment

IN THIS SECTION

- [Requirements | 953](#)
- [Overview | 954](#)
- [Configuration | 955](#)
- [Verification | 997](#)

This example shows how to configure IGMP snooping on provider edge (PE) devices in an Ethernet VPN (EVPN)-Virtual Extensible LAN. When multicast traffic arrives at the VXLAN core, a PE device configured with EVPN forwards traffic only to the local access interfaces where there are IGMP listeners.

Requirements

This example uses the following hardware and software components:

- Two QFX10000 switches configured as multihomed PE devices that are connected to the CE, one QFX10000 device configured as a PE device connected to the multihomed PEs and a QFX5110 configured as a CE device.
- Junos OS Release 17.2R1 or later running on all devices.

Overview

IN THIS SECTION

- [Topology](#) | 954

IGMP snooping is used to constrain multicast traffic in a broadcast domain to interested receivers and multicast devices. In an environment with significant multicast traffic, IGMP snooping preserves bandwidth because multicast traffic is forwarded only on those interfaces where there are IGMP listeners. IGMP is enabled to manage multicast group membership.

When you enable IGMP snooping on each VLAN, the device advertises EVPN Type 7 and Type 8 routes (Join and Leave Sync Routes) to synchronize IGMP join and leave states among multihoming peer devices in the EVPN instance. On the access side, devices only forward multicast traffic to subscribed listeners. However, multicast traffic is still flooded in the EVPN core even when there are no remote receivers.

Configuring IGMP snooping in an EVPN-VXLAN environment requires the following:

- Multihoming peer PE devices in all-active mode.
- IGMP version 2 only. (IGMP versions 1 and 3 are not supported.)
- IGMP snooping configured in proxy mode for the PE to become the IGMP querier for the local access interfaces.

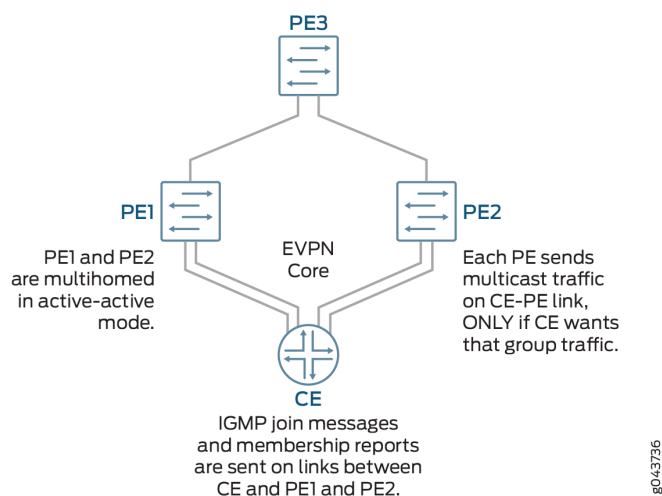
This feature supports both intra-VLAN and inter-VLAN multicast forwarding. You can configure a PE device to perform either or both. In this example, to enable inter-VLAN forwarding, each PE device is configured as a statically defined Protocol Independent Multicast (PIM) rendezvous point (RP) to enable multicast forwarding. You also configure the `distributed-dr` statement at the `[edit protocols pim interface interface-name]` hierarchy level for each IRB interface. This mode enables PIM to forward multicast traffic more efficiently by disabling PIM features that are not required in this scenario. When you configure this statement, PIM ignores the designated router (DR) status of the interface when processing IGMP reports received on the interface. When the interface receives the IGMP report, the PE device sends PIM upstream join messages to pull the multicast stream and forward it to the interface—regardless of the DR status of the interface.

Topology

[Figure 91 on page 955](#) illustrates an EVPN-VXLAN environment where two PE devices (PE1 and PE2) are connected to the customer edge (CE) device. These PEs are dual-homed in active-active mode to provide redundancy. A third PE device forwards traffic to the PE devices that face the CE. IGMP is

enabled on integrated routing and bridging (IRB) interfaces. The CE device hosts five VLANs; IGMP snooping is enabled on all of the VLANs. Because this implementation does not support the use of a multicast router, each VLAN in the PE is enabled as an IGMP Layer 2 querier. The multihomed PE devices forward traffic towards the CE only on those interfaces where there are IGMP listeners.

Figure 91: IGMP Snooping in an EVPN-VXLAN Environment



Configuration

IN THIS SECTION

- [CLI Quick Configuration | 956](#)
- [Configuring PE1 | 962](#)
- [Configuring PE2 | 973](#)
- [Configuring CE Device | 984](#)
- [Configuring PE3 | 988](#)

To configure IGMP Snooping in an EVPN-VXLAN environment, perform these tasks:

CLI Quick Configuration

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

Device PE1

```
set chassis aggregated-devices ethernet device-count 2
set interfaces xe-0/0/0:1 description "Connected to CE"
set interfaces xe-0/0/0:2 ether-options 802.3ad ae1
set interfaces xe-0/0/1:0 enable
set interfaces xe-0/0/1:0 unit 0 description "Connected to PE3"
set interfaces xe-0/0/1:0 unit 0 family inet address 192.0.2.1/24
set interfaces ae0 enable
set interfaces ae1 enable
set interfaces lo0 unit 0 family inet address 192.168.1.1/32
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:01
set interfaces ae0 unit 0 description "Connected to CE"
set interfaces ae1 esi 00:22:22:22:22:22:22:22:22:22
set interfaces ae1 esi all-active
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options periodic fast
set interfaces ae1 aggregated-ether-options lacp system id 00:00:00:00:00:02
set interfaces ae1 unit 0 description "Connected to CE"
set interfaces ae0 family ethernet-switching interface-mode trunk
set interfaces ae0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ]
set interfaces ae1 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 0 family ethernet-switching vlan [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ]
set interfaces irb unit 1 family inet address 10.1.1.1/24 virtual-gateway-address 10.1.1.10
set interfaces irb unit 2 family inet address 10.2.1.1/24 virtual-gateway-address 10.2.1.10
set interfaces irb unit 3 family inet address 10.3.1.1/24 virtual-gateway-address 10.3.1.10
set interfaces irb unit 4 family inet address 10.4.1.1/24 virtual-gateway-address 10.4.1.10
set interfaces irb unit 5 family inet address 10.5.1.1/24 virtual-gateway-address 10.5.1.10
set routing-options router-id 192.168.1.1
set routing-options autonomous-system 65536
set protocols ospf area 0.0.0.0 interface xe-0/0/1:0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

```

set protocols bgp group INT type internal
set protocols bgp group INT local-address 192.168.1.1
set protocols bgp group INT family evpn signaling
set protocols bgp group INT local-as 65536
set protocols bgp group INT neighbor 172.16.1.1
set vlans VLAN vlan-id 1
set vlans VLAN1 l3-interface irb.1
set vlans VLAN1 vxlan vni 1
set vlans VLAN2 vlan-id 2
set vlans VLAN2 l3-interface irb.2
set vlans VLAN2 vxlan vni2
set vlans VLAN3 vlan-id 3
set vlans VLAN3 l3-interface irb.3
set vlans VLAN3 vxlan vni 3
set vlans VLAN4 vlan-id 4
set vlans VLAN4 l3-interface irb.4
set vlans VLAN4 vxlan vni 4
set vlans VLAN5 vlan-id 5
set vlans VLAN5 l3-interface irb.5
set vlans VLAN5 vxlan vni 5
set protocols evpn encapsulation vxlan
set protocols evpn ingress-replication
set protocols evpn extended-vni-list 1-5
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set routing-options forwarding-table export evpn-pplb
set switch-options vtep-source-interface lo0:0
set switch-options route distinguisher 192.168.1.1:1
set switch-options vrf-target target:1:1
set protocols igmp interface irb.1
set protocols igmp interface irb.2
set protocols igmp interface irb.3
set protocols igmp interface irb.4
set protocols igmp interface irb.5
set protocols igmp-snooping vlan VLAN1 l2-querier source-address 10.1.1.1
set protocols igmp-snooping vlan VLAN1 proxy
set protocols igmp-snooping vlan VLAN2 l2-querier source-address 10.2.1.1
set protocols igmp-snooping vlan VLAN2 proxy
set protocols igmp-snooping vlan VLAN3 l2-querier source-address 10.3.1.1
set protocols igmp-snooping vlan VLAN3 proxy
set protocols igmp-snooping vlan VLAN4 l2-querier source-address 10.4.1.1
set protocols igmp-snooping vlan VLAN4 proxy
set protocols igmp-snooping vlan VLAN4 l2-querier source-address 10.5.1.1

```

```

set protocols igmp-snooping vlan VLAN5 proxy
set protocols pim rp static address 172.16.1.1
set protocols pim interface irb.1 distributed-dr
set protocols pim interface irb.2 distributed-dr
set protocols pim interface irb.3 distributed-dr
set protocols pim interface irb.4 distributed-dr
set protocols pim interface irb.5 distributed-dr

```

Device PE2

```

set chassis aggregated-devices ethernet device-count 2
set interfaces xe-0/0/0:0 ether-options 802.3ad ae1
set interfaces xe-0/0/0:1 description "Connected to CE"
set interfaces xe-0/0/0:1 ether-options 802.3ad ae0
set interfaces xe-0/0/1:0 enable
set interfaces xe-0/0/1:0 unit 0 description "Connected to PE3"
set interfaces xe-0/0/1:0 family inet address 198.51.100.1/24
set interfaces ae0 enable
set interfaces ae1 enable
set interfaces lo0 unit 0 family inet address 192.168.2.1/32
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated ether-options lacp periodic fast
set interfaces ae0 aggregated ether-options lacp system-id 00:00:00:00:00:01
set interfaces ae1 esi 00:22:22:22:22:22:22:22:22
set interfaces ae1 esi all-active
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 aggregated-ether-options lacp system-id 00:00:00:00:00:02
set interfaces ae0 unit 0 description "Connected to CE"
set interfaces ae0 unit family ethernet-switching interface-mode trunk
set interfaces ae0 unit family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ]
set interfaces ae1 unit 0 description "Connected to CE"
set interfaces ae1 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3 VLAN4
VLAN5 ]
set interfaces irb unit 1 family inet address 10.1.1.2/24 virtual-gateway-address 10.1.1.10
set interfaces irb unit 2 family inet address 10.2.1.2/24 virtual-gateway-address 10.2.1.10
set interfaces irb unit 3 family inet address 10.3.1.2/24 virtual-gateway-address 10.3.1.10
set interfaces irb unit 4 family inet address 10.4.1.2/24 virtual-gateway-address 10.4.1.10
set interfaces irb unit 5 family inet address 10.5.1.2/24 virtual-gateway-address 10.5.1.10

```

```

set routing-options router-id 192.168.2.1
set routing-options autonomous-system 65536
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface xe-0/0/1:0.0
set protocols bgp group INT type internal
set protocols bgp group INT local-address 192.168.2.1
set protocols bgp group INT family evpn signaling
set protocols bgp group INT local-as 65536
set protocols bgp group INT neighbor 172.16.1.1
set vlans VLAN1 vlan-id 1
set vlans VLAN1 l3-interface irb.1
set vlans VLAN1 vxlan vni 1
set vlans VLAN2 vlan-id 2
set vlans VLAN2 l3-interface irb.2
set vlans VLAN2 vxlan vni 2
set vlans VLAN3 vlan-id 3
set vlans VLAN3 l3-interface irb.3
set vlans VLAN3 vxlan vni 3
set vlans VLAN4 vlan-id 4
set vlans VLAN4 l3-interface irb.4
set vlans VLAN4 vxlan vni 4
set vlans VLAN5 vlan-id 5
set vlans VLAN5 l3-interface irb.5
set vlans VLAN5 vxlan vni 3
set protocols evpn encapsulation vxlan
set protocols evpn muticast-mode ingress-encapsulation
set protocols evpn extended-vni-list 1-5
set policy-options policy statement evpn-pplb from protocol evpn
set policy-options policy statement evpn-pplb then load-balance per-packet
set routing-options forwarding-table export evpn-pplb
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 192.168.2.1:1
set switch-options vrf-target target:1:1
set protocols igmp interface irb.1
set protocols igmp interface irb.2
set protocols igmp interface irb.3
set protocols igmp interface irb.4
set protocols igmp interface irb.5
set protocols igmp-snooping vlan VLAN1 l2-querier source-address 10.1.1.2
set protocols igmp-snooping vlan VLAN1 proxy
set protocols igmp-snooping vlan VLAN2 l2-querier source-address 10.2.1.2
set protocols igmp-snooping vlan VLAN2 proxy
set protocols igmp-snooping vlan VLAN3 l2-querier source-address 10.3.1.2

```



```

set protocols igmp-snooping vlan VLAN3 proxy
set protocols igmp-snooping vlan VLAN4 l2-querier source-address 10.4.1.2
set protocols igmp-snooping vlan VLAN4 proxy
set protocols igmp-snooping vlan VLAN5 l2-querier source-address 10.5.1.2
set protocols igmp-snooping vlan VLAN5 proxy
set protocols pim rp static address 172.16.1.1
set protocols pim interface irb.1 distributed-dr
set protocols pim interface irb.2 distributed-dr
set protocols pim interface irb.3 distributed-dr
set protocols pim interface irb.4 distributed-dr
set protocols pim interface irb.5 distributed-dr

```

CE

```

set chassis aggregated devices ethernet device-count 2
set interfaces xe-0/2/0 ether-options 802.3ad ae0
set interfaces xe-0/2/1 ether-options 802.3ad ae0
set interfaces xe-0/2/0 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/2/0 unit 0 family ethernet-switching vlan members 1-5
set interfaces xe-0/2/1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/2/1 unit 0 family ethernet-switching vlan members 1-5
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members 1-5
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 0 family ethernet-switching vlan members 1-5
set vlans BD-1 vlan-id 1
set vlans BD-2 vlan-id 2
set vlans BD-3 vlan-id 3
set vlans BD-4 vlan-id 4
set vlans BD-5 vlan-id 5

```

PE3

```

set interfaces xe-0/0/0 enable
set interfaces xe-/0/0/0 unit 0 description "Connected to PE1"

```

```

set interfaces xe-0/0/0 unit 0 family inet 192.0.2.2/24
set interfaces xe-0/0/1 enable
set interfaces xe-0/0/1 unit 0 description "Connected to PE2"
set interfaces xe-0/0/1 unit 0 family inet 198.51.100.2/24
set interfaces lo0 unit 0 family inet address 172.16.1.1/32
set interfaces xe-0/0/0:1 enable
set interfaces xe-0/0/0:1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/0:1 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3
VLAN4 VLAN5 ]
set interfaces xe-0/0/1:1 enable
set interfaces xe-0/0/1:1 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/1:1 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3
VLAN4 VLAN5 ]
set interfaces irb unit 1 family inet address 10.1.1.5/24 virtual-gateway address 10.1.1.10
set interfaces irb unit 2 family inet address 10.2.1.5/24 virtual-gateway address 10.2.1.10
set interfaces irb unit 3 family inet address 10.3.1.5/24 virtual-gateway address 10.3.1.10
set interfaces irb unit 4 family inet address 10.4.1.5/24 virtual-gateway address 10.4.1.10
set interfaces irb unit 5 family inet address 10.5.1.5/24 virtual-gateway address 10.5.1.10
set routing-options router-id 172.16.1.1
set routing-options autonomous-system 65536
set ospf area 0.0.0.0 interface all
set ospf area 0.0.0.0 fxp0.0 disable
set protocols bgp group INT type internal
set protocols bgp group INT local-address 172.16.1.1
set protocols bgp group INT family evpn signaling
set protocols bgp group INT local-as 65536
set protocols bgp group INT neighbor 192.168.1.1
set protocols bgp group INT neighbor 192.168.2.1
set vlans VLAN1 vlan-id 1
set vlans VLAN1 l3-interface irb.1
set vlans VLAN1 vxlan vni 1
set vlans VLAN2 vlan-id 2
set vlans VLAN2 l3-interface irb.2
set vlans VLAN2 vxlan vni 2
set vlans VLAN3 vlan-id 3
set vlans VLAN3 l3-interface irb.3
set vlans VLAN3 vxlan vni 3
set vlans VLAN4 vlan-id 4
set vlans VLAN4 l3-interface irb.4
set vlans VLAN4 vxlan vni 4
set vlans VLAN5 vlan-id 5
set vlans VLAN5 l3-interface irb.5
set vlans VLAN5 vxlan vni 5

```

```

set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set protocols evpn extended-vni-list 1-5
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pbllb then load-balance per packet
set routing-options forwarding-table export evpn-pplb
set switch-options vtep-source-interface lo0
set switch-options route-distinguisher 172.16.1.1:1
set switch-option vrf-target target:1:1
set protocols igmp interface irb.1
set protocols igmp interface irb.2
set protocols igmp interface irb.3
set protocols igmp interface irb.4
set protocols igmp interface irb.5
set protocols igmp-snooping vlan VLAN1 l2-querier source-address 10.1.1.5
set protocols igmp-snooping vlan VLAN1 proxy
set protocols igmp-snooping vlan VLAN2 l2-querier source-address 10.2.1.5
set protocols igmp-snooping vlan VLAN2 proxy
set protocols igmp-snooping vlan VLAN3 l2-querier source-address 10.3.1.5
set protocols igmp-snooping vlan VLAN3 proxy
set protocols igmp-snooping vlan VLAN4 l2-querier source-address 10.4.1.5
set protocols igmp-snooping vlan VLAN4 proxy
set protocols igmp-snooping vlan VLAN5 l2-querier source-address 10.5.1.5
set protocols igmp-snooping vlan VLAN5 proxy
set protocols pim rp local 172.16.1.1
set protocols pim interface irb.1 distributed-dr
set protocols pim interface irb.2 distributed-dr
set protocols pim interface irb.3 distributed-dr
set protocols pim interface irb.4 distributed-dr
set protocols pim interface irb.5 distributed-dr

```

Configuring 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. Specify the number of aggregated Ethernet logical interfaces.

```
[edit chassis]
user@PE1# set aggregated-devices ethernet device-count 2
```

2. Configure the interfaces.

```
[edit interfaces]
user@PE1# set xe-0/0/0:1 description "Connected to CE"
user@PE1# set xe-0/0/0:2 ether-options 802.3ad ae1
user@PE1# set xe-0/0/1:0 enable
user@PE1# set xe-0/0/1:0 unit 0 description "Connected to PE3"
user@PE1# set xe-0/0/1:0 unit 0 family inet address 192.0.2.1/24
user@PE1# set ae0 enable
user@PE1# set ae1 enable
user@PE2# set lo0 unit 0 family inet address 192.168.1.1/32
```

3. Configure active-active multihoming and enable the Link Aggregation Control Protocol (LACP) on each aggregated Ethernet interface.

```
[edit interfaces]
user@PE1# set ae0 esi 00:11:11:11:11:11:11:11:11
user@PE1# set esi all-active
user@PE1# set ae0 aggregated-ether-options lacp active
user@PE1# set ae0 aggregated-ether-options lacp periodic-fast
user@PE1# set ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:01
user@PE1# set ae1 esi 00:22:22:22:22:22:22:22:22
user@PE1# set ae1 esi all-active
user@PE1# set ae1 aggregated-ether-options lacp active
user@PE1# set ae1 aggregated-ether-options lacp periodic-fast
user@PE1# set ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:02
```

4. Configure each aggregated Ethernet interface as a trunk port.

```
[edit interfaces]
user@PE1# set ae0 unit 0 description "Connected to CE"
```

```

user@PE1# set ae0 unit 0 family ethernet-switching interface-mode trunk
user@PE1# set ae0 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3 VLAN4
VLAN5 ]
user@PE1# set ae1 unit 0 description "Connected to CE"
user@PE1# set ae1 unit 0 family ethernet-switching interface-mode trunk
user@PE1# set ae1 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3 VLAN4
VLAN5 ]

```

5. Configure IRB interfaces and virtual-gateway addresses.

```

[edit interfaces]
user@PE1# set irb unit 1 family inet address 10.1.1.1/24 virtual-gateway-address 10.1.1.10
user@PE1# set irb unit 2 family inet address 10.2.1.1/24 virtual-gateway-address 10.2.1.10
user@PE1# set irb unit 3 family inet address 10.3.1.1/24 virtual-gateway-address 10.3.1.10
user@PE1# set irb unit 4 family inet address 10.4.1.1/24 virtual-gateway-address 10.4.1.10
user@PE1# set irb unit 5 family inet address 10.5.1.1/24 virtual-gateway-address 10.5.1.10

```

6. Configure the autonomous system.

```

[edit routing-options]
user@PE1# set router-id 192.168.1.1
user@PE1# set autonomous-system 65536

```

7. Configure OSPF.

```

[edit protocols ospf]
user@PE1# set area 0.0.0.0 interface xe-0/0/1:0.0
user@PE1# set area 0.0.0.0 interface lo0. passive

```

8. Configure BGP internal peering.

```

[edit protocols bgp]
user@PE1# set group INT type internal
user@PE1# set group INT local-address 192.168.1.1
user@PE1# set group INT family evpn signaling
user@PE1# set group INT local-as 65536
user@PE1# set group INT neighbor 172.16.1.1

```

9. Configure the VLANs.

```
[set vlans]
user@PE1# set VLAN1 vlan-id 1
user@PE1# set VLAN1 l3-interface irb.1
user@PE1# set VLAN1 vxlan vni 1
user@PE1# set VLAN2 vlan-id 2
user@PE1# set VLAN2 l3-interface irb.2
user@PE1# set VLAN2 vxlan vni 2
user@PE1# set VLAN3 vlan-id 3
user@PE1# set VLAN3 l3-interface irb.3
user@PE1# set VLAN3 vxlan vni 3
user@PE1# set VLAN4 vlan-id 4
user@PE1# set VLAN4 l3-interface irb.4
user@PE1# set VLAN4 vxlan vni 4
user@PE1# set VLAN5 vlan-id 5
user@PE1# set VLAN5 l3-interface irb.5
user@PE1# set VLAN2 vxlan vni 5
```

10. Enable EVPN.

```
[edit protocols evpn]
user@PE1# set encapsulation vxlan
user@PE1# set multicast-mode ingress-replication
user@PE1# set extended-vni-list 1-5
```

11. Configure an export routing policy to load balance EVPN traffic.

```
[edit policy-options]
user@PE1# set policy-statement evpn-pplb from protocol evpn
user@PE1# set policy-statement evpn-pplb then load-balance per packet

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

12. Configure the source interface for the VXLAN tunnel.

```
[edit switch-options]
user@PE1# set vtep-source-interface lo0.0
```

```

user@PE1# set route-distinguisher 192.168.1.1:1
user@PE1# set vrf-target target:1:1

```

13. Enable IGMP on the IRB interfaces associated with the VLANs.

```

[edit protocols igmp]
user@PE1# set interface irb.1
user@PE1# set interface irb.2
user@PE1# set interface irb.3
user@PE1# set interface irb.4
user@PE1# set interface irb.5

```

14. Enable IGMP snooping on the VLANs.

```

[edit protocols igmp-snooping vlan]
user@PE1# set VLAN1 l2-querier source-address 10.1.1.1
user@PE1# set VLAN1 proxy
user@PE1# set VLAN2 l2-querier source-address 10.2.1.1
user@PE1# set VLAN2 proxy
user@PE1# set VLAN3 l2-querier source address 10.3.1.1
user@PE1# set VLAN3 proxy
user@PE1# set VLAN4 l2-querier source-address 10.4.1.1
user@PE1# set VLAN4 proxy
user@PE1# set VLAN5 l2-querier source-address 10.5.1.1
user@PE1# set VLAN5 proxy

```

15. Configure PIM by defining a static rendezvous point and enabling on the IRB interfaces associated with the VLANs..



NOTE: This step is required only if you want to configure inter-VLAN forwarding. If your PE device is performing only intra-VLAN forwarding, omit this step.

```

[edit protocols pim]
user@PE1# set rp static address 172.16.1.1
user@PE1# set interface irb.1 distributed-dr
user@PE1# set interface irb.2 distributed-dr
user@PE1# set interface irb.3 distributed-dr

```

```

user@PE1# set interface irb.4 distributed-dr
user@PE1# set interface irb.5 distributed-dr

```

Results

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

```

user@PE1# show chassis
aggregated-devices {
  ethernet {
    device-count 2;
  }
}

```

```

user@PE1# show
interfaces {
  xe-0/0/0:1 {
    description "Connected to CE";
    ether-options {
      802.3ad ae0;
    }
  }
  xe-0/0/0:2 {
    ether-options {
      802.3ad ae1;
    }
  }
  xe-0/0/1:0 {
    enable;
    unit 0 {
      description "Connected to PE3";
      family inet {
        address 192.0.2.1/24;
      }
    }
  }
  xe-0/0/1:1 {

```



```

    enable;
}
ae0 {
    enable;
    esi {
        00:11:11:11:11:11:11:11:11:11;
        all-active;
    }
    aggregated-ether-options {
        lacp {
            active;
            periodic fast;
            system-id 00:00:00:00:00:01;
        }
    }
    unit 0 {
        description "Connected to CE";
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ];
            }
        }
    }
}
ae1 {
    enable;
    esi {
        00:22:22:22:22:22:22:22:22:22;
        all-active;
    }
    aggregated-ether-options {
        lacp {
            active;
            periodic fast;
            system-id 00:00:00:00:00:02;
        }
    }
    unit 0 {
        description "Connected to CE";
        family ethernet-switching {
            interface-mode trunk;
            vlan {

```

```

        members [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ];
    }
}

irb {
    unit 1 {
        family inet {
            address 10.1.1.1/24 {
                virtual-gateway-address 10.1.1.10;
            }
        }
    }
    unit 2 {
        family inet {
            address 10.2.1.1./24 {
                virtual-gateway-address 10.2.1.10;
            }
        }
    }
    unit 3 {
        family inet {
            address 10.3.1.1./24 {
                virtual-gateway-address 10.3.1.10;
            }
        }
    }
    unit 4 {
        family inet {
            address 10.4.1.1./24 {
                virtual-gateway-address 10.4.1.10;
            }
        }
    }
    unit 5 {
        family inet {
            address 10.5.1.1./24 {
                virtual-gateway-address 10.5.1.10;
            }
        }
    }
}

lo0 {

```

```

        unit 0 {
            family inet {
                address 192.168.1.1/32;
            }
        }
    }
}

routing-options {
    router-id 192.168.1.1;
    autonomous-system 65536;
    forwarding-table {
        export evpn-pplb;
    }
}

protocols {
    igmp {
        interface irb.1;
        interface irb.2;
        interface irb.3;
        interface irb.4;
        interface irb.5;
    }
    bgp {
        group INT {
            type internal;
            local-address 192.168.1.1;
            family evpn {
                signaling;
            }
            local-as 65536;
            neighbor 172.16.1.1;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface xe-0/0/1:0.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    pim {
        rp {

```

```

        static {
            address 172.16.1.1;
        }
    }
    interface irb.1 {
        distributed-dr;
    }
    interface irb.2 {
        distributed-dr;
    }
    interface irb.3 {
        distributed-dr;
    }
    interface irb.4 {
        distributed-dr;
    }
    interface irb.5 {
        distributed-dr;
    }
}
evpn {
    encapsulation vxlan;
    multicast-mode ingress-replication;
    extended-vni-list 1-5;
}
igmp-snooping {
    vlan VLAN1 {
        l2-querier {
            source-address 10.1.1.1;
        }
        proxy;
    }
    vlan VLAN2 {
        l2-querier {
            source-address 10.2.1.1;
        }
        proxy;
    }
    vlan VLAN3 {
        l2-querier {
            source-address 10.3.1.1;
        }
        proxy;
    }
}

```

```

    }
    vlan VLAN4 {
        l2-querier {
            source-address 10.4.1.1;
        }
        proxy;
    }
    vlan VLAN5 {
        l2-querier {
            source-address 10.5.1.1;
        }
        proxy;
    }
}
policy-options {
    policy-statement evpn-pplb {
        from protocol evpn;
        then {
            load-balance per-packet;
        }
    }
}
switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 192.168.1.1:1;
    vrf-target target:1:1;
}
vlans {
    VLAN1 {
        vlan-id 1;
        l3-interface irb.1;
        vxlan {
            vni 1;
        }
    }
    VLAN2 {
        vlan-id 2;
        l3-interface irb.2;
        vxlan {
            vni 2;
        }
    }
}

```

```

VLAN3 {
    vlan-id 3;
    l3-interface irb.3;
    vxlan {
        vni 3;
    }
}
VLAN4 {
    vlan-id 4;
    l3-interface irb.4;
    vxlan {
        vni 4;
    }
}
VLAN5 {
    vlan-id 5;
    l3-interface irb.5;
    vxlan {
        vni 5;
    }
}
}

```

Configuring 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* in the *CLI User Guide*.

To configure device PE2:

1. Specify the number of aggregated Ethernet logical interfaces.

```

[edit chassis]
user@PE2# set aggregated-devices ethernet device-count 2

```

2. Configure the interfaces.

```
[edit interfaces]
user@PE2# set xe-0/0/0:0 ether-options 802.3ad ae1
user@PE2# set xe-0/0/0:1 description "Connected to CE"
user@PE2# set xe-0/0/1:0 enable
user@PE2# set xe-0/0/1:0 unit 0 description "Connected to PE3"
user@PE2# set xe-0/0/1:0 unit 0 family inet address 198.51.100.1/24
user@PE2# set ae0 enable
user@PE2# set ae1 enable
user@PE2# set lo0 unit 0 family inet address 192.168.2.1/32
```

3. Configure active-active multihoming and enable the Link Aggregation Control Protocol (LACP) on each aggregated Ethernet interface.

```
[edit interfaces]
user@PE2# set ae0 esi 00:11:11:11:11:11:11:11:11:11
user@PE2# set esi all-active
user@PE2# set ae0 aggregated-ether-options lacp active
user@PE2# set ae0 aggregated-ether-options lacp periodic-fast
user@PE2# set ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:01
user@PE2# set ae1 esi 00:22:22:22:22:22:22:22:22:22
user@PE2# set ae1 esi all-active
user@PE2# set ae1 aggregated-ether-options lacp active
user@PE2# set ae1 aggregated-ether-options lacp periodic-fast
user@PE2# set ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:02
```

4. Configure each aggregated Ethernet interface as a trunk port.

```
[edit interfaces]
user@PE2# set ae0 unit 0 description "Connected to CE"
user@PE2# set ae0 unit 0 family ethernet-switching interface-mode trunk
user@PE2# set ae0 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3 VLAN4
VLAN5 ]
user@PE2# set ae1 unit 0 description "Connected to CE"
user@PE2# set ae1 unit 0 family ethernet-switching interface-mode trunk
user@PE2# set ae1 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3 VLAN4
VLAN5 ]
```

5. Configure IRB interfaces and virtual-gateway addresses.

```
[edit interfaces]
user@PE2# set irb unit 1 family inet address 10.1.1.2/24 virtual-gateway-address 10.1.1.10
user@PE2# set irb unit 2 family inet address 10.2.1.2/24 virtual-gateway-address 10.2.1.10
user@PE2# set irb unit 3 family inet address 10.3.1.2/24 virtual-gateway-address 10.3.1.10
user@PE2# set irb unit 4 family inet address 10.4.1.2/24 virtual-gateway-address 10.4.1.10
user@PE2# set irb unit 5 family inet address 10.5.1.2/24 virtual-gateway-address 10.5.1.10
```

6. Configure the autonomous system.

```
[edit routing-options]
user@PE2# set router-id 192.168.2.1
user@PE2# set autonomous-system 65536
```

7. Configure OSPF.

```
[edit protocols ospf]
user@PE2# set area 0.0.0.0 interface xe-0/0/1:0.0
user@PE2# set area 0.0.0.0 interface lo0. passive
```

8. Configure BGP internal peering.

```
[edit protocols bgp]
user@PE2# set group INT type internal
user@PE2# set group INT local-address 192.168.2.1
user@PE2# set group INT family evpn signaling
user@PE2# set group INT local-as 65536
user@PE2# set group INT neighbor 172.16.1.1
```

9. Configure the VLANs.

```
[edit vlans]
user@PE2# set VLAN1 vlan-id 1
user@PE2# set VLAN1 l3-interface irb.1
user@PE2# set VLAN1 vxlan vni 1
user@PE2# set VLAN2 vlan-id 2
user@PE2# set VLAN2 l3-interface irb.2
```



```

user@PE2# set VLAN2 vxlan vni 2
user@PE2# set VLAN3 vlan-id 3
user@PE2# set VLAN3 l3-interface irb.3
user@PE2# set VLAN3 vxlan vni 3
user@PE2# set VLAN4 vlan-id 4
user@PE2# set VLAN4 l3-interface irb.4
user@PE2# set VLAN4 vxlan vni 4
user@PE2# set VLAN5 vlan-id 5
user@PE2# set VLAN5 l3-interface irb.5
user@PE2# set VLAN2 vxlan vni 5

```

10. Enable EVPN.

```

[edit protocols evpn]
user@PE2# set encapsulation vxlan
user@PE2# set multicast-mode ingress-replication
user@PE2# set extended-vni-list 1-5

```

11. Configure an export routing policy to load balance EVPN traffic and apply it to the forwarding-table.

```

[edit policy-options]
user@PE2# set policy-statement evpn-pplb from protocol evpn
user@PE2# set policy-statement evpn-pplb then load-balance per packet

[edit routing-options]
user@PE2# set forwarding-table export evpn-pplb

```

12. Configure the source interface for the VXLAN tunnel.

```

[edit switch-options]
user@PE2# set vtep-source-interface lo0.0
user@PE2# set route-distinguisher 192.168.2.1:1
user@PE2# set vrf-target target:1:1

```

13. Enable IGMP on the IRB interfaces.

```

[edit protocols igmp]
user@PE2# set interface irb.1

```

```

user@PE2# set interface irb.2
user@PE2# set interface irb.3
user@PE2# set interface irb.4
user@PE2# set interface irb.5

```

14. Enable IGMP snooping on the IRB interfaces.

```

[edit protocols igmp-snooping vlan]
user@PE2# set VLAN1 l2-querier source-address 10.1.1.2
user@PE2# set VLAN1 proxy
user@PE2# set VLAN2 l2-querier source-address 10.2.1.2
user@PE2# set VLAN2 proxy
user@PE2# set VLAN3 l2-querier source address 10.3.1.2
user@PE2# set VLAN3 proxy
user@PE2# set VLAN4 l2-querier source-address 10.4.1.2
user@PE2# set VLAN4 proxy
user@PE2# set VLAN5 l2-querier source-address 10.5.1.2
user@PE2# set VLAN5 proxy

```

15. Configure PIM by defining a static rendezvous point and enabling on the IRB interfaces.



NOTE: This step is required only if you want to configure inter-VLAN forwarding. If your PE device is performing only intra-VLAN forwarding, omit this step.

```

[edit protocols pim]
user@PE2# set rp static address 172.16.1.1
user@PE2# set interface irb.1 distributed-dr
user@PE2# set interface irb.2 distributed-dr
user@PE2# set interface irb.3 distributed-dr
user@PE2# set interface irb.4 distributed-dr
user@PE2# set interface irb.5 distributed-dr

```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, `show vlans`, `show policy-options`, and `show switch-options` commands. If the output

does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE2# show chassis
aggregated-devices {
    ethernet {
        device-count 2;
    }
}
```

```
user@PE2# show
interfaces {
    xe-0/0/0:0 {
        ether-options {
            802.3ad ae1;
        }
    }
    xe-0/0/0:1 {
        description "Connected to CE";
        ether-options {
            802.3ad ae0;
        }
    }
    xe-0/0/1:0 {
        enable;
        unit 0 {
            description "Connected to PE3";
            family inet {
                address 198.51.100.1/24;
            }
        }
    }
}
ae0 {
    esi {
        00:11:11:11:11:11:11:11:11:11;
        all-active;
    }
    aggregated-ether-options {
        lacp {
            active;
            periodic fast;
        }
    }
}
```

```

        system-id 00:00:00:00:00:01;
    }
}
unit 0 {
    description "Connected to CE";
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ];
        }
    }
}
}
ae1 {
    enable;
    esi {
        00:22:22:22:22:22:22:22:22:22;
        all-active;
    }
    aggregated-ether-options {
        lacp {
            active;
            periodic fast;
            system-id 00:00:00:00:00:02;
        }
    }
    unit 0 {
        description "CONNECTED TO CE";
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ];
            }
        }
    }
}
}
irb {
    unit 1 {
        family inet {
            address 10.1.1.2/24 {
                virtual-gateway-address 10.1.1.10;
            }
        }
    }
}

```

```

    }
    unit 2 {
        family inet {
            address 10.2.1.2/24 {
                virtual-gateway-address 10.2.1.10;
            }
        }
    }
    unit 3 {
        family inet {
            address 10.3.1.2/24 {
                virtual-gateway-address 10.3.1.10;
            }
        }
    }
    unit 4 {
        family inet {
            address 10.4.1.2/24 {
                virtual-gateway-address 10.4.1.10;
            }
        }
    }
    unit 5 {
        family inet {
            address 10.5.1.2/24 {
                virtual-gateway-address 10.5.1.10;
            }
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.2.1/32;
        }
    }
}
}
routing-options {
    router-id 192.168.2.1;
    autonomous-system 65536;
    forwarding-table {
        export evpn-pplb;
    }
}

```

```

    }
}
protocols {
    igmp {
        interface irb.1;
        interface irb.2;
        interface irb.3;
        interface irb.4;
        interface irb.5;
    }
    bgp {
        group INT {
            type internal;
            local-address 192.168.2.1;
            family evpn {
                signaling;
            }
            local-as 65536;
            neighbor 172.16.1.1;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface xe-0/0/1:0.0;
        }
    }
    pim {
        rp {
            static {
                address 172.16.1.1;
            }
        }
        interface irb.1 {
            distributed-dr;
        }
        interface irb.2 {
            distributed-dr;
        }
        interface irb.3 {
            distributed-dr;
        }
    }
}

```

```

    }
    interface irb.4 {
        distributed-dr;
    }
    interface irb.5 {
        distributed-dr;
    }
}
evpn {
    encapsulation vxlan;
    multicast-mode ingress-replication;
    extended-vni-list 1-5;
}
igmp-snooping {
    vlan VLAN1 {
        l2-querier {
            source-address 10.1.1.2;
        }
        proxy;
    }
    vlan VLAN2 {
        l2-querier {
            source-address 10.2.1.2;
        }
        proxy;
    }
    vlan VLAN3 {
        l2-querier {
            source-address 10.3.1.2;
        }
        proxy;
    }
    vlan VLAN4 {
        l2-querier {
            source-address 10.4.1.2;
        }
        proxy;
    }
    vlan VLAN5 {
        l2-querier {
            source-address 10.5.1.2;
        }
        proxy;
    }
}

```

```

    }
  }
}
policy-options {
  policy-statement evpn-pplb {
    from protocol evpn;
    then {
      load-balance per-packet;
    }
  }
}
switch-options {
  vtep-source-interface lo0.0;
  route-distinguisher 192.168.2.1:1;
  vrf-target target:1:1;
}
vlans {
  VLAN1 {
    vlan-id 2;
    l3-interface irb.2;
    vxlan {
      vni 2;
    }
  }
  VLAN2 {
    vlan-id 1;
    l3-interface irb.1;
    vxlan {
      vni 1;
    }
  }
  VLAN3 {
    vlan-id 3;
    l3-interface irb.3;
    vxlan {
      vni 3;
    }
  }
  VLAN4 {
    vlan-id 4;
    l3-interface irb.4;
    vxlan {
      vni 4;
    }
  }
}

```



```

    }
  }
  VLAN5 {
    vlan-id 5;
    l3-interface irb.5;
    vxlan {
      vni 5;
    }
  }
}

```

Configuring CE Device

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 CE:

1. Specify the number of aggregated Ethernet logical interfaces.

```

[edit chassis]
user@CE# set aggregated-devices ethernet device-count 2

```

2. Configure the interfaces and enable LACP on the aggregated Ethernet interfaces.

```

[edit interfaces]
user@CE# set xe-0/2/0 ether-options 802.3ad ae0
user@CE# set xe-0/2/1 ether-options 802.3ad ae0
user@CE# set ae0 aggregated ether-options lacp active
user@CE# set ae0 aggregated ether-options lacp periodic fast
user@CE# set ae1 aggregated ether-options lacp active
user@CE# set ae1 aggregated ether-options lacp periodic fast

```

3. Create the Layer 2 customer bridge domains and the VLANs associated with the domains.

```

[edit]
user@CE# set vlans BD-1 vlan-id 1

```

```

user@CE# set vlans BD-2 vlan-id 2
user@CE# set vlans BD-3 vlan-id 3
user@CE# set vlans BD-4 vlan-id 4
user@CE# set vlans BD-5 vlan-id 5

```

4. Configure each interface to include in CE domain as a trunk port for accepting packets tagged with the specified VLAN identifiers.

```

[edit interfaces]
user@CE# set xe-0/2/0 unit 0 family ethernet-switching interface-mode trunk
user@CE# set xe-0/2/0 unit 0 family ethernet-switching vlan members 1-5
user@CE# set xe-0/2/1 unit 0 family ethernet-switching interface-mode trunk
user@CE# set xe-0/2/1 unit 0 family ethernet-switching vlan members 1-5
user@CE# set ae0 unit 0 family ethernet-switching interface-mode trunk
user@CE# set ae0 unit 0 family ethernet-switching vlan members 1-5
user@CE# set ae1 unit 0 family ethernet-switching interface-mode trunk
user@CE# set ae1 unit 0 family ethernet-switching vlan members 1-5

```

Results

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

```

user@CE# show chassis
aggregated-devices {
  ethernet {
    device-count 2;
  }
}

```

```

user@CE# show interfaces
xe-0/2/0 {
  ether-options {
    802.3ad ae0;
  }
  unit 0 {
    family ethernet-switching {
      interface-mode trunk;
    }
  }
}

```

```

        vlan {
            members 1-5;
        }
    }
}
xe-0/2/1 {
    ether-options {
        802.3ad ae0;
    }
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members 1-5;
            }
        }
    }
}
ae0 {
    aggregated-ether-options {
        lacp {
            active;
            periodic fast;
        }
    }
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members 1-5;
            }
        }
    }
}
ae1 {
    aggregated-ether-options {
        lacp {
            active;
            periodic fast;
        }
    }
    unit 0 {

```

```
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members 1-5;
            }
        }
    }
}
```

user@CE# **show vlans**

```
BD-1 {
    vlan-id 1;
    domain-type bridge;
}
BD-2 {
    vlan-id 2;
    domain-type bridge;
}
BD-3 {
    vlan-id 3;
    domain-type bridge;
}
BD-3 {
    vlan-id 3;
    domain-type bridge;
}
BD-4 {
    vlan-id 4;
    domain-type bridge;
}
BD-5 {
    vlan-id 5;
    domain-type bridge;
}
```

Configuring PE3

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 PE3:

1. Configure the interfaces.

```
[edit interfaces]
user@PE3# set xe-0/0/0 enable
user@PE3# set xe-0/0/0 unit 0 description "Connected to PE1"
user@PE3# set xe-0/0/0 unit 0 family inet 192.0.2.2/24
user@PE3# set xe-0/0/1 unit 0 description "Connected to PE2"
user@PE3# set xe-0/0/1 198.51.100.2/24
user@PE3# set lo0 unit 0 family inet address 172.16.1.1/32
```

2. Configure each logical Ethernet interface as a trunk port for accepting packets tagged with the specified VLAN identifiers.

```
[edit interfaces]
user@PE3# set xe-0/0/0:1 enable
user@PE3# set xe-0/0/0:1 unit 0 family ethernet-switching interface-mode trunk
user@PE3# set xe-0/0/0:1 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3
VLAN4 VLAN5 ]
user@PE3# set xe-0/0/1:1 enable
user@PE3# set xe-0/0/1:1 unit 0 family ethernet-switching interface-mode trunk
user@PE3# set xe-0/0/1:1 unit 0 family ethernet-switching vlan members [ VLAN1 VLAN2 VLAN3
VLAN4 VLAN5 ]
```

3. Configure IRB interfaces and virtual-gateway addresses.

```
[edit interfaces]
user@PE3# set irb unit 1 family inet address 10.1.1.5/24 virtual-gateway-address 10.1.1.10
user@PE3# set irb unit 2 family inet address 10.2.1.5/24 virtual-gateway-address 10.2.1.10
user@PE3# set irb unit 3 family inet address 10.3.1.5/24 virtual-gateway-address 10.3.1.10
```

```

user@PE3# set irb unit 4 family inet address 10.4.1.5/24 virtual-gateway-address 10.4.1.10
user@PE3# set irb unit 5 family inet address 10.5.1.5/24 virtual-gateway-address 10.5.1.10

```

4. Configure the autonomous system.

```

[edit routing-options]
user@PE3# set router-id 172.16.1.1
user@PE3# set autonomous-system 65536

```

5. Configure OSPF

```

[edit protocols ospf]
user@PE3# set area 0.0.0.0 interface all
user@PE3# set ospf area 0.0.0.0 fxp0.0 disable

```

6. Configure BGP internal peering with PE1 and PE2.

```

[edit protocols bgp]
user@PE3# set group INT type internal
user@PE3# set group INT local-address 172.16.1.1
user@PE3# set group INT family evpn signaling
user@PE3# set group INT local-as 65536
user@PE3# set group INT neighbor 192.168.1.1
user@PE3# set group INT neighbor 192.168.2.1

```

7. Configure the VLANs.

```

[edit vlans]
user@PE3# set VLAN1 vlan-id 1
user@PE3# set VLAN1 l3-interface irb.1
user@PE3# set VLAN1 vxlan vni 1
user@PE3# set VLAN2 vlan-id 2
user@PE3# set VLAN2 l3-interface irb.2
user@PE3# set VLAN2 vxlan vni 2
user@PE3# set VLAN3 vlan-id 3
user@PE3# set VLAN3 l3-interface irb.3
user@PE3# set VLAN3 vxlan vni 3
user@PE3# set VLAN4 vlan-id 4
user@PE3# set VLAN4 l3-interface irb.4

```

```

user@PE3# set VLAN4 vxlan vni 4
user@PE3# set VLAN5 vlan-id 5
user@PE3# set VLAN5 l3-interface irb.5
user@PE3# set VLAN2 vxlan vni 5

```

8. Enable EVPN.

```

[edit protocols evpn]
user@PE3# set encapsulation vxlan
user@PE3# set multicast-mode ingress-replication
user@PE3# set extended-vni-list 1-5

```

9. Configure an export routing policy to load balance EVPN traffic.

```

[edit policy-options]
user@PE3# set policy-statement evpn-pplb from protocol evpn
user@PE3# set policy-statement evpn-pplb then load-balance per packet

[edit routing-options]
user@PE3# set forwarding-table export evpn-pplb

```

10. Configure the source interface for the VXLAN tunnel.

```

[edit switch-options]
user@PE3# set vtep-source-interface lo0.0
user@PE3# set route-distinguisher 172.16.1.1:1
user@PE3# set vrf-target target:1:1

```

11. Enable IGMP on the IRB interfaces.

```

[edit protocols igmp]
user@PE1# set interface irb.1
user@PE1# set interface irb.2
user@PE1# set interface irb.3
user@PE1# set interface irb.4
user@PE1# set interface irb.5

```

12. Enable IGMP snooping on the IRB interfaces.

```
[edit protocols igmp-snooping vlan]
user@PE1# set VLAN1 l2-querier source-address 10.1.1.5
user@PE1# set VLAN1 proxy
user@PE1# set VLAN2 l2-querier source-address 10.2.1.5
user@PE1# set VLAN2 proxy
user@PE1# set VLAN3 l2-querier source address 10.3.1.5
user@PE1# set VLAN3 proxy
user@PE1# set VLAN4 l2-querier source-address 10.4.1.5
user@PE1# set VLAN4 proxy
user@PE1# set VLAN5 l2-querier source-address 10.5.1.5
user@PE1# set VLAN5 proxy
```

13. Configure PIM by defining the local rendezvous point and enabling on the IRB interfaces.



NOTE: This step is required only if you want to configure inter-VLAN forwarding. If your PE device is performing only intra-VLAN forwarding, omit this step.

```
[edit protocols pim]
user@PE1# set rp local address 172.16.1.1
user@PE1# set interface irb.1 distributed-dr
user@PE1# set interface irb.2 distributed-dr
user@PE1# set interface irb.3 distributed-dr
user@PE1# set interface irb.4 distributed-dr
user@PE1# set interface irb.5 distributed-dr
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, `show vlans`, `show policy-options`, and `show switch-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 {
  xe-0/0/0 {
    enable;
```



```

    unit 0 {
        description "Connected to PE1";
        family inet {
            address 192.0.2.2/24;
        }
    }
}
xe-0/0/1 {
    enable;
    unit 0 {
        description "Connected to PE2";
        family inet {
            address 198.51.100.2/24
        }
    }
}
xe-0/0/0:1 {
    enable;
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ];
            }
        }
    }
}
xe-0/0/1:1 enable;
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members [ VLAN1 VLAN2 VLAN3 VLAN4 VLAN5 ];
            }
        }
    }
}
irb {
    unit 1 {
        family inet {
            address 10.1.1.5/24 {
                virtual-gateway-address 10.1.1.10;
            }
        }
    }
}

```

```

    }
}
unit 2 {
    family inet {
        address 10.2.1.5/24 {
            virtual-gateway-address 10.2.1.10;
        }
    }
}
unit 3 {
    family inet {
        address 10.3.1.5/24 {
            virtual-gateway-address 10.3.1.10;
        }
    }
}
unit 4 {
    family inet {
        address 10.4.1.5/24 {
            virtual-gateway-address 10.4.1.10;
        }
    }
}
unit 5 {
    family inet {
        address 10.5.1.5/24 {
            virtual-gateway-address 10.5.1.10;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 172.16.1.1/32;
        }
    }
}
}
routing-options {
    router-id 172.16.1.1;
    autonomous-system 65536
    forwarding-table {

```

```

        export evpn-pplb;
    }
}
protocols {
    igmp {
        interface irb.1;
        interface irb.2;
        interface irb.3;
        interface irb.4;
        interface irb.5;
    }
    bgp {
        group INT {
            type internal;
            local-address 172.16.1.1;
            family evpn {
                signaling;
            }
            local-as 65546;
            neighbor 192.168.1.1;
            neighbor 192.168.2.1;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface all ;
            interface fxp0.0;
            disable;
        }
    }
}
pim {
    rp {
        local {
            address 172.16.1.1;
        }
    }
    interface irb.1 {
        distributed-dr;
    }
    interface irb.2 {
        distributed-dr;
    }
}

```

```

interface irb.3 {
    distributed-dr;
}
interface irb.4 {
    distributed-dr;
}
interface irb.5 {
    distributed-dr;
}
}
evpn {
    encapsulation vxlan;
    multicast-mode ingress-replication;
    extended-vni-list 1-5;
}
igmp-snooping {
    vlan VLAN1 {
        l2-querier {
            source-address 10.1.1.5;
        }
        proxy;
    }
    vlan VLAN2 {
        l2-querier {
            source-address 10.2.1.5;
        }
        proxy;
    }
    vlan VLAN3 {
        l2-querier {
            source-address 10.3.1.5;
        }
        proxy;
    }
    vlan VLAN4 {
        l2-querier {
            source-address 10.4.1.5;
        }
        proxy;
    }
    vlan VLAN5 {
        l2-querier {
            source-address 10.5.1.5;
        }
    }
}

```

```

        }
        proxy;
    }
}
}
policy-options {
    policy-statement evpn-pplb {
        from protocol evpn;
        then {
            load-balance per-packet;
        }
    }
}
switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 172.16.1.1:1;
    vrf-target target:1:1;
}
vlans {
    VLAN1 {
        vlan-id 1;
        l3-interface irb.1;
        vxlan {
            vni 1;
        }
    }
    VLAN2 {
        vlan-id 2;
        l3-interface irb.2;
        vxlan {
            vni 2;
        }
    }
    VLAN3 {
        vlan-id 3;
        l3-interface irb.3;
        vxlan {
            vni 3;
        }
    }
    VLAN4 {
        vlan-id 4;
        l3-interface irb.4;
    }
}

```

```

        vxlan {
            vni 4;
        }
    }
    VLAN5 {
        vlan-id 5;
        l3-interface irb.5;
        vxlan {
            vni 5;
        }
    }
}

```

Verification

IN THIS SECTION

- [Verifying IGMP Messages are Synced | 997](#)
- [Verifying Source Addresses Are Learned and Multicast Traffic Is Being Forwarded | 1000](#)

Confirm that the configuration is working properly.

Verifying IGMP Messages are Synced

Purpose

Verify on each PE that IGMP join and leave messages are synced.

Action

From operational mode, run the `show evpn instance extensive` command.

```

user@PE1> show evpn instance extensive
Instance: default-switch
Route Distinguisher: 192.168.1.1:1
Encapsulation type: VXLAN
MAC database status          Local  Remote
MAC advertisements:         25      40

```

```

MAC+IP advertisements:          10      15
Default gateway MAC advertisements:  10      10
Number of local interfaces: 3 (3 up)
Interface name  ESI                      Mode      Status    AC-Role
ae0.0          00:11:11:11:11:11:11:11:11 all-active Up         Root
ae1.0          00:22:22:22:22:22:22:22:22 all-active Up         Root
xe-0/0/1:1.0   00:00:00:00:00:00:00:00:00 single-homed Up         Root
Number of IRB interfaces: 5 (5 up)
Interface name  VLAN  VNI  Status  L3 context
irb.1           1     Up   master
irb.2           2     Up   master
irb.3           3     Up   master
irb.4           4     Up   master
irb.5           5     Up   master
Number of bridge domains: 5
VLAN  Domain ID  Intfs / up  IRB intf  Mode      MAC sync  IM route label  SG
sync  IM core nexthop
1     1           3 3      irb.1     Extended  Enabled  1
Enabled 2097159
2     2           3 3      irb.2     Extended  Enabled  2
Enabled 2097161
3     3           3 3      irb.3     Extended  Enabled  3
Enabled 2097162
4     4           3 3      irb.4     Extended  Enabled  4
Enabled 2097163
5     5           3 3      irb.5     Extended  Enabled  5
Enabled 2097164
Number of neighbors: 2
Address          MAC    MAC+IP    AD      IM      ES Leaf-label
192.168.2.1     25     10        9       5       0
172.16.1.1      15     10        1       5       0
Number of ethernet segments: 7
ESI: 00:11:11:11:11:11:11:11:11
Status: Resolved by IFL ae0.0
Local interface: ae0.0, Status: Up/Forwarding
Number of remote PEs connected: 1
Remote PE      MAC label  Aliasing label  Mode
192.168.2.1    5          0               all-active
DF Election Algorithm: MOD based
Designated forwarder: 192.168.2.1
Backup forwarder: 192.168.1.1
Last designated forwarder update: Jul 13 01:22:45
ESI: 00:22:22:22:22:22:22:22:22

```

```

Status: Resolved by IFL ae1.0
Local interface: ae1.0, Status: Up/Forwarding
Number of remote PEs connected: 1
  Remote PE      MAC label  Aliasing label  Mode
  192.168.2.1    5          0              all-active
DF Election Algorithm: MOD based
Designated forwarder: 192.168.2.1
Backup forwarder: 192.168.1.1
Last designated forwarder update: Jul 13 21:02:47
ESI: 05:00:00:00:64:00:00:00:01:00
Local interface: irb.1, Status: Up/Forwarding
Number of remote PEs connected: 2
  Remote PE      MAC label  Aliasing label  Mode
  192.168.2.1    1          0              all-active
  172.16.1.1     1          0              single-homed
ESI: 05:00:00:00:64:00:00:00:02:00
Local interface: irb.2, Status: Up/Forwarding
Number of remote PEs connected: 2
  Remote PE      MAC label  Aliasing label  Mode
  192.168.2.1    2          0              all-active
  172.16.1.1     2          0              single-homed
ESI: 05:00:00:00:64:00:00:00:03:00
Local interface: irb.3, Status: Up/Forwarding
Number of remote PEs connected: 2
  Remote PE      MAC label  Aliasing label  Mode
  192.168.2.1    3          0              all-active
  172.16.1.1     3          0              single-homed
ESI: 05:00:00:00:64:00:00:00:04:00
Local interface: irb.4, Status: Up/Forwarding
Number of remote PEs connected: 2
  Remote PE      MAC label  Aliasing label  Mode
  192.168.2.1    4          0              all-active
  172.16.1.1     4          0              single-homed
ESI: 05:00:00:00:64:00:00:00:05:00
Local interface: irb.5, Status: Up/Forwarding
Number of remote PEs connected: 2
  Remote PE      MAC label  Aliasing label  Mode
  172.16.1.1     5          0              all-active
  192.168.2.1    5          0              all-active
Router-ID: 192.168.1.1

```


Meaning

The SG Sync is Enabled and the IM Core next-hop field displays a valid route.

Verifying Source Addresses Are Learned and Multicast Traffic Is Being Forwarded

Purpose

Verify on each PE that multicast receivers have learned the source interface for the VXLAN tunnel.

Action

From operational mode, enter the `show evpn igmp-snooping database extensive l2-domain-id 1` command and the `show igmp snooping evpn database vlan VLAN1` commands.

These commands displays output for VLAN1. You can use them to display output for each configured VLAN

From operational mode, enter the `show evpn multicast-snooping next-hops` to verify that downstream interface has been learned.

```

user@PE1> show evpn igmp-snooping database extensive
l2-domain-id 1
Instance: default-switch
  VN Identifier: 1
    Group IP: 225.1.1.1
      Access OIF Count: 2
        Interface      ESI              Local  Remote
        ae1.0          00:22:22:22:22:22:22:22:22 1        0
        ae0.0          00:11:11:11:11:11:11:11:11 0        1
      Remote OIF Count: 2, Core NH: 2097159
        Interface      Nbr
        vtep.32770     192.168.2.1
        vtep.32769     172.16.1.1
    Group IP: 225.1.1.2
      Access OIF Count: 2
        Interface      ESI              Local  Remote
        ae1.0          00:22:22:22:22:22:22:22:22 1        0
        ae0.0          00:11:11:11:11:11:11:11:11 0        1
      Remote OIF Count: 2, Core NH: 2097159
        Interface      Nbr
        vtep.32770     192.168.2.1

```

```

vtep.32769 172.16.1.1
Group IP: 225.1.1.3
Access OIF Count: 2
      Interface          ESI          Local  Remote
      ae0.0             00:11:11:11:11:11:11:11:11 1         0
      ae1.0             00:22:22:22:22:22:22:22:22 1         0
Remote OIF Count: 2, Core NH: 2097159
      Interface          Nbr
      vtep.32770         192.168.2.1
      vtep.32769         172.16.1.1
Group IP: 225.1.1.4
Access OIF Count: 2
      Interface          ESI          Local  Remote
      ae0.0             00:11:11:11:11:11:11:11:11 1         0
      ae1.0             00:22:22:22:22:22:22:22:22 0         1
Remote OIF Count: 2, Core NH: 2097159
      Interface          Nbr
      vtep.32770         192.168.2.1
      vtep.32769         172.16.1.1
Group IP: 225.1.1.5
Access OIF Count: 2
      Interface          ESI          Local  Remote
      ae0.0             00:11:11:11:11:11:11:11:11 1         0
      ae1.0             00:22:22:22:22:22:22:22:22 1         0
Remote OIF Count: 2, Core NH: 2097159
      Interface          Nbr
      vtep.32770         192.168.2.1
      vtep.32769         172.16.1.1
Group IP: 225.1.1.6
Access OIF Count: 2
      Interface          ESI          Local  Remote
      ae0.0             00:11:11:11:11:11:11:11:11 0         1
      ae1.0             00:22:22:22:22:22:22:22:22 1         0
Remote OIF Count: 2, Core NH: 2097159
      Interface          Nbr
      vtep.32770         192.168.2.1
      vtep.32769         172.16.1.1
Group IP: 225.1.1.7
Access OIF Count: 2
      Interface          ESI          Local  Remote
      ae0.0             00:11:11:11:11:11:11:11:11 0         1
      ae1.0             00:22:22:22:22:22:22:22:22 1         0
Remote OIF Count: 2, Core NH: 2097159

```

```

      Interface      Nbr
      vtep.32770     192.168.2.1
      vtep.32769     172.16.1.1
Group IP: 225.1.1.8
Access OIF Count: 2
      Interface      ESI          Local  Remote
      ae0.0          00:11:11:11:11:11:11:11:11 1      0
      ae1.0          00:22:22:22:22:22:22:22:22 0      1
Remote OIF Count: 2, Core NH: 2097159
      Interface      Nbr
      vtep.32770     192.168.2.1
      vtep.32769     172.16.1.1
Group IP: 225.1.1.9
Access OIF Count: 2
      Interface      ESI          Local  Remote
      ae0.0          00:11:11:11:11:11:11:11:11 1      0
      ae1.0          00:22:22:22:22:22:22:22:22 0      1
Remote OIF Count: 2, Core NH: 2097159
      Interface      Nbr
      vtep.32770     192.168.2.1
      vtep.32769     172.16.1.1
Group IP: 225.1.1.10
Access OIF Count: 2
      Interface      ESI          Local  Remote
      ae0.0          00:11:11:11:11:11:11:11:11 0      1
      ae1.0          00:22:22:22:22:22:22:22:22 0      1
Remote OIF Count: 2, Core NH: 2097159
      Interface      Nbr
      vtep.32770     192.168.2.1
      vtep.32769     172.16.1.1

```

```

user@PE1> show igmp snooping evpn database vlan
VLAN1
Instance: default-switch
Bridge-Domain: VLAN1, VN Identifier: 1
Group IP: 225.1.1.1
Core NH: 2097159
Access OIF Count: 3
      Interface  Local  Remote
      ae0.0      0      1
      xe-0/0/1:1.0 1      0

```

```

        ae1.0          1          0
Group IP: 225.1.1.2
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
        ae0.0         0         1
        xe-0/0/1:1.0   1         0
        ae1.0         1         0
Group IP: 225.1.1.3
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
        xe-0/0/1:1.0   1         0
        ae1.0         1         0
        ae0.0         1         0
Group IP: 225.1.1.4
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
        ae1.0         0         1
        xe-0/0/1:1.0   1         0
        ae0.0         1         0
Group IP: 225.1.1.5
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
        ae1.0         1         0
        xe-0/0/1:1.0   1         0
        ae0.0         1         0
Group IP: 225.1.1.6
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
        ae0.0         0         1
        ae1.0         1         0
        xe-0/0/1:1.0   1         0
Group IP: 225.1.1.7
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
        ae0.0         0         1
        ae1.0         1         0
        xe-0/0/1:1.0   1         0

```

```

Group IP: 225.1.1.8
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
      ae1.0        0        1
      xe-0/0/1:1.0  1        0
      ae0.0        1        0
Group IP: 225.1.1.9
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
      ae1.0        0        1
      xe-0/0/1:1.0  1        0
      ae0.0        1        0
Group IP: 225.1.1.10
Core NH: 2097159
Access OIF Count: 3
      Interface    Local    Remote
      ae1.0        0        1
      ae0.0        0        1
      xe-0/0/1:1.0  1        0

```

```

user@PE1> show evpn multicast-snooping next-hops
Family: INET
ID          Refcount KRefCount Downstream interface Addr
2097159     3         1 vtep.32769
              vtep.32770
2097161     3         1 vtep.32769
              vtep.32770
2097162     3         1 vtep.32769
              vtep.32770
2097163     3         1 vtep.32769
              vtep.32770
2097164     3         1 vtep.32769
              vtep.32770

```

RELATED DOCUMENTATION

[Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment](#) | 934

Overview of Selective Multicast Forwarding

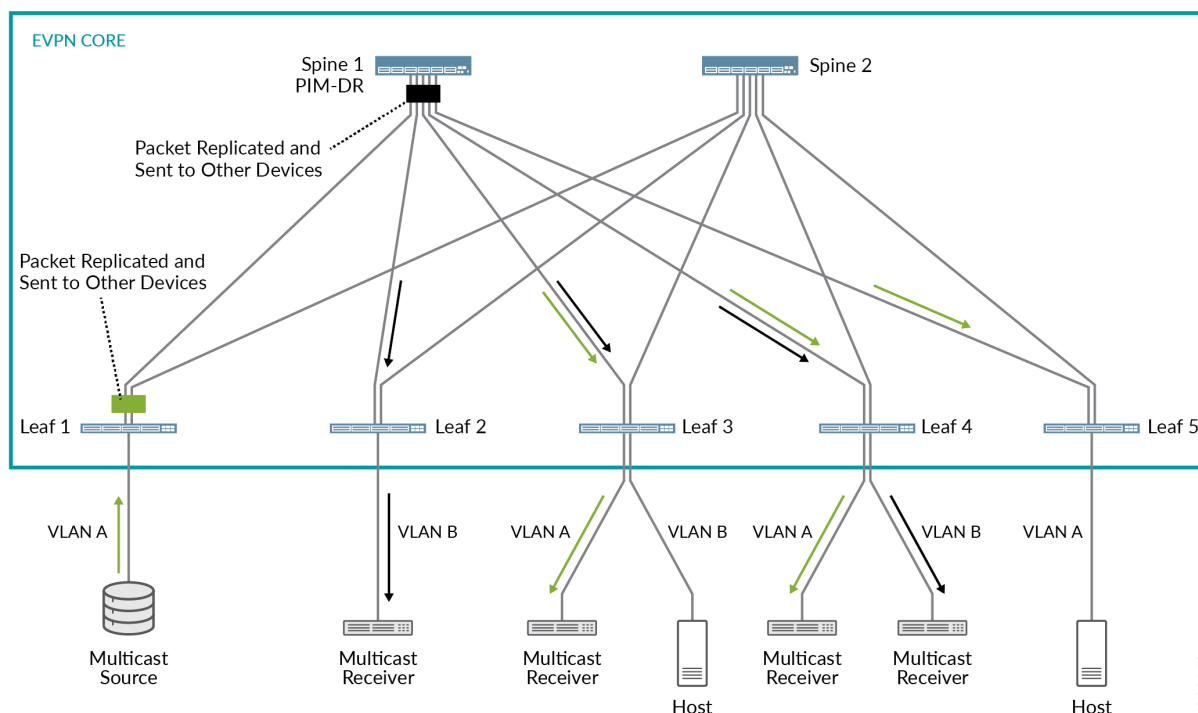
IN THIS SECTION

- [Benefits of Selective Multicast Forwarding | 1008](#)
- [Limitations of Selective Multicast Forwarding | 1008](#)

Prior to Junos OS Release 18.4R1, PE devices with IGMP snooping enabled only constrained the local multicast traffic going to its access interface. For intra-VLAN multicast traffic in a leaf-spine topology, when a leaf device receives multicast traffic from another device or from a multicast sender on one of its access interface, it replicates and forwards the multicast traffic to its own interfaces with interested receivers. The leaf device will also replicate and send the multicast traffic to all the leaf and spine devices in the EVPN core. For inter-VLAN multicast traffic, the spine device in a centrally routed EVPN-VXLAN network would route the multicast traffic between the VLANs through the IRB interface.

Figure 1 shows how inclusive multicast traffic flows in a centrally-routed EVPN network.

Figure 92: Inclusive Multicast Forwarding in a Centrally-Routed EVPN-VXLAN Network



When the multicast source sends a packet from VLAN A, the following flows occur:

- **Flow 1: Intra-VLAN traffic**—Based on the ingress replication mechanism, Leaf 1 replicates and switches the packet to all spine and other leaf devices. Leafs 3 and 4, on which VLAN A is configured, receive and forward the packet to the connected multicast receivers. Leaf 5, which is also on VLAN A network still receives the multicast packet even though there are no receivers.
- **Flow 2: Inter-VLAN traffic**—Upon receipt of the packet from Leaf 1 as described in flow 1, Spine 1, which is the PIM DR, takes the following action:
 - Routes the packet over the IRB interface associated with VLAN B.
 - Based on the ingress replication mechanism, Spine 1 replicates and forwards the packet to the other spine and leafs.

Leafs 2 and 4, on which VLAN B is configured, receive and forward the packet to the connected multicast receivers. Leaf 3 receives the multicast packet, but does not forward the packet since it does not have a receiver.

Starting in Junos OS Release 18.4R1, devices with IGMP snooping enabled use selective multicast forwarding by default in a centrally routed EVPN-VXLAN network to replicate and forward multicast traffic.

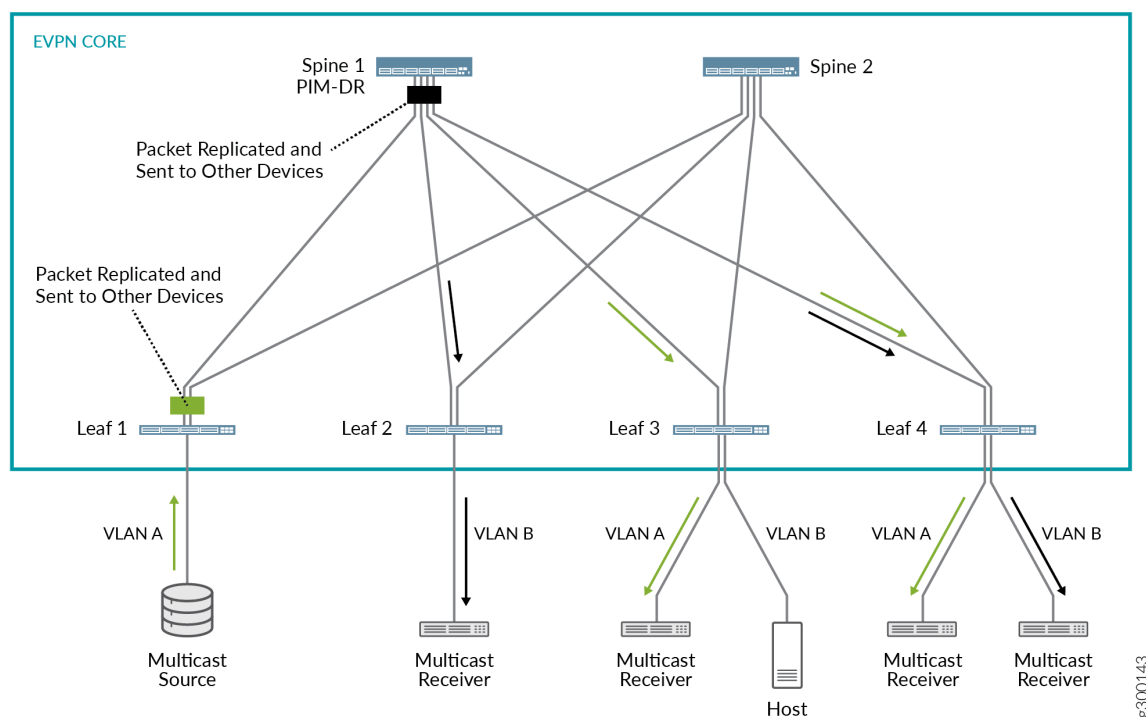
Similarly, starting in Junos OS Release 20.4R1, QFX Series switches with MLD snooping enabled in a centrally routed EVPN-VXLAN network also enable selective multicast forwarding by default.

This is how selective multicast forwarding works with IGMP or MLD snooping on the leaf devices:

- IGMP or MLD snooping allows leaf devices to send multicast traffic only to the access interface with an interested receiver.
- When you enable IGMP or MLD snooping, a leaf device selectively sends multicast traffic to only the other leaf devices across the EVPN core that have expressed an interest in that multicast group.
- With selective multicast forwarding, leaf devices also always send multicast traffic to a spine device so that the spine device can route inter-VLAN multicast traffic through its IRB interfaces.

Figure 2 shows how selective multicast traffic flows in a centrally-routed EVPN network.

Figure 93: Selective Multicast Forwarding in a Centrally-Routed EVPN-VXLAN Network



When the multicast source sends a packet from VLAN A, the following flows occur:

- **Flow 1: Intra-VLAN traffic**—Based on the ingress replication mechanism, Leaf 1 replicates and switches the packet to all spine and other leaf devices with interested receivers. In this case, leafs 3 and 4, on which VLAN A is configured and forward the packet to the connected interested multicast receivers.

- **Flow 2: Inter-VLAN traffic**—Upon receipt of the packet from Leaf 1 as described in flow 1, Spine 1, which is the PIM DR, takes the following action:
 - Routes the packet over the IRB interface associated with VLAN B.
 - Based on the ingress replication mechanism, Spine 1 replicates and forwards the packet to the other spine and interested leaf devices.

Leafs 2 and 4, on which VLAN B is configured, receive and forward the packet to the connected multicast receivers. Leaf 3 does not have receivers and does not get a multicast packet.

Benefits of Selective Multicast Forwarding

Selective multicast forwarding provides greater network efficiency and reduces traffic in the EVPN network. Selective multicast forwarding conserves bandwidth usage in the core and reduces the load on egress PE devices that do not have listeners. The benefits of selective multicast forwarding increases when there are listeners in multiple VLANs.

Limitations of Selective Multicast Forwarding

- Supported in a centrally-routed leaf-spine topology.
- Support for EVPN-VXLAN encapsulation only.
- Support for EVPN-ELAN services only.
- Supported with IGMPv2, MLDv1, and MLDv2 traffic on particular QFX Series platforms and Junos OS releases.

RELATED DOCUMENTATION

[Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment | 934](#)

[igmp-snooping](#)

[mld-snooping](#)

Configuring the Number of SMET Nexthops

Junos OS uses the EVPN Type 6 selective multicast Ethernet tag (SMET) route message to support the following:

- External multicast sender (IGMP proxy)—EVPN Type 6 route messages are used within the EVPN network. The ingress PE device translates the IGMP message to an EVPN Type 6 route message and the egress PE device translates the EVPN Type 6 route message back to an IGMP message.
- Inter-VLAN multicast—EVPN Type 6 route messages are used to create a PIM states on a PE device with an IRB interface. This allows multicast traffic to be sent across VLANs.
- Selective multicast forwarding—The EVPN Type 6 route messages are used to distribute the routing information indicating a PE device's interest for a multicast group.

The information in the EVPN Type 6 route message are used to build a list of SMET next hops, which can be used to selectively replicate and forward multicast packets. SMET next hop is a list of outgoing interfaces (OIFs) identifying the interested PEs. Multicast groups are mapped to SMET next hops. If multicast groups that have the same set of interested PEs, they can share a SMET next hop. The number of SMET nexthops defaults to 10,000 and can be increased by configuring the `smet-nexthop-limit` option. When a device reaches the SMET nexthop limit, the device will start using inclusive multicast forwarding for multicast traffic.

Devices in the network that do not support snooping or cannot send EVPN Type 6 route messages are always included in the SMET next hop. This practice ensures those devices that do not support EVPN Type 6 route messages will be able to receive multicast traffic.

To configure the number of SMET nexthops, you can use the following statement:

```
User@PE1# set forwarding-options multicast-replication evpn smet-nexthop-limit nexthop range.
```

In some cases, you may want to bypass selective multicast forwarding and send multicast traffic to all devices. If you wish to send multicast traffic to a group, you can list the multicast group address with the following statement

```
User@PE1# set multicast-snooping-options flood-groups [ip-addresses]
```



NOTE: OSPF messages, which use multicast addresses for communication, are automatically included in the multicast snooping forwarding table.

RELATED DOCUMENTATION

[*multicast-replication*](#)

[multicast-snooping-options](#)

[*show evpn igmp-snooping proxy*](#)

[*show evpn instance*](#)

[show multicast snooping route](#)

| *show route table*

Assisted Replication Multicast Optimization in EVPN Networks

SUMMARY

Assisted replication (AR) helps to optimize multicast traffic flow in EVPN networks by offloading traffic replication to devices that can more efficiently handle the task.

IN THIS SECTION

- [Assisted Replication in EVPN Networks | 1010](#)
- [How AR Works | 1013](#)
- [AR Leaf Device Load Balancing with Multiple Replicators | 1018](#)
- [AR Limitations with IGMP Snooping or MLD Snooping | 1028](#)
- [AR with Optimized Intersubnet Multicast \(OISM\) | 1029](#)
- [Multicast Forwarding Use Cases in an EVPN Network with AR Devices | 1032](#)
- [Configure Assisted Replication | 1038](#)
- [Verify Assisted Replication Setup and Operation | 1043](#)

Assisted Replication in EVPN Networks

IN THIS SECTION

- [Benefits of Assisted Replication | 1011](#)
- [AR Device Roles | 1012](#)

Assisted replication (AR) is a Layer 2 (L2) multicast traffic optimization feature supported in EVPN networks. With AR enabled, an ingress device replicates a multicast stream to another device in the EVPN network that can more efficiently handle replicating and forwarding the stream to the other devices in the network. For backward compatibility, ingress devices that don't support AR operate

transparently with other devices that have AR enabled. These devices use ingress replication to distribute the traffic to other devices in the EVPN network.

AR provides an overlay multicast optimization L2 solution. AR does not require PIM to be enabled in the underlay.



NOTE: This documentation describes AR functions in terms of multicast traffic replication and forwarding, but applies to any broadcast, unknown unicast, and multicast (BUM) traffic in general.

In releases before Junos OS Release 22.2R1, you can enable AR only in the default switch EVPN instance on supported Junos OS devices. Starting in Junos OS Release 22.2R1, you can also enable AR on supported Junos OS devices in MAC-VRF EVPN instances.

Starting in Junos OS Evolved Release 22.2R1, you can enable AR on supported Junos OS Evolved devices in MAC-VRF EVPN instances.

As of Junos OS and Junos OS Evolved Release 22.2R1, you can also configure both AR and optimized intersubnet multicast (OISM) in the same EVPN-VXLAN fabric.



On Junos OS Evolved devices, we support EVPN-VXLAN using EVPN configurations with MAC-VRF instances only (and not in the default switch instance). As a result, on these devices we support AR only in MAC-VRF EVPN instances.

Benefits of Assisted Replication

- In an EVPN network with a significant volume of broadcast, unknown unicast, and multicast (BUM) traffic, AR helps to optimize BUM traffic flow by passing replication and forwarding tasks to other devices that have more capacity to better handle the load.
- In an EVPN-VXLAN environment, AR reduces the number of hardware next hops to multiple remote virtual tunnel endpoints (VTEPs) over traditional multicast ingress replication.
- AR can operate with other multicast optimizations such as IGMP snooping, MLD snooping, and selective multicast Ethernet tag (SMET) forwarding.
- AR operates transparently with devices that do not support AR for backward compatibility in existing EVPN networks.

AR Device Roles

To enable AR, you configure devices in the EVPN network into AR replicator and AR leaf roles. For backward compatibility, your EVPN network can also include devices that don't support AR. Devices that don't support AR operate in a regular network virtualization edge (NVE) device role.

[Table 54 on page 1012](#) summarizes these roles:

Table 54: AR Device Roles

Role	Description	Supported Platforms
AR leaf device	Device in an EVPN network that offloads multicast ingress replication tasks to another device in the EVPN network to handle the replication and forwarding load.	<p>Starting in Junos OS Releases 18.4R2 and 19.4R1: QFX5110, QFX5120, and QFX10000 line of switches</p> <p>Starting in Junos OS Release 22.2R1: EX4650</p> <p>Starting in Junos OS Evolved Release 22.2R1: QFX5130-32CD and QFX5700</p>
AR replicator device	Device in an EVPN network that helps perform multicast ingress replication and forwarding for traffic received from AR leaf devices on an AR overlay tunnel to other ingress replication overlay tunnels.	<p>Starting in Junos OS Releases 18.4R2 and 19.4R1: QFX10000 line of switches</p> <p>Starting in Junos OS Evolved Release 22.2R1: QFX5130-32CD and QFX5700 (and with OISM, in standalone mode only; see "AR with Optimized Intersubnet Multicast (OISM)" on page 1029 for details)</p>
Regular NVE device	Device that does not support AR or that you did not configure into an AR role in an EVPN network. The device replicates and forwards multicast traffic using the usual EVPN network ingress replication.	N/A

If you enable other supported multicast optimization features in your EVPN network, such as IGMP snooping, MLD snooping, and SMET forwarding, AR replicator and AR leaf devices forward traffic in the EVPN core or on the access side only toward interested receivers.

If you want to enable AR in a fabric with OISM, see ["AR with Optimized Intersubnet Multicast \(OISM\)" on page 1029](#) for considerations when planning which devices to configure as AR replicators and AR leaf devices in that case.



NOTE: With supported EX Series and QFX Series AR devices in EVPN-VXLAN networks, AR replicator and AR leaf devices must operate in extended AR mode. In this mode, AR leaf devices that have multihomed Ethernet segments share some of the replication load with the AR replicator devices, which enables them to use split horizon and local bias rules to prevent traffic loops and duplicate forwarding. See ["Extended AR Mode for Multihomed Ethernet Segments" on page 1017](#). For details on how devices in an EVPN-VXLAN environment forward traffic to multihomed Ethernet segments using local bias and split horizon filtering, see ["EVPN-over-VXLAN Supported Functionality" on page 599](#).

How AR Works

IN THIS SECTION

- [AR Route Advertisements | 1016](#)
- [Extended AR Mode for Multihomed Ethernet Segments | 1017](#)

In general, devices in the EVPN network use ingress replication to distribute BUM traffic. The ingress device (where the source traffic enters the network) replicates and sends the traffic on overlay tunnels to all other devices in the network. For EVPN topologies with EVPN multihoming (ESI-LAGs), network devices employ local bias or designated forwarder (DF) rules to avoid duplicating forwarded traffic to receivers on multihomed Ethernet segments. With multicast optimizations like IGMP snooping or MLD snooping and SMET forwarding enabled, forwarding devices avoid sending unnecessary traffic by replicating the traffic only to other devices that have active listeners.

AR operates within the existing EVPN network overlay and multicast forwarding mechanisms. However, AR also defines special AR overlay tunnels over which AR leaf devices pass traffic from multicast sources to AR replicator devices. AR replicator devices treat incoming source traffic on AR overlay tunnels as requests to replicate the traffic toward other devices in the EVPN network on behalf of the sending AR leaf device.

AR basically works as follows:

1. Each AR replicator device advertises its replicator capabilities and AR IP address to the EVPN network using EVPN Type 3 (inclusive multicast Ethernet tag [IMET]) routes. You configure a

secondary IP address on the loopback interface (lo0) as an AR IP address when you assign the replicator role to the device. AR leaf devices use the secondary IP address for the AR overlay tunnel.

2. An AR leaf device receives these advertisements to learn about available AR replicator devices and their capabilities.
3. AR leaf devices advertise EVPN Type 3 routes for ingress replication that include the AR leaf device ingress replication tunnel IP address.
4. The AR leaf device forwards multicast source traffic to a selected AR replicator on its AR overlay tunnel.

When the network has multiple AR replicator devices, AR leaf devices automatically load-balance among them. (See ["AR Leaf Device Load Balancing with Multiple Replicators"](#) on page 1018.)

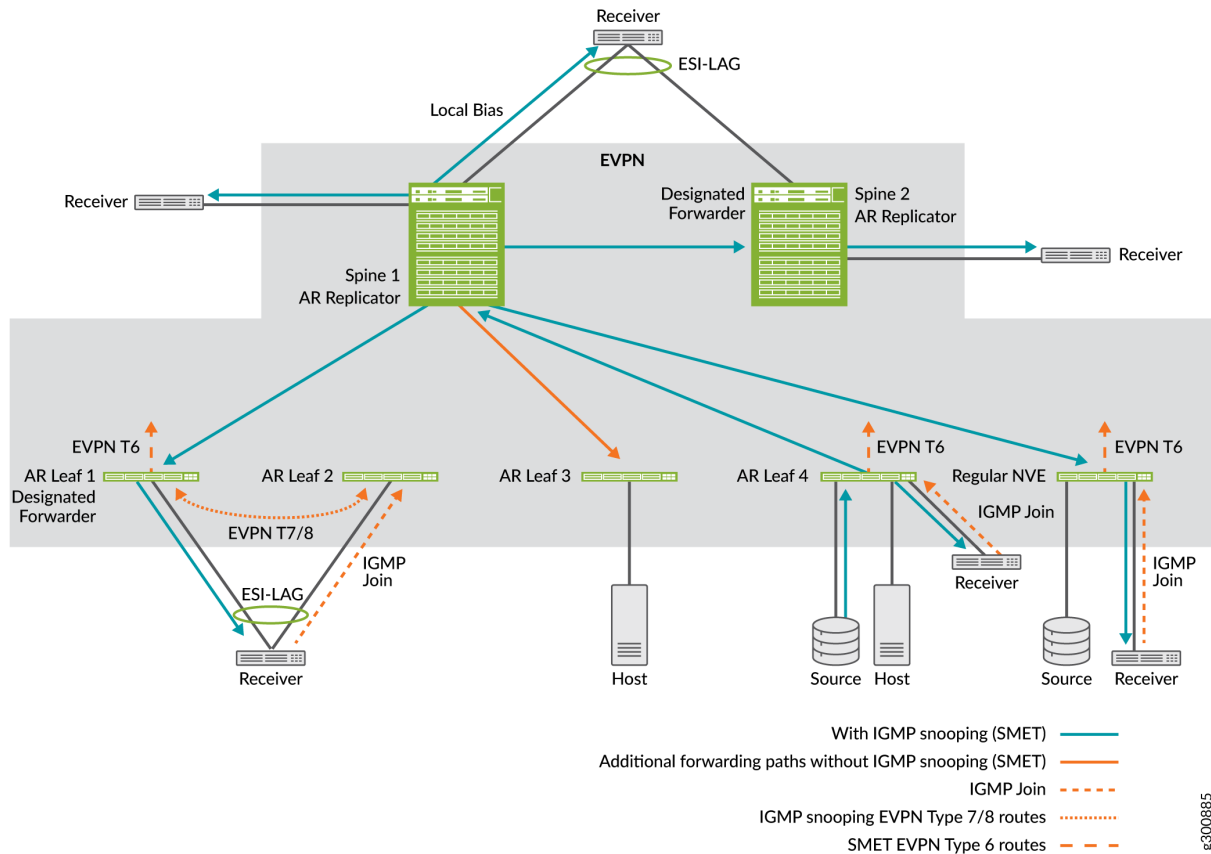
5. The AR replicator device receives multicast source traffic from an AR leaf device on an AR overlay tunnel and replicates it to the other devices in the network using the established ingress replication overlay tunnels. It also forwards the traffic toward local receivers and external gateways, including those on multihomed Ethernet segments, using the same local bias or DF rules it would use for traffic received from directly connected sources or regular NVE devices.



NOTE: When AR operates in extended AR mode, an AR leaf device with a multihomed source handles the replication to its EVPN multihoming peers, and the AR replicator device receiving source traffic from that AR leaf device skips forwarding to the multihomed peers. (See ["Extended AR Mode for Multihomed Ethernet Segments"](#) on page 1017; [Figure 100 on page 1037](#) illustrates this use case.)

[Figure 94 on page 1015](#) shows an example of multicast traffic flow in an EVPN network with two AR replicators, four AR leaf devices, and a regular NVE device.

Figure 94: AR Traffic Flow



In [Figure 94 on page 1015](#), AR Leaf 4 forwards multicast source traffic on an AR overlay tunnel to Spine 1, one of the available AR replicators. Spine 1 receives the traffic on the AR tunnel and replicates it on the usual ingress replication overlay tunnels to all other devices in the EVPN network, including AR leaf devices, other AR replicators, and regular NVE devices.

In this example, Spine 1 also forwards the traffic to its local receivers. Applying local bias rules, Spine 1 forwards the traffic to its multihomed receiver regardless of whether it is the DF for that multihomed Ethernet segment. In addition, according to split horizon rules, AR replicator Spine 1 does not forward the traffic to AR Leaf 4. Instead, AR Leaf 4, the ingress device, does local bias forwarding to its attached receiver.



NOTE: QFX5130-32CD and QFX5700 switches serving as AR replicators don't support replication and forwarding from locally-attached sources or to locally-attached receivers.

When you enable IGMP snooping or MLD snooping, you also enable SMET forwarding by default. With SMET, the AR replicator skips sending the traffic on ingress replication overlay tunnels to any devices that don't have active listeners.

Figure 94 on page 1015 shows a simplified view of the control messages and traffic flow with IGMP snooping enabled:

1. Receivers attached to AR Leaf 1 (multihomed to AR Leaf 2), AR Leaf 4, and Regular NVE send IGMP join messages to express interest in receiving the multicast stream.
2. For IGMP snooping to work with EVPN multihoming and avoid duplicate traffic, multihomed peers AR Leaf 1 and AR Leaf 2 synchronize the IGMP state using EVPN Type 7 and 8 (Join Sync and Leave Sync) routes.

Spine 1 and Spine 2 do the same for their multihomed receiver, though the figure doesn't show that specifically.
3. The EVPN devices hosting receivers (and only the DF for multihomed receivers) advertise EVPN Type 6 routes in the EVPN core (see ["Overview of Selective Multicast Forwarding" on page 1005](#)), so forwarding devices only send the traffic to the other EVPN devices with interested receivers.

For IPv6 multicast traffic with MLD and MLD snooping enabled, you see the same behavior as illustrated for IGMP snooping.

Regular NVE devices and other AR replicator devices don't send source traffic on AR overlay tunnels to an AR replicator device, and AR replicators don't perform AR tasks when receiving multicast source traffic on regular ingress replication overlay tunnels. AR replicators forward traffic they did not receive on AR tunnels the way they normally would using EVPN network ingress replication.

Similarly, if an AR leaf device doesn't see any available AR replicators, the AR leaf device defaults to using the usual ingress replication forwarding behavior (the same as a regular NVE device). In that case, AR show commands display the AR operational mode as No replicators/Ingress Replication. See the `show evpn multicast-snooping assisted-replication replicators` command for more details.

AR Route Advertisements

Devices in an EVPN network advertise EVPN Type 3 (IMET) routes for regular ingress replication and AR.

AR replicator devices advertise EVPN Type 3 routes for regular ingress replication that include:

- The AR replicator device ingress replication tunnel IP address
- Tunnel type—IR
- AR device role—AR-REPLICATOR

AR replicators also advertise EVPN Type 3 routes for the special AR overlay tunnels that include:

- An AR IP address you configure on a loopback interface on the AR replicator device (see the *replicator* configuration statement).

- Tunnel type—AR.
- AR device role—AR-REPLICATOR.
- EVPN multicast flags extended community with the Extended-MH-AR flag when operating in extended AR mode (see ["Extended AR Mode for Multihomed Ethernet Segments" on page 1017](#)).

AR leaf devices advertise EVPN Type 3 routes for regular ingress replication that include:

- The AR leaf device ingress replication tunnel IP address
- Tunnel type—IR
- AR device role—AR-LEAF

Regular NVE devices don't support AR or are devices you didn't configure as AR leaf devices. Regular NVE devices ignore AR route advertisements and forward multicast traffic using the usual EVPN network ingress replication rules.

Extended AR Mode for Multihomed Ethernet Segments

EVPN networks employ split horizon and local bias rules to enable multicast optimizations when the architecture includes multihomed Ethernet segments (see ["EVPN-over-VXLAN Supported Functionality" on page 599](#)). These methods include information about the ingress AR leaf device in forwarded packets to ensure that the traffic isn't unnecessarily forwarded or looped back to the source by way of the ingress device's multihomed peers.

Some devices serving in the AR replicator role can't retain the source IP address or Ethernet segment identifier (ESI) label on behalf of the ingress AR leaf device when forwarding the traffic to other overlay tunnels. AR replicator devices with this limitation operate in *extended AR mode*, where the AR replicator forwards traffic toward other devices in the EVPN network with the AR replicator's ingress replication IP address as the source IP address.



NOTE: QFX Series AR devices in an EVPN-VXLAN environment use extended AR mode. We currently only support AR in VXLAN overlay architectures, and only support extended AR mode.

AR replicators include the EVPN multicast flags extended community in the EVPN Type 3 AR route advertisement. The Extended-MH-AR flag indicates the AR operating mode.

When AR devices operate in extended AR mode:

- Besides forwarding the traffic to the AR replicator device, an AR leaf device with multihomed sources *also* handles replicating the traffic to its multihoming peers.

- An AR replicator device receiving source traffic from an AR leaf device with multihomed sources skips forwarding to the AR leaf device's multihoming peers.

See ["Source behind an AR Leaf Device \(Multihomed Ethernet Segment\) with Extended AR Mode"](#) on [page 1036](#), which shows the traffic flow in extended AR mode for an EVPN network with a multihomed source.

Extended AR mode behavior applies only when AR leaf devices have multihomed Ethernet segments with other AR leaf devices, not with AR replicators from which the AR leaf requests replication assistance. When AR replicator and AR leaf devices share multihomed Ethernet segments in environments that require extended AR mode, AR can't work correctly. As a result, ingress AR leaf devices don't use the AR tunnels and default to using only ingress replication. In this case, AR show commands display the AR operational mode as Misconfiguration/Ingress Replication.

AR replicators that can retain the source IP address or ESI label of the ingress AR leaf device operate in *regular AR mode*.

See the `show evpn multicast-snooping assisted-replication replicators` command for more information about AR operational modes.

AR Leaf Device Load Balancing with Multiple Replicators

IN THIS SECTION

- [Default AR Leaf Load-Balancing When Detecting Multiple AR Replicators | 1018](#)
- [Deterministic Load Balancing and Traffic Steering to AR Replicators | 1019](#)

When the EVPN network has more than one advertised AR replicator device, the AR leaf devices automatically load-balance among the available AR replicator devices.

Default AR Leaf Load-Balancing When Detecting Multiple AR Replicators

For QFX Series switches in an EVPN-VXLAN network:

- AR leaf devices that are EX4650 switches or switches in the QFX5000 line designate a particular AR replicator device for a VLAN or VXLAN network identifier (VNI) to load-balance among the available AR replicators. This default method on these devices uses a hash mod algorithm based on the VLAN or VNI and the number of AR replicators.
- AR leaf devices in the QFX10000 line of switches actively load-balance among the available AR replicators based on traffic flow levels (packets per second) within a VLAN or VNI. This method on

these devices use a flow level hash mod algorithm based on the multicast group and the number of AR replicators.

Deterministic Load Balancing and Traffic Steering to AR Replicators

For AR leaf devices that don't use a flow level load balancing mechanism by default (see ["Default AR Leaf Load-Balancing When Detecting Multiple AR Replicators" on page 1018](#)), we support a feature with which you can deterministically configure AR leaf devices to send multicast flows to particular AR replicators.



NOTE: We support setting deterministic AR replicator policies only:

- In EVPN-VXLAN networks running AR with OISM.
- For multicast traffic flows.

For unknown unicast and broadcast traffic, the device uses the usual bridge domain or VLAN flooding behavior.

With this deterministic AR replicator load balancing method, you:

1. Define routing policies to match one or more multicast flows in the policy from conditions.

For example, you can set a condition to match a particular multicast group address using the route-filter *multicast-group-address* statement at the [edit policy-options policy-statement *name* from] hierarchy.

2. In the then clause of each routing policy, for the policy action, specify the AR replicator(s) to which the AR leaf device sends the matching flows.

You set the policy action with the assisted-replication *replicator-ip replicator-ip-addr* statement at the [edit policy-options policy-statement *name* then] hierarchy. (See *assisted-replication (Deterministic AR Replicator Policy Actions)*.)

You have options to strictly steer matching flows only to a preferred AR replicator, or also include a fallback AR replicator to use in case the preferred one goes down.

3. Assign the routing policy or policies to an EVPN instance on the AR leaf device using the *deterministic-ar-policy* statement at the [edit routing-instances *name* protocols evpn assisted-replication leaf] hierarchy level. (See *deterministic-ar-policy*.)

An example use case where you might want to deterministically steer AR leaf traffic to particular replicators is when an AR leaf device serves a source device where you know particular multicast flows are usually significantly larger or smaller. [Figure 95 on page 1020](#) shows how the AR leaf device might load-balance the flows using the default load balancing method that results in a very unbalanced traffic distribution.

Figure 95: Default AR Load Balancing with Multicast Flows of Different Sizes

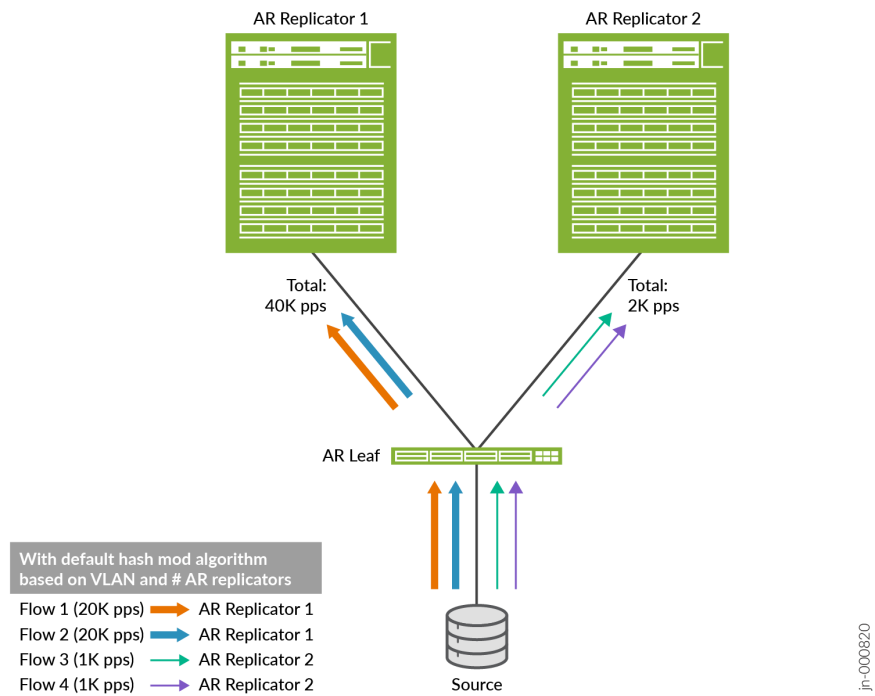


Figure 96 on page 1021 illustrates how you can configure deterministic load balancing to more equally distribute (steer) the flows among the available replicators.

Figure 96: Deterministic AR Load Balancing to Handle Multicast Flows of Significantly Different Sizes



AR Load Balancing Routing Policy Options

Use the guidelines here to define and assign AR load balancing routing policies.

You can include routing policy match conditions for IPv4 or IPv6 multicast traffic flows associated with IGMP or MLD multicast groups as follows:

- With IGMPv2 or MLDv1, you can match a multicast flow based on either or both of the following:
 - A multicast group.
 - A bridge domain or VLAN-mapped virtual network identifier (VNI).



NOTE: This feature introduces the bridge-domain *bridge-domain-name* option in the policy statement from clause so you can specify a bridge domain or VLAN-mapped VNI as a match condition.

With the source-specific multicast (SSM) protocols IGMPv3 or MLDv2, you can match a multicast flow based on any combination of the following:

- A multicast group.
- A bridge domain or VLAN-mapped VNI.
- A multicast source address.

In the routing policy from clause, use the following options to set the match conditions for the desired multicast flows:

- **Multicast group:** route-filter
- **Bridge domain or VLAN-mapped VNI:** bridge-domain
- **Multicast source:** source-address-filter

See *Routing Policy Match Conditions* for more on the routing policy match options mentioned here.

See ["Sample Deterministic AR Policy Configurations" on page 1027](#) for some sample policy match condition configurations.



NOTE: To match MLDv1 or MLDv2 IPv6 multicast flows, include the family inet6 option in the from clause when you specify IPv6 addresses or prefixes with the route-filter and source-address-filter match condition options.

In the then clause, you can configure the deterministic AR replicator policy action (*assisted-replication* (*Deterministic AR Replicator Policy Actions*)) to use either of the following modes, which are mutually exclusive:

- **Strict mode**—Strictly direct flows to a specified preferred AR replicator. If the preferred AR replicator goes down, the AR leaf device drops the matching flows.

To configure strict mode, include the strict option with the AR deterministic load balancing policy action when you specify the AR replicator IP address:

```
[edit policy-options policy-statement name then]
user@ar-leaf# set assisted-replication replicator-ip replicator-ip strict
```

- **Loose mode**—This is the default mode if you don't include the strict option. In this mode, you can optionally include a fallback AR replicator to use if the specified AR replicator goes down.

To specify a fallback AR replicator, include the `fallback-replicator-ip` *fallback-replicator-ip* option with the AR deterministic load balancing policy action when you specify the AR replicator IP address:

```
[edit policy-options policy-statement name then]
user@ar-leaf# set assisted-replication replicator-ip replicator-ip fallback-replicator-ip
fallback-replicator-ip
```



NOTE: The AR replicator IP addresses for the primary and fallback AR replicators are the secondary loopback addresses you assign to the AR replicator function when you configure those devices as AR replicators. See ["Configure an AR Replicator Device" on page 1040](#) for details.

Behavior Notes on Deterministic AR Load Balancing Policies

Note the following behaviors with deterministic AR load balancing policies:

- When you assign a deterministic AR policy to an EVPN instance, the device applies the policy only for advertised EVPN Type 6 routing table entries. The device creates core next-hop multicast snooping entries for those routes.
- Deterministic AR policies match a multicast source based on the presence of the source address in the EVPN Type 6 routes, which are in the EVPN Type 6 advertisements only with SSM IGMPv3 or MLDv2 flows.
- If no local receivers generate EVPN Type 6 routes expressing interest in a multicast group, the device still sends locally sourced multicast flows for the group to the border leaf devices so the traffic can reach any external receivers.
- For the multicast flows that don't match any of the policies you assign on an AR leaf device, the device uses its default AR load balancing method. See ["Default AR Leaf Load-Balancing When Detecting Multiple AR Replicators" on page 1018](#) for more on the default method on different platforms.
- In loose mode (no strict option), if the preferred AR replicator isn't available and you didn't specify a fallback or the fallback is also down, the AR leaf device uses its default AR replicator load balancing method to distribute any multicast traffic among the remaining available AR replicators. (See ["Default AR Leaf Load-Balancing When Detecting Multiple AR Replicators" on page 1018](#).)

If the AR leaf device doesn't detect any available AR replicators in the network, the device reverts to the default multicast traffic ingress replication behavior (the device doesn't use AR at all).



NOTE: Ingress replication is the default multicast traffic forwarding method with AR on:

- Devices in the network that don't support AR.
 - Leaf devices that support AR but you didn't configure them as AR leaf devices.
 - Leaf devices you configure as AR leaf devices, but those AR leaf devices don't detect any available AR replicators.
- If you apply a deterministic AR policy on multihoming peer AR leaf devices, those devices ignore any such policies when forwarding to each other. Instead, those devices invoke extended AR mode forwarding behavior—they use ingress replication to forward multicast source traffic directly to their peer AR leaf devices in the same multihomed Ethernet segment. See ["Source behind an AR Leaf Device \(Multihomed Ethernet Segment\) with Extended AR Mode" on page 1036](#) for details on how this use case works.
 - If you change a policy assignment or assigned policy conditions, you might see some multicast traffic loss until the routing tables converge after the policy update.

Assign a Deterministic AR Policy to an EVPN Instance on an AR Leaf Device

To enable a deterministic AR policy on an AR leaf device, assign the policy to an EVPN instance on the device using the `deterministic-ar-policy` statement at the `[edit routing-instances name protocols evpn assisted-replication leaf]` hierarchy level.

For example, if you define a policy called `arpol1`, you can assign it to the EVPN instance called `evpn-vxlanA` as follows:

```
set routing-instances evpn-vxlanA protocols evpn assisted-replication leaf deterministic-ar-policy arpol1;
```

You can use the `show evpn igmp-snooping proxy` command with the `deterministic-ar` option to see the configured mode (strict or loose), preferred AR replicator, and fallback AR replicator (if any).

Use the `show evpn multicast-snooping assisted-replication replicators` command to see the available AR replicators and the VTEP interface toward each one.

For example:

- **Strict mode example:** With a strict mode policy configuration such as the following arpol1, you configure to strictly use a preferred AR replicator:

```
set policy-options policy-statement arpol1 from route-filter 233.252.0.2/24 orlonger;
set policy-options policy-statement arpol1 from bridge-domain 1100;
set policy-options policy-statement arpol1 then assisted-replication replicator-ip
192.168.104.1 strict
set policy-options policy-statement arpol1 then accept;
set routing-instances evpn-vxlanA protocols evpn assisted-replication leaf deterministic-ar-
policy arpol1;
```

You can see the mode and preferred AR replicator, and follow the next hops for the flow to see that the device uses the available preferred AR replicator:

```
user@ar-leaf1> show evpn igmp-snooping proxy deterministic-ar
Instance: evpn-vxlanA
VN Identifier: 1100
  Group          Source   Local  Remote  Corenh   Flood  Deterministic-AR(Mode/
Preferred replicator/Fallback)
  233.252.0.1    0.0.0.0   0      1       4375     0      N/A
  233.252.0.2    0.0.0.0   0      1       4372     0      Strict/192.168.104.1/0.0.0.0

user@ar-leaf1> show evpn multicast-snooping next-hops 4372 detail
Family: INET
ID          Refcount   KRefCount  Downstream interface Addr
4372        6           2          vtep.32778-(11148)
Flags 0x2100 type 0x18 members 0/0/0/1/0
Address 0x769b404

user@ar-leaf1> show evpn multicast-snooping assisted-replication replicators
Instance: evpn-vxlanA
AR Role: AR Leaf

VN Identifier: 1100
Operational Mode: Extended AR
  Replicator IP  Nexthop Index  Interface      Mode
  192.168.104.1  11148          vtep.32778     Extended AR
  192.168.105.1  11357          vtep.32782     Extended AR
  192.168.106.1  11180          vtep.32780     Extended AR
```



NOTE: With a strict mode deterministic AR policy, the `show evpn igmp-snooping proxy deterministic-ar` command displays a null fallback replicator address (0.0.0.0).

- **Loose mode example:** With a loose mode policy configuration such as the following `arpol2`, you enable the device to use a fallback AR replicator in case the preferred AR replicator becomes unavailable:

```
set policy-options policy-statement arpol2 from route-filter 233.252.0.2/24 orlonger;
set policy-options policy-statement arpol2 from bridge-domain 1100;
set policy-options policy-statement arpol2 then assisted-replication replicator-ip
192.168.104.1 fallback-replicator-ip 192.168.105.1
set policy-options policy-statement arpol2 then accept;
set routing-instances evpn-vxlanA protocols evpn assisted-replication leaf deterministic-ar-
policy arpol2;
```

You can see in the case below that the AR leaf device didn't detect the preferred AR replicator, so the device uses the specified fallback AR replicator instead:

```
user@ar-leaf1> show evpn igmp-snooping proxy deterministic-ar
Instance: evpn-vxlanA
VN Identifier: 1100
  Group      Source   Local  Remote  Corenh   Flood  Deterministic-AR(Mode/
Preferred replicator/Fallback)
  233.252.0.1 0.0.0.0  0      1       4375     0      N/A
  233.252.0.2 0.0.0.0  0      1       4453     0      Loose/
192.168.104.1/192.168.105.1
```

```
user@ar-leaf1> show evpn multicast-snooping next-hops 4453 detail
Family: INET
ID      Refcount  KRefCount  Downstream interface Addr
4453    3          1          vtep.32782-(11357)
Flags 0x2100 type 0x18 members 0/0/0/1/0
Address 0x55a0ce415604
```

```
user@ar-leaf1> show evpn multicast-snooping assisted-replication replicators
Instance: evpn-vxlanA
AR Role: AR Leaf

VN Identifier: 1100
Operational Mode: Extended AR
```

Replicator IP	NextHop Index	Interface	Mode
192.168.105.1	11357	vtep.32782	Extended AR
192.168.106.1	11180	vtep.32780	Extended AR

Sample Deterministic AR Policy Configurations

Here are a few sample deterministic AR replicator routing policies:

- This sample policy matches IGMP multicast flows for multicast groups under 233.252.0.2/24 and bridge domain or VNI 1100. The configuration steers those flows to AR replicator 192.168.104.1 in strict mode (no fallback or default load balancing):

```
policy-options {
  policy-statement arpol1 {
    from {
      route-filter 233.252.0.2/24 orlonger;
      bridge-domain-id 1100;
    }
    then {
      assisted-replication replicator-ip 192.168.104.1 strict;
      accept;
    }
  }
}
```

- This sample policy matches multicast flows for IGMP multicast groups under 233.252.0.2/24 and any bridge domain or VNI. The configuration steers those flows to AR replicator 192.168.104.1 in loose mode with AR replicator 192.168.105.1 as the fallback:

```
policy-options {
  policy-statement arpol2 {
    from {
      route-filter 233.252.0.2/24 orlonger;
    }
    then {
      assisted-replication replicator-ip 192.168.104.1 fallback-replicator-ip 192.168.105.1;
      accept;
    }
  }
}
```

- This sample policy matches MLDv2 multicast flows for IPv6 multicast groups under ffe9::2:1/112 and ffe9::2:2/112, any bridge domain or VNI, and source IPv6 address 2001:db8::1/128. The

configuration steers those flows to AR replicator 192.168.104.1 in loose mode with AR replicator 192.168.105.1 as the fallback:



NOTE: Include `family inet6` in the `from` clause for IPv6 multicast flows with MLD.

```
policy-options {
  policy-statement arpol5 {
    from {
      family inet6;
      route-filter ffe9::2:1/112 orlonger;
      route-filter ffe9::2:2/112 orlonger
      source-address-filter 2001:db8::1/128 exact
    }
    then {
      assisted-replication replicator-ip 192.168.104.1 fallback-replicator-ip 192.168.105.1;
      accept;
    }
  }
}
```

AR Limitations with IGMP Snooping or MLD Snooping

AR replicator devices with IGMP snooping or MLD snooping enabled will silently drop multicast traffic toward AR leaf devices that don't support IGMP snooping or MLD snooping in an EVPN-VXLAN environment.

If you want to include IGMP snooping or MLD snooping multicast traffic optimization with AR, you can work around this limitation by disabling AR on the EVPN devices that don't support IGMP snooping or MLD snooping so they don't function as AR leaf devices. Those devices then behave like regular NVE devices and can receive the multicast traffic by way of the usual EVPN network ingress replication, although without the benefits of AR and IGMP snooping or MLD snooping optimizations. You can configure devices that do support IGMP snooping or MLD snooping in this environment as AR leaf devices with IGMP snooping or MLD snooping.

See [Table 54 on page 1012](#) for more about the behavior differences between AR leaf devices and regular NVE devices.



BEST PRACTICE: For best results in EVPN-VXLAN networks with a combination of leaf devices that don't support IGMP snooping or MLD snooping (such as QFX5100 switches), and leaf devices that do support IGMP snooping or MLD snooping (such as

QFX5120 switches), follow these recommendations based on the VTEP scaling you need:

- VTEP scaling with 100 or more VTEPs: Enable AR for all supporting devices in the network. Don't use IGMP snooping or MLD snooping.
- VTEP scaling with less than 100 VTEPs: Enable AR and IGMP snooping on the AR replicators and other AR leaf devices that support IGMP snooping or MLD snooping. Don't enable AR on leaf devices that don't support IGMP snooping or MLD snooping, so those devices then act as regular NVE devices and not as AR leaf devices.

AR with Optimized Intersubnet Multicast (OISM)

IN THIS SECTION

- [AR and OISM Device Roles | 1030](#)
- [Guidelines to Integrate AR and OISM Device Roles | 1030](#)
- [How AR and OISM Work Together | 1031](#)

OISM is a multicast traffic optimization feature that operates at L2 and Layer 3 (L3) in EVPN-VXLAN edge-routed bridging (ERB) overlay fabrics. With OISM, leaf devices in the fabric route intersubnet multicast traffic locally through IRB interfaces. This design minimizes the amount of traffic the devices send out into the EVPN core and avoids traffic hairpinning. OISM uses IGMP snooping (or MLD snooping) and selective multicast forwarding (SMET) to further limit EVPN core traffic only to destinations with interested listeners. Finally, OISM enables ERB fabrics to effectively support multicast traffic between sources and receivers inside and outside of the fabric.

In contrast, AR focuses on optimizing the L2 functions of replicating and forwarding BUM traffic in the fabric.



NOTE: OISM optimizes only multicast traffic flow, not broadcast or unknown unicast traffic flows. AR helps to optimize any BUM traffic flows.

Starting in Junos OS and Junos OS Evolved Release 22.2R1, you can enable AR with OISM on supported devices in an ERB overlay fabric. With this support, OISM uses the symmetric bridge domains model, in which you configure the same subnet information symmetrically on all OISM devices.

AR and OISM Device Roles

With AR, you configure devices to function in the AR replicator or AR leaf role. When you don't assign an AR role to a device, the device functions as a regular NVE device. See [Table 54 on page 1012](#) for more on these roles.

With OISM, you configure devices in an ERB overlay fabric to function in one of the following roles:

- OISM border leaf role.
- OISM server leaf role.
- No OISM role—These devices are usually lean spine devices in the fabric.

See ["Optimized Intersubnet Multicast in EVPN Networks" on page 1049](#) for details on how OISM works and how to configure OISM devices.

When you integrate AR and OISM functions, we support the AR replicator role in the following modes:

- Collocated: You configure the AR replicator role on the same device as the OISM border leaf role.
- Standalone: The AR replicator function is not collocated with the OISM border leaf role on the same device. In this case, the AR replicator is usually a lean spine device in an ERB fabric running OISM.



NOTE: On QFX5130-32CD and QFX5700 switches, we only support standalone mode. You can configure the AR replicator role only on a device in the fabric that isn't also an OISM border leaf device.

Guidelines to Integrate AR and OISM Device Roles

Use these guidelines to integrate AR and OISM roles on the devices in the fabric:

- You can configure any OISM leaf devices (border leaf or server leaf) with the AR leaf role.
- Use the following rules when you configure the AR replicator role:
 1. We support the AR replicator role on any of following devices: QFX5130-32CD, QFX5700, QFX10002, QFX10008, and QFX10016.



NOTE: QFX5130-32CD and QFX5700 devices support the AR replicator role only in standalone mode.

2. The devices you use as AR replicators must be devices that support OISM, even if you configure an AR replicator in standalone mode (see ["AR and OISM Device Roles" on page 1030](#)). The AR

replicator devices must have the same tenant virtual routing and forwarding (VRF) and VLAN information as the OISM devices (see rule "3" on page 1031).

3. Even if the AR replicator device is in standalone mode, you must configure the device with the same tenant VRF instances, corresponding IRB interfaces, and member VLANs as the OISM leaf devices. The AR replicator requires this information to install the correct L2 multicast states to properly forward the multicast traffic.

See ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices "](#) on page 1158 for details on the OISM configuration steps you duplicate on standalone AR replicators for these elements.

How AR and OISM Work Together

When you enable AR and OISM together, you can configure higher capacity devices in the fabric to do the replication and lessen the load on the OISM leaf devices. With OISM, the ingress leaf device replicates and sends copies of the packets on the ingress VLAN toward interested receivers, where:

- The ingress leaf device might be an OISM server leaf device or an OISM border leaf device.
- For multicast traffic sourced within the fabric, the ingress VLAN is a revenue VLAN.
- For multicast traffic coming into the fabric from an external multicast source, the ingress VLAN is the supplemental bridge domain (SBD).

When you configure the AR leaf role on the ingress OISM leaf device, the device sends one copy of the traffic toward the AR replicator. The AR replicator takes care of replicating and forwarding the traffic toward the other OISM server leaf or border leaf devices. The OISM leaf devices do local routing to minimize the multicast traffic in the EVPN core.



NOTE: OISM server leaf and border leaf devices send EVPN Type 6 routes into the EVPN core when their receivers join a multicast group. OISM devices derive the multicast (*,G) states for IGMPv2 or (S,G) states for IGMPv3 from these EVPN Type 6 routes. The devices install these derived states on the OISM SBD and revenue bridge domain VLANs in the MAC-VRF instance for the VLANs that are part of OISM-enabled L3 tenant VRF instances. If you enable AR with OISM, the AR replicator devices use the Type 6 routes to install multicast states in the same way the OISM devices do. However, QFX5130-32CD and QFX5700 switches configured as AR replicators with OISM might have scaling issues when they install the multicast states in a fabric with many VLANs. As a result, these switches install the derived multicast states only on the OISM SBD VLAN. They don't install these states on all OISM revenue bridge domain

VLANs. On these devices when configured as AR replicators, you see multicast group routes only on the SBD in `show multicast snooping route` command output.

See ["OISM and AR Scaling with Many VLANs" on page 1124](#) for details.

See the following illustrations of AR and OISM working together:

- ["AR and OISM with an Internal Multicast Source" on page 1106](#)
- ["AR and OISM with an External Multicast Source" on page 1110](#)

See ["Configure Assisted Replication" on page 1038](#) for details on how to configure AR, including when you integrate AR and OISM device roles.

Multicast Forwarding Use Cases in an EVPN Network with AR Devices

IN THIS SECTION

- [Source behind a Regular NVE Device | 1033](#)
- [Source behind an AR Replicator Device | 1034](#)
- [Source behind an AR Leaf Device \(Single-Homed Ethernet Segment\) | 1035](#)
- [Source behind an AR Leaf Device \(Multihomed Ethernet Segment\) with Extended AR Mode | 1036](#)

This section shows multicast traffic flow in several common use cases where the source is located behind different AR devices or regular NVE devices in an EVPN network.

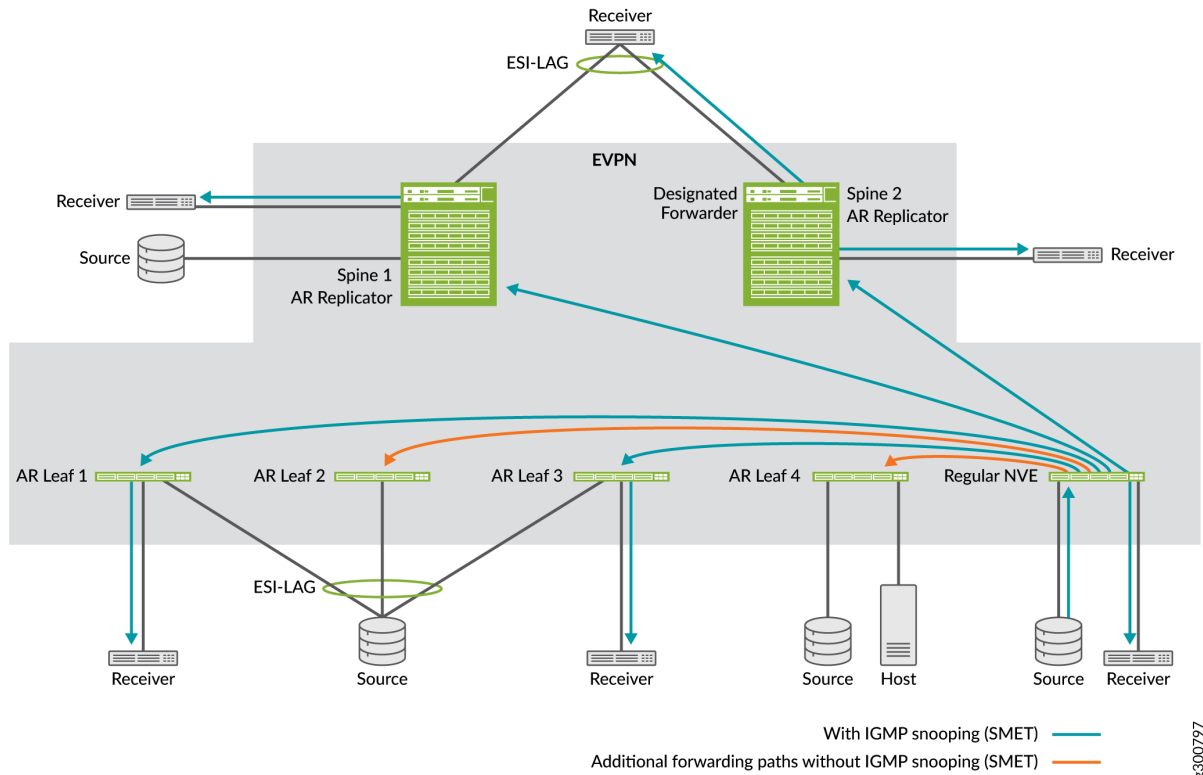
Some use cases or aspects of these use cases don't apply based on the following platform limitations with AR:

- QFX5130-32CD and QFX5700 devices serving as AR replicators don't support replication and forwarding from locally-attached sources or to locally-attached receivers. As a result, with these switches as AR replicators, we don't support the following:
 - The use case in ["Source behind an AR Replicator Device" on page 1034](#).
 - Replicating to a single-homed or multihomed receiver locally attached to an AR replicator device as the other use cases show.

Source behind a Regular NVE Device

Figure 97 on page 1033 shows an EVPN network with multicast traffic from a source attached to Regular NVE, a device that doesn't support AR.

Figure 97: Multicast Source Traffic Ingress at a Regular NVE Device



In this case, forwarding behavior is the same whether you enable AR or not, because the ingress device, Regular NVE, is not an AR leaf device sending traffic for replication to an AR replicator. Regular NVE employs the usual ingress replication forwarding rules as follows:

- Without IGMP snooping or MLD snooping, and without SMET forwarding enabled, Regular NVE floods the traffic to all the other devices in the EVPN network.
- With IGMP snooping or MLD snooping, and with SMET forwarding enabled, Regular NVE only forwards the traffic to other devices in the EVPN network with active listeners. In this case, Regular NVE forwards to all the other devices except AR Leaf 2 and AR Leaf 4.

Spine 1 and Spine 2 replicate and forward the traffic toward their local single-homed or multihomed receivers or external gateways using the usual EVPN network ingress replication local bias or DF behaviors. In this case, Figure 97 on page 1033 shows that Spine 2 is the elected DF and forwards the traffic to a multihomed receiver on an Ethernet segment that the two spine devices share.

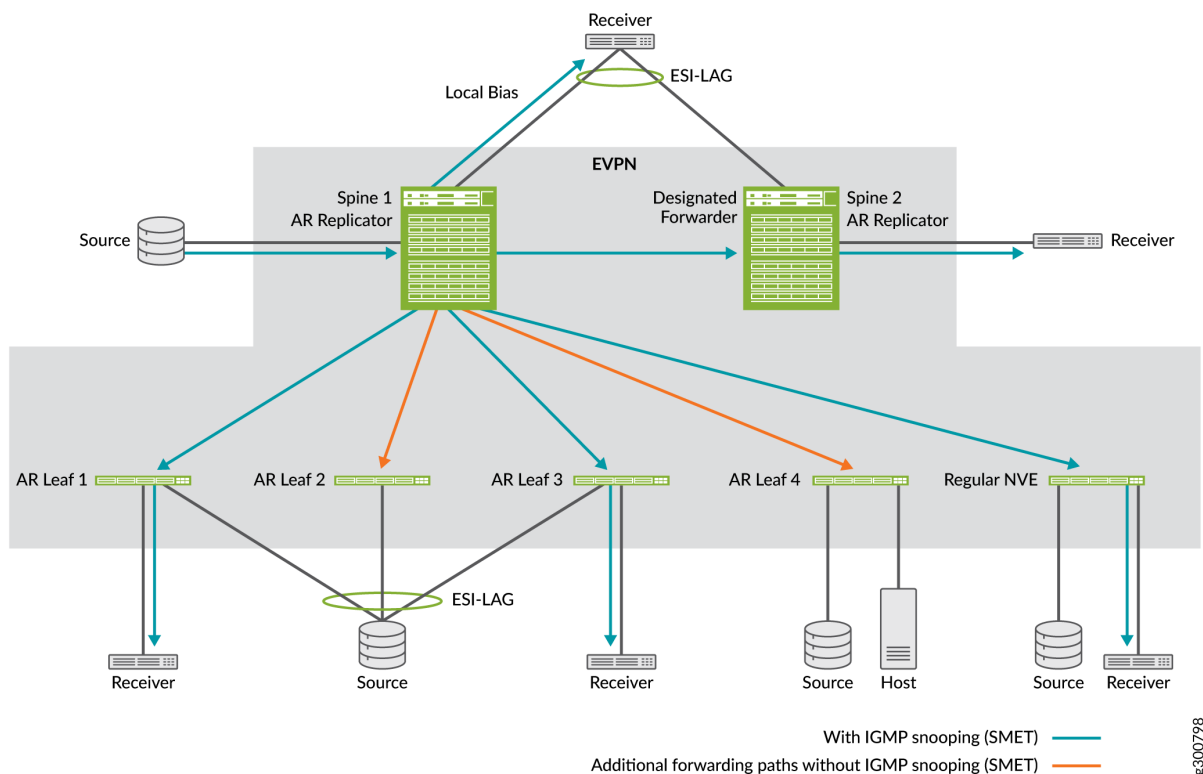
Source behind an AR Replicator Device

Figure 98 on page 1034 shows an EVPN network with multicast traffic from a source attached to an AR replicator device called Spine 1.



NOTE: QFX5130-32CD and QFX5700 devices serving as AR replicators don't support replication and forwarding from locally-attached sources or to locally-attached receivers. As a result, this use case doesn't apply when these switches act as AR replicators.

Figure 98: Multicast Source Traffic Ingress at an AR Replicator Device



In this case, forwarding behavior is the same whether you enable AR or not, because the ingress device, Spine 1, is not an AR leaf device sending traffic for replication to an AR replicator. Spine 1, although it's configured as an AR replicator device, does not act as an AR replicator. Instead, it employs the usual ingress replication forwarding rules as follows:

- Without IGMP snooping or MLD snooping, and without SMET forwarding enabled, Spine 1, the ingress device, floods the traffic to all the other devices in the EVPN network.
- With IGMP snooping or MLD snooping, and with SMET forwarding enabled, Spine 1 only forwards the traffic to other devices in the EVPN network with active listeners. In this case, Spine 1 forwards

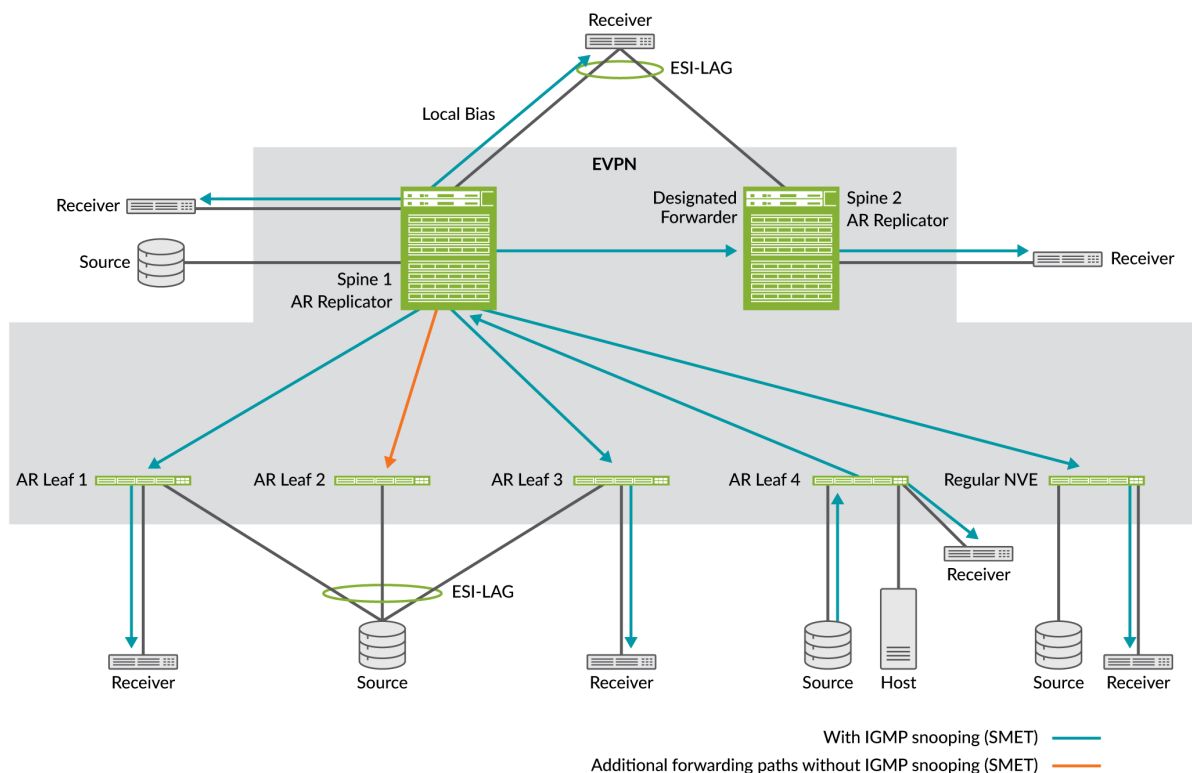
- Spine 1 also replicates the traffic toward local single-homed or multihomed receivers, or external gateways, using the usual EVPN network ingress replication local bias or DF behaviors. In this case, Spine 1 uses local bias and forwards the traffic to a multihomed receiver due to local bias rules (whether it is the DF on that Ethernet segment or not).

Spine 2 forwards the traffic it received from Spine 1 to its local receiver. Spine 2 also does a local bias check for its multihomed receiver, and skips forwarding to the multihomed receiver even though it is the DF for that Ethernet segment.

Source behind an AR Leaf Device (Single-Homed Ethernet Segment)

Figure 99 on page 1035 shows an EVPN network with multicast traffic from a source attached to the AR leaf device called AR Leaf 4. Traffic flow is the same whether the ingress AR leaf device and AR replicator are operating in extended AR mode or not, because the source isn't multihomed to any other leaf devices.

Figure 99: Source Traffic Ingress at an AR Leaf Device (Single-Homed Source)



Without IGMP snooping or MLD snooping, and without SMET forwarding enabled:

- AR Leaf 4 forwards one copy of the multicast traffic to an advertised AR replicator device, in this case Spine 1, using the AR overlay tunnel secondary IP address that you configured on the loopback interface, lo0, for Spine 1.
- AR Leaf 4 also applies local bias forwarding rules and replicates the multicast traffic to its locally attached receiver.
- Spine 1 receives the multicast traffic on the AR overlay tunnel and replicates and forwards it on behalf of Leaf 4 using the usual ingress replication overlay tunnels to all the other devices in the EVPN network, including Spine 2. Spine 1 skips forwarding the traffic back to the ingress device AR Leaf 4 (according to split horizon rules).

With IGMP snooping or MLD snooping, and with SMET forwarding enabled:

- AR replicator Spine 1 further optimizes replication by only forwarding the traffic toward other devices in the EVPN network with active listeners. In this case, Spine 1 skips forwarding to AR Leaf 2.

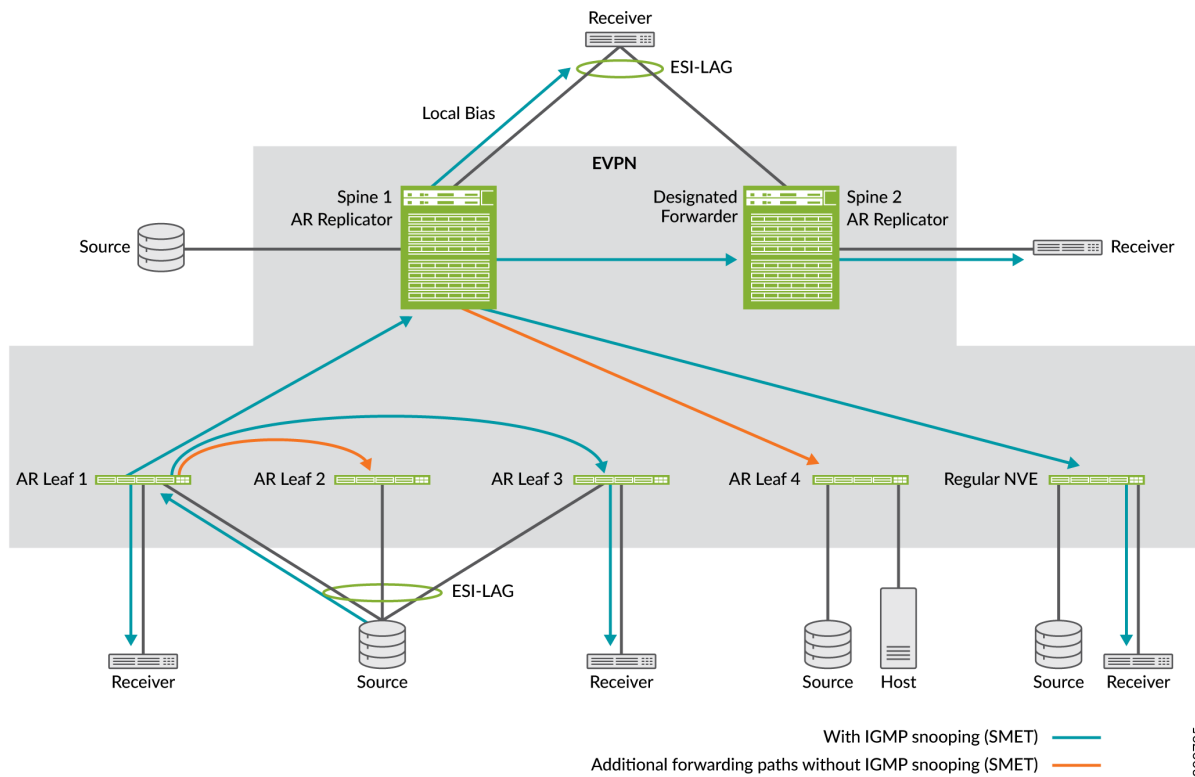
Spine 1 also replicates and forwards the traffic to its local receivers and toward multihomed receivers or external gateways using the usual EVPN network ingress replication local bias or DF behaviors. In this case, Spine 1 forwards the traffic to a local receiver. Spine 1 uses local bias to forward to a multihomed receiver (although it is not the DF for that Ethernet segment).

Spine 2 forwards the traffic it received from Spine 1 to its local receiver. Spine 2 does a local bias check for its multihomed receiver, and skips forwarding to the multihomed receiver even though it is the DF for that Ethernet segment.

Source behind an AR Leaf Device (Multihomed Ethernet Segment) with Extended AR Mode

[Figure 100 on page 1037](#) shows an EVPN network with multicast traffic from a source on a multihomed Ethernet segment operating in extended AR mode. The source is multihomed to three AR leaf devices and might send the traffic to any one of them. In this case, AR Leaf 1 is the ingress device.

Figure 100: Source Traffic Ingress at an AR Leaf Device (Multihomed Source)



Without IGMP snooping or MLD snooping, and without SMET forwarding enabled:

- AR Leaf 1 forwards one copy of the multicast traffic to one of the advertised AR replicator devices, in this case, Spine 1, using the AR overlay tunnel secondary IP address that you configured on the loopback interface, lo0, for Spine 1.
- Operating in extended AR mode:
 - AR Leaf 1 also replicates and forwards the multicast traffic to all of its multihoming peers (AR Leaf 2 and AR Leaf 3) for the source Ethernet segment.
 - Spine 1 receives the multicast traffic on the AR overlay tunnel and replicates and forwards it using the usual ingress replication overlay tunnels to the other devices in the EVPN network *except* AR Leaf 1 and its multihoming peers AR Leaf 2 and AR Leaf 3.

With IGMP snooping or MLD snooping, and with SMET forwarding enabled, the AR leaf and AR replicator devices further optimize multicast replication in extended AR mode, as follows:

- Besides sending to AR replicator Spine 1, AR Leaf 1 also replicates and forwards the multicast traffic only to its multihoming peers that have active listeners (AR Leaf 3).

- Spine 1 replicates traffic for AR Leaf 1 only to other devices in the EVPN network with active listeners. In this case, that includes only the regular NVE device and Spine 2.

Spine 1 also replicates and forwards the traffic to its local receivers and toward multihomed receivers or external gateways using the usual EVPN network ingress replication local bias or DF behaviors. In this case, Spine 1 uses local bias to forward to a multihomed receiver although it is not the DF for that Ethernet segment. Spine 2 receives the traffic from Spine 1 and forwards the traffic to its local receiver. Spine 2 also does a local bias check for its multihomed receiver, and skips forwarding to the multihomed receiver even though it is the DF for that Ethernet segment.

Configure Assisted Replication

IN THIS SECTION

- [Configure IGMP Snooping or MLD Snooping with AR | 1039](#)
- [Configure an AR Replicator Device | 1040](#)
- [Configure an AR Leaf Device | 1042](#)

Assisted replication (AR) helps optimize multicast traffic flow in EVPN networks. To enable AR, you configure devices in the EVPN network to operate as AR replicator and AR leaf devices. Available AR replicator devices that are better able to handle the processing load help perform multicast traffic replication and forwarding tasks for AR leaf devices.

You don't need to configure any options on AR replicator or AR leaf devices to accommodate devices that don't support AR. We refer to devices that don't support AR as regular network virtualization edge (NVE) devices. Regular NVE devices use the usual EVPN network and overlay tunnel ingress replication independently of AR operation within the same EVPN network. AR replicator devices also don't need to distinguish between AR leaf and regular NVE devices when forwarding multicast traffic, because AR replicator devices also use the existing ingress replication overlay tunnels for all destinations.

In general, to configure AR:

- You can enable AR on Junos OS devices in default switch EVPN instance configurations.
- (Starting in Junos OS 22.2R1) You can enable AR on Junos OS devices in MAC-VRF EVPN instance configurations with service types `vlan-based` or `vlan-aware`.
- (Starting in Junos OS Evolved 22.2R1) You can enable AR on Junos OS Evolved devices only in MAC-VRF EVPN instance configurations with service types `vlan-based` or `vlan-aware`.

Configure IGMP Snooping or MLD Snooping with AR

You can enable IGMP snooping or MLD snooping with AR to further optimize multicast forwarding in the fabric. Use the following guidelines:

- With default switch EVPN instance configurations:

Enable IGMP snooping or MLD snooping and the relevant options for each VLAN (or all VLANs) at the `[edit protocols igmp-snooping]` hierarchy.

- With MAC-VRF EVPN instance configurations:

We support IGMP snooping with MAC-VRF instance configurations. Enable IGMP snooping and the relevant options for each VLAN (or all VLANs) in the MAC-VRF routing instances at the `[edit routing-instances mac-vrf-instance-name protocols igmp-snooping]` hierarchy.

- When you configure AR and OISM roles on the same device, configure IGMP snooping for each of the OISM revenue VLANs, SBD, and the M-VLAN (if using the M-VLAN method for external multicast).

See ["Supported IGMP or MLD Versions and Group Membership Report Modes" on page 936](#) for details on any-source multicast (ASM) and source-specific multicast (SSM) support with IGMP or MLD in EVPN-VXLAN fabrics.

See for more on considerations to configure OISM with AR to support both IGMPv2 receivers and IGMPv3 receivers.



NOTE: When you enable IGMP snooping or MLD snooping with AR or OISM for multicast traffic in EVPN-VXLAN fabrics, you also automatically enable the selective multicast forwarding (SMET) optimization feature. See ["Overview of Selective Multicast Forwarding" on page 1005](#).

1. To enable IGMP snooping with IGMPv2 (or MLD snooping with MLDv1, if supported), enable IGMP (version 2 is the default). Include the `proxy` option with the `igmp-snooping` statement for each VLAN with the snooping configuration, as follows:

```
set protocols igmp interface all
set <routing-instances mac-vrf-instance-name> protocols igmp-snooping vlan vlan-name proxy
```


2. To enable IGMP snooping with IGMPv3 (or MLD snooping with MLDv2, if supported), enable IGMP version 3 (or MLD version 2) globally. Include the `evpn-ssm-reports-only` option with the snooping configuration for each VLAN, as follows:

```
set protocols igmp interface all version 3
set <routing-instances mac-vrf-instance-name> protocols igmp-snooping vlan vlan-name evpn-ssm-reports-only
```



NOTE: QFX5130-32CD and QFX5700 switches can be OISM border leaf devices that support external multicast sources or receivers using a multicast L3 IRB interface that you don't extend in the EVPN instance. We call this interface a *non-EVPN IRB interface for external multicast*. When you configure OISM (with or without an AR leaf role) on these border leaf devices, you don't need to include the `evpn-ssm-reports-only` option for the non-EVPN IRB interfaces when you configure IGMP snooping.

Configure an AR Replicator Device

Devices you configure as AR replicator devices advertise their AR capabilities and AR IP address to the EVPN network. The AR IP address is a loopback interface address you configure on the AR replicator device. AR leaf devices receive these advertisements to learn about available AR replicators to which the leaf devices can offload multicast replication and forwarding tasks. AR leaf devices automatically load-balance among multiple available AR replicator devices.

If you want to enable AR and OISM together in your fabric, you can assign the AR replicator role in standalone or collocated mode:

- In standalone mode, you assign the AR replicator role to devices that don't act as OISM border leaf devices. These devices are usually the lean spine devices in the ERB overlay fabrics that support OISM.



NOTE: On QFX5130-32CD and QFX5700 switches, we only support standalone mode.

- In collocated mode, you can assign the AR replicator role on OISM border leaf devices.

In either case, the AR replicator device needs the common OISM tenant VRF instances, revenue VLANs, SBD, and corresponding IRB interfaces to properly forward the multicast traffic across the L3 VRFs. You'll need to configure these items as part of the AR configuration on the standalone AR replicator device. See step 4 below.) In collocated mode, you configure these items as part of the OISM configuration as well as other OISM role-specific configuration.

To configure an AR replicator device:

1. Configure the device loopback interface lo0 with a secondary IP address specifically for AR functions. The AR replicator advertises this IP address to the network in the EVPN Type 3 AR tunnel routes. (See ["AR Route Advertisements" on page 1016](#) for details.)

```
[edit]
user@ar-replicator# set interfaces lo0 unit 0 family inet address AR-IP-address
```

2. Configure the AR replicator device using the loopback interface you configured in Step 1.

In the default switch instance on Junos OS devices:

```
[edit]
user@ar-replicator# set protocols evpn assisted-replication replicator inet AR-IP-address
```

Or in a MAC-VRF EVPN instance on supported Junos OS or Junos OS Evolved devices:

```
[edit]
user@ar-replicator# set routing-instances mac-vrf-instance-name protocols evpn assisted-
replication replicator inet AR-IP-address
```

3. Set the type of IP address the AR replicator uses as the ingress source address in the overlay tunnel encapsulation for the replicated traffic.

You set this option using the `vxlan-encapsulation-source-ip` statement at the `[edit protocols evpn assisted-replication replicator]` hierarchy for the default switch instance, or the `[edit routing-instances mac-vrf-instance-name protocols evpn assisted-replication replicator]` hierarchy for MAC-VRF instances.

The default value for `vxlan-encapsulation-source-ip` is `retain-incoming-source-ip`. (See `replicator` .) With AR replicators that are not capable of retaining the source IP address, such as Junos OS or Junos OS Evolved QFX Series devices in an EVPN-VXLAN network, you must set this option to `ingress-replication-ip` instead of the default value.



NOTE: All Junos OS or Junos OS Evolved devices that currently support AR require this setting in the AR replicator role configuration.

When you set `ingress-replication-ip`, the AR replicator advertises that it is not capable of retaining the source IP address of the originating AR leaf device when replicating and forwarding the traffic to the regular ingress replication overlay tunnels. This means it supports only extended AR mode. (See ["Extended AR Mode for Multihomed Ethernet Segments" on page 1017](#) for details.) Other AR

devices in the EVPN network receive these advertisements. As a result, those devices also use extended AR mode for compatible operation.

We reserve the `retain-incoming-source-ip` setting for future compatibility with EVPN network devices and overlays that are capable of retaining the incoming source IP address in replicated traffic.

Set the VXLAN encapsulation source IP address type to `ingress-replication-ip` in the default switch instance on Junos OS devices:

```
[edit]
user@ar-replicator# set protocols evpn assisted-replication replicator vxlan-encapsulation-
source-ip ingress-replication-ip
```

Or in a MAC-VRF EVPN instance on supported Junos OS or Junos OS Evolved devices:

```
[edit]
user@ar-replicator# set routing-instances mac-vrf-instance-name protocols evpn assisted-
replication replicator vxlan-encapsulation-source-ip ingress-replication-ip
```

4. (Only for AR and OISM integration with AR replicator in standalone mode) Configure the AR replicator device with the L2 and L3 OISM configuration elements common to the OISM border leaf and server leaf devices. We describe these steps in ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices" on page 1158](#).

This step ensures that the AR replicator can install the correct L2 multicast states to properly forward the multicast traffic across the L3 VRFs.

Configure an AR Leaf Device

Devices you configure as AR leaf devices receive traffic from multicast sources and offload the replication and forwarding to an AR replicator device. AR replicator devices advertise their AR capabilities and AR IP address, and AR leaf devices automatically load-balance among the available AR replicators.

If you want to enable AR and OISM together in your fabric, you can configure the AR leaf role on any devices configured in the OISM border leaf or server leaf roles in the fabric. For details on OISM leaf device configuration, see:

- ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices" on page 1158](#)
- ["Configure Server Leaf Device OISM Elements" on page 1169](#)
- ["Configure Border Leaf Device OISM Elements with M-VLAN IRB Method \(Symmetric Bridge Domains Model Only\)" on page 1170](#)

- ["Configure Border Leaf Device OISM Elements with Classic L3 Interface Method" on page 1175](#)
- ["Configure Border Leaf Device OISM Elements with Non-EVPN IRB Method" on page 1177](#)

To configure an AR leaf device:

1. Configure the device into the AR leaf role.

In the default switch instance on Junos OS devices:

```
[edit]user@ar-leaf# set protocols evpn assisted-replication leaf
```

Or in a MAC-VRF EVPN instance on supported Junos OS or Junos OS Evolved devices:

```
[edit]user@ar-leaf# set routing-instances mac-vrf-instance-name protocols evpn assisted-replication leaf
```

2. (Optional) By default, AR leaf devices delay for 10 seconds after receiving an AR replicator advertisement before starting to send traffic to that AR replicator device. If needed, you can adjust the delay (in seconds) to make sure AR replicator devices have fully learned the current EVPN state from the network. You might need this delay in cases such as after an AR replicator goes down and comes up again.

Configure the AR activation delay in the default switch instance on Junos OS devices:

```
[edit]
user@ar-leaf# set protocols evpn assisted-replication leaf replicator-activation-delay seconds
```

Or in a MAC-VRF EVPN instance on supported Junos OS or Junos OS Evolved devices:

```
[edit]
user@ar-leaf# set routing-instances mac-vrf-instance-name protocols evpn assisted-replication leaf replicator-activation-delay seconds
```

Verify Assisted Replication Setup and Operation

Several commands help verify that AR replicator devices have advertised their AR IP address and capabilities. Some commands verify that AR leaf devices have learned how to reach available AR replicator devices and EVPN multihoming peers to operate in extended AR mode.

1. View AR information received from AR replicator EVPN Type 3 (IMET) route advertisements.

```
user@device> show route table bgp.evpn.0 match-prefix 3:* extensive
```

This command displays the Type 3 routes for an EVPN instance to the available AR replicator devices (AR overlay tunnel next hops), including Type ASSISTED-REPLICATION and Role AR-REPLICATOR in the PMSI section of the output. The following (truncated) example shows the EVPN Type 3 route for an available AR replicator device with AR IP address 192.168.102.1 seen on an AR leaf device. The output also includes the advertised EVPN multicast extended community flags that show the AR operating mode is extended AR mode:

```
user@ar-leaf> show route table bgp.evpn.0 match-prefix 3:* extensive
...
3:192.168.2.1:10000::100001::192.168.102.1/248 IM (2 entries, 0 announced)
    *BGP      Preference: 170/-101
              Route Distinguisher: 192.168.2.1:10000
              PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.1 Role AR-
REPLICATOR
              Next hop type: Indirect, Next hop index: 0
              Address: 0x1a6b08f0
              Next-hop reference count: 4000
              Source: 192.168.2.1
              Protocol next hop: 192.168.102.1
              Indirect next hop: 0x2 no-forward INH Session ID: 0x0
              State: <Active Int Ext>
              Local AS: 4210000001 Peer AS: 4210000001
              Age: 4:58:08      Metric2: 0
              Validation State: unverified
              Task: BGP_4210000001.192.168.2.1
              AS path: I
              Communities: target:32897:268535457 encapsulation:vxlan(0x8) evpn-mcast-
flags:0x4:extended-MH-AR
              Import Accepted
              Localpref: 100
              Router ID: 192.168.2.1
              Secondary Tables: default-switch.evpn.0
              Indirect next hops: 1
                  Protocol next hop: 192.168.102.1
                  Indirect next hop: 0x2 no-forward INH Session ID: 0x0
```

```

Indirect path forwarding next hops: 1
  Next hop type: Router
  Next hop: 172.16.109.0 via ae3.0
  Session Id: 0x0
  192.168.102.1/32 Originating RIB: inet.0
  Node path count: 1
  Forwarding nexthops: 1
    Nexthop: 172.16.109.0 via ae3.0
    Session Id: 0\
...

```

2. Check the available AR replicators and AR mode in which the replicator devices are operating.

```
user@device> show evpn multicast-snooping assisted-replication replicators
```

You can run this command on an AR leaf or AR replicator device. The output shows the available AR replicator devices and their AR operating mode. The output also shows the next hops and overlay tunnel interfaces used to reach each AR replicator device from the device where you run the command. For example:

```

user@ar-leaf> show evpn multicast-snooping assisted-replication replicators
Instance: default-switch
AR Role: AR Leaf

VN Identifier: 100
Operational Mode: Extended AR

```

Replicator IP	Nexthop Index	Interface	Mode
192.168.102.1	1772	vtep.32770	Extended AR
192.168.102.2	1797	vtep.32772	Extended AR

3. (Available when you enable IGMP snooping or MLD snooping) View AR leaf device overlay tunnel next hops to load-balance among the advertised AR replicator devices.

```
user@ar-leaf> show evpn multicast-snooping assisted-replication next-hops
```

The following sample command shows the load-balancing next hops toward the advertised AR replicators that the AR leaf device detected in [Step 2](#).

Also, when an AR leaf device load-balances by assigning an AR replicator device to each VLAN or VNI, this command tags that AR replicator as the (Designated Node) in the output (which the example below also shows). AR leaf devices that load-balance based on active traffic flow don't display this tag. (See ["AR Leaf Device Load Balancing with Multiple Replicators" on page 1018.](#))

```
user@ar-leaf> show evpn multicast-snooping assisted-replication next-hops
Instance: default-switch
```

```
VN Identifier: 100
```

```
Load Balance Nexthop Index: 131091
```

```
Load balance to:
```

Nexthop Index	Interface	AR IP
1772	vtep.32770	192.168.102.1 (Designated Node)
1797	vtep.32772	192.168.102.2

4. (Available when you enable IGMP snooping or MLD snooping) View AR leaf device multihomed peers. Include the extensive option to also see the multihomed Ethernet segments shared with peers.

```
user@ar-leaf> show evpn multicast-snooping assisted-replication multihomed-peers <extensive>
```

For example:

```
user@ar-leaf> show evpn multicast-snooping assisted-replication multihomed-peers extensive
```

```
Instance: default-switch
```

```
Neighbor address: 192.168.0.2
```

```
Nexthop Index: 1746
```

```
Interface: vtep.32768
```

```
Local Multi-homed ESIs
```

```
00:11:22:33:44:55:66:77:88:99
```

```
00:11:22:33:44:55:66:77:88:aa
```

5. (Available when you enable IGMP snooping or MLD snooping) View IGMP snooping or MLD snooping core next hops on the AR replicator for each multicast group and VLAN or VNI. You can follow the core next hops to the corresponding AR load-balancing and multihoming peer device next hops. Use the following commands:

- show evpn igmp-snooping proxy extensive
- show evpn mld-snooping proxy extensive

- `show evpn multicast-snooping next-hops detail`

For example, in the output from the following commands, you can see the Downstream interface Addr field for next-hop ID 131092 shows the following:

- The load-balancing next-hop index 131091, which you also see with the `show evpn multicast-snooping assisted-replication next-hops` command in [Step 3](#).
- The multihoming peer device interface and next hop index vtep.32768-(1746), which you also see with the `show evpn multicast-snooping assisted-replication next-hops` command in [Step 4](#).

```
user@ar-replicator> show evpn igmp-snooping proxy extensive
```

```
Instance: default-switch
```

```
VN Identifier: 100
```

Group	Source	Local	Remote	Corenh
233.252.0.1	0.0.0.0	1	5	131092

```
VN Identifier: 200
```

Group	Source	Local	Remote	Corenh
233.252.0.1	0.0.0.0	1	5	131092

```
user@ar-replicator> show evpn multicast-snooping next-hops detail
```

```
Family: INET
```

ID	RefCount	KRefCount	Downstream interface Addr
131092	6	2	131091 vtep.32678-(1746) Flags 0x2100 type 0x18 members 0/0/0/5/0 Address 0xc54ba44

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
23.4R2-EVO	Starting in Junos OS Evolved Release 23.4R2, QFX5130-32CD and QFX5700 switches running as AR leaf devices support deterministically steering AR multicast flows toward configured preferred AR replicators for load balancing.
23.1R1-EVO	Starting in Junos OS Evolved Release 23.1R1, QFX5130-32CD and QFX5700 switches support AR and OISM for IPv6 multicast traffic with MLDv1, MLDv2, and MLD snooping.

22.2R1-EVO	Starting in Junos OS Evolved Release 22.2R1, you can enable AR on QFX5130-32CD and QFX5700 switches in MAC-VRF EVPN instance configurations to optimize unknown unicast and broadcast traffic replication in the fabric. You can configure these switches as AR replicators or AR leaf devices. As AR replicators, these switches don't support replication and forwarding for locally-attached sources and receivers. Also, these devices don't support AR as border spine devices with seamless VXLAN to VXLAN stitching.
22.2R1-EVO	Starting in Junos OS Evolved Release 22.2R1, QFX5130-32CD and QFX5700 switches support AR with OISM in MAC-VRF EVPN instance configurations to optimize forwarding multicast traffic. You can configure these switches as AR replicators or AR leaf devices. The AR replicator role operates only in standalone mode, which means the AR replicator role can't be collocated with the OISM border leaf role on the same device. As AR replicators, these switches don't support replication and forwarding for locally-attached sources and receivers. We support AR and OISM with IGMPv2 or IGMPv3, and IGMP snooping.
22.2R1	Starting in Junos OS Release 22.2R1, you can enable AR with optimized intersubnet multicast (OISM) in the default switch instance or MAC-VRF EVPN instances on EX4650, QFX5110, QFX5120, QFX10002 (except QFX10002-60C), QFX10008, and QFX10016 switches. The AR replicator role can be collocated with the OISM border leaf role on the same device, or you can configure the AR replicator role in standalone mode on a lean spine device in the fabric. (Only QFX10000 line switches can be AR replicators.) We support AR and OISM with IGMPv2 or IGMPv3, and IGMP snooping.
22.2R1	Starting in Junos OS Release 22.2R1, you can enable AR on EX4650, QFX5110, QFX5120, QFX10002, QFX10008, and QFX10016 switches in MAC-VRF EVPN instance configurations instead of in the default switch instance configuration. All of the listed devices can be AR leaf devices. QFX10002, QFX10008, and QFX10016 switches can be AR replicators.
20.4R1	Starting in Junos OS Release 20.4R1, you can enable AR with MLDv1 or MLDv2 and MLD snooping on QFX5110, QFX5120, QFX10002, QFX10008, and QFX10016 switches in the default switch instance.
18.4R2	Starting in Junos OS Releases 18.4R2 and 19.4R1, you can enable AR on QFX5110, QFX5120, QFX10002, QFX10008, and QFX10016 switches in the default switch instance with IGMPv2 or IGMPv3 and IGMP snooping. You can configure any of these switches as AR leaf devices. You can configure QFX10002, QFX10008, and QFX10016 switches as AR replicators.

RELATED DOCUMENTATION

[Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment | 934](#)

[Overview of Selective Multicast Forwarding | 1005](#)

assisted-replication

[igmp-snooping](#)

[mld-snooping](#)

Optimized Intersubnet Multicast in EVPN Networks

SUMMARY

Enable intersubnet multicast (OISM) to optimize multicast traffic routing and forwarding in an EVPN edge-routed bridging (ERB) overlay fabric. OISM avoids multicast data flooding to efficiently support scaled multicast environments. Also, with OISM your network can support multicast traffic flow among devices inside and outside of the EVPN fabric.

IN THIS SECTION

- [Overview of OISM | 1050](#)
- [Overview of Enhanced OISM | 1057](#)
- [OISM Components | 1061](#)
- [How OISM Works | 1088](#)
- [How Enhanced OISM Works | 1112](#)
- [Considerations for OISM Configurations | 1117](#)
- [Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices | 1158](#)
- [Configure Server Leaf Device OISM Elements | 1169](#)
- [Configure Border Leaf Device OISM Elements with M-VLAN IRB Method \(Symmetric Bridge Domains Model Only\) | 1170](#)
- [Configure Border Leaf Device OISM Elements with Classic L3 Interface Method | 1175](#)
- [Configure Border Leaf Device OISM Elements with Non-EVPN IRB Method | 1177](#)
- [CLI Commands to Verify the OISM Configuration | 1180](#)

Overview of OISM

IN THIS SECTION

- Benefits of OISM | 1051
- OISM Support in EVPN Instances | 1051
- OISM Support with Tenant L3 VRF Instances or the Default L3 Routing Instance | 1052
- OISM with Multicast Protocols and Other Multicast Optimizations in EVPN Fabrics | 1053
- Platform-Specific OISM Behavior with Regular OISM | 1056

Traditional methods to support multicast traffic use ingress replication and flood multicast packets into the network to reach any interested listeners. Those methods don't scale well and have latency issues when your network has large multicast flows. Also, configuring the network to properly and efficiently handle multicast traffic from sources and to receivers outside of your network is complex.

Optimized intersubnet multicast (OISM) is a multicast traffic optimization feature that operates at Layer 2 (L2) and Layer 3 (L3) in EVPN-VXLAN edge-routed bridging (ERB) overlay fabrics. OISM solves many of the issues inherent in other multicast methods. The OISM design is based on the IETF draft specification <https://datatracker.ietf.org/doc/html/draft-ietf-bess-evpn-irb-mcast>.

We refer to our original OISM implementation as *regular OISM*. Regular OISM uses a symmetric bridge domains OISM model that requires you to configure all revenue VLANs (also called tenant VLANs) in the network on all OISM leaf devices.

Starting in Junos OS Release 23.4R1, we also support an enhanced version of OISM. Enhanced OISM uses an asymmetric bridge domains OISM model with which you don't need to configure all revenue VLANs in the network on all OISM devices. On each device, you can configure only the revenue VLANs that the device hosts. To support the asymmetric bridge domains model, enhanced OISM has some operational differences from the symmetric bridge domains model and small configuration differences. The differences are called out throughout this document.

You can apply OISM configuration and operation to multicast traffic but not to broadcast or unknown unicast traffic.

In EVPN ERB overlay fabric designs, the leaf devices in the fabric route traffic between tenant bridge domains (that is, between VLANs). When you enable OISM, the leaf devices route intersubnet multicast traffic locally through IRB interfaces using the control plane multicast states. With local routing between VLANs, the receiver IRB interface doesn't send the routed multicast traffic out into the EVPN core. The local routing model helps minimize the traffic load within the EVPN core. It also avoids traffic hairpinning.

OISM leaf devices also selectively forward traffic into the EVPN core only toward other EVPN devices with interested receivers. Selective forwarding further improves multicast traffic performance in the EVPN fabric.

With OISM enabled, ERB overlay fabrics can efficiently and effectively support multicast traffic flow between devices inside and outside the EVPN fabric. Without OISM, fabric designers must use the centrally routed bridging (CRB) overlay model to support multicast with external sources or receivers. OISM border leaf devices support different methods to route traffic to and from an external PIM domain. These methods use either integrated routing and bridging (IRB) interfaces or Layer 3 (L3) interfaces. OISM also employs a supplemental bridge domain (SBD) inside the fabric as follows:

- The SBD has a different VLAN ID from any of the revenue VLANs.
- Border leaf devices use the SBD to carry the traffic from external sources toward receivers within the EVPN fabric.
- In enhanced OISM mode, server leaf devices use the SBD to carry traffic from internal sources to other server leaf devices in the EVPN fabric that are not multihoming peers. Enhanced mode leaf devices use the source VLAN only to send multicast traffic to their multihoming peer leaf devices.

Benefits of OISM

- Enables EVPN-VXLAN fabrics with the ERB overlay model to support multicast traffic with sources and receivers outside of the fabric.
- Minimizes multicast control packets and replicated data packets in the EVPN fabric core to optimize fabric multicast performance in scaled designs.
- With enhanced OISM mode, you can further support scaled network designs with leaf devices that host a large number of diverse VLANs (on each leaf device, you need to configure only the VLANs that the device hosts).

OISM Support in EVPN Instances

We support OISM in EVPN-VXLAN fabrics in the following types of EVPN instances:

- EVPN in the default switch instance (on platforms that support a default switch instance)
- MAC-VRF EVPN routing instances with `vlan-aware` and `vlan-based` service types only (see ["MAC-VRF Routing Instance Type Overview" on page 38](#))



On Junos OS Evolved devices, we support EVPN-VXLAN using EVPN configurations with MAC-VRF instances only, and not in the default switch instance. As a result, on these devices we support OISM only in MAC-VRF EVPN instances.

OISM Support with Tenant L3 VRF Instances or the Default L3 Routing Instance

You can configure OISM with the following types of tenant L3 routing instances:

- `instance-type vrf`, which are named virtual routing and forwarding (VRF) instances supported by all platforms that support OISM. OISM configurations usually include multiple tenant VRF instances.
- The default L3 routing instance on the device, supported only on some platforms. See Feature Explorer for currently supported platforms.

We support using the default L3 routing instance with:

- Regular OISM mode on server leaf and border leaf devices, including border leaf devices acting in the PIM EVPN gateway (PEG) role (see [Table 57 on page 1062](#)).
- IPv4 underlay peering in the EVPN network.
- IPv4 multicast data traffic with IGMPv1, IGMPv2, and IGMP snooping.
- PEG designated forwarder (DF) election options (see ["PEG DF Election" on page 1129](#)).

Any use cases or behaviors that are not supported with OISM in a configured VRF instance on a particular platform are likewise not supported in the default L3 routing instance on that platform. For example, if we don't support assisted replication (AR) with OISM in a configured VRF on a platform, then we also don't support AR with OISM in the default L3 routing instance on that platform.

Throughout this OISM documentation, we describe configuring particular elements in an L3 VRF instance on the EVPN and OISM devices. If your configuration uses the default L3 routing instance instead of VRF instances, then you configure those elements at the global level instead of at the `[edit routing-instances name ...]` hierarchy level. For example, substitute the configuration statements at this hierarchy level:

```
[edit routing-instances L3VRF-1 protocols evpn ...]
```

with the same statements at the global hierarchy level:

```
[edit protocols evpn ...]
```

In OISM configurations that use the default L3 routing instance, we also require that you add a global routing policy configuration to match overlay VXLAN interface next hops and avoid installing those routes for underlay reachability. With EVPN-VXLAN configurations, the VXLAN tunnels are resolved only in the underlay. The device overlay peering uses the underlay to establish reachability among the EVPN devices, which is a valid use of underlay routes. However, the device's underlay peering might use overlay IRB routes for reachability in default L3 routing instance configurations, which can cause routing loops in the network.

In this case, to prevent the underlay peering from using overlay routes for underlay reachability on OISM devices, you must include defining and applying a routing policy with the `install-nexthop except overlay-vxlan-interfaces` policy statement action, as follows:

```
set policy-options policy-statement policy-name term 1 then install-nexthop strict
set policy-options policy-statement policy-name term 1 then install-nexthop except overlay-vxlan-interfaces
set routing-options forwarding-table export policy-name
```

OISM with Multicast Protocols and Other Multicast Optimizations in EVPN Fabrics

OISM works with the following multicast protocols and other EVPN multicast optimization features.

Multicast Protocols Supported with OISM

- IGMPv2, IGMPv3, and IGMP snooping for IPv4 multicast traffic
- MLDv1, MLDv2, and MLD snooping for IPv6 multicast traffic
- PIM, which facilitates both local routing and external multicast traffic routing

See [Feature Explorer](#) for specific platform and release support.

OISM supports IGMP snooping with both IGMPv2 and IGMPv3 on the same device at the same time only under certain configuration constraints. Similarly, OISM supports MLD snooping with both MLDv1 and MLDv2 at the same time under the same configuration constraints. See ["IGMPv2 and IGMPv3 \(or MLDv1 and MLDv2\) in the Same EVPN-VXLAN Fabric" on page 1117](#) for details.

Also see ["Supported IGMP or MLD Versions and Group Membership Report Modes" on page 936](#) for information on IGMP or MLD any-source multicast (ASM) and source-specific multicast (SSM) mode support in EVPN-VXLAN fabrics.

Other Multicast Optimization Features That Work with OISM

OISM works with these other multicast optimization features:

- IGMP snooping or MLD snooping (some platforms) on the access side on the leaf devices.

With IGMP snooping or MLD snooping enabled, a leaf device that receives multicast traffic forwards it only toward other devices with interested receivers.

- Multihoming support in an Ethernet segment (ES) using EVPN Type 7 (Join Sync) and Type 8 (Leave Sync) routes.

EVPN fabric devices advertise these route types to synchronize the multicast state among EVPN devices that are multihoming peers.



NOTE: ACX Series OISM leaf devices can be multihoming peer PE devices only with ACX Series devices.

- Selective multicast Ethernet tag (SMET) forwarding in the EVPN fabric core using EVPN Type 6 routes.

EVPN devices use Type 6 routes to limit forwarding within the EVPN core only to receivers interested in receiving traffic for a multicast group. You can use OISM to make this optimization work in EVPN ERB overlay fabrics. When you configure IGMP or MLD snooping, the fabric enables SMET forwarding with OISM automatically.

- Assisted replication (AR) (some platforms and releases).

[Table 55 on page 1054](#) lists how you can integrate AR into a fabric running OISM using the platforms and releases that support the different AR and OISM device roles. We support AR only with regular OISM on any platforms.

Table 55: AR Integrated with Regular OISM—Platform-Specific Supported Roles and Behavior

Platforms	Starting Release	Supported Roles and Behavior
EX4650 QFX5110 QFX5120 QFX10002 (except QFX10002-60C) QFX10008 QFX10016	Junos OS Release 22.2R1	You can configure the AR leaf role on any of these devices that are also acting as OISM border leaf or server leaf devices.

Table 55: AR Integrated with Regular OISM—Platform-Specific Supported Roles and Behavior
(Continued)

Platforms	Starting Release	Supported Roles and Behavior
QFX10002 (except QFX10002-60C) QFX10008 QFX10016	Junos OS Release 22.2R1	<p>You can configure any of these switches as AR replicators, in either of the following AR replicator modes:</p> <ul style="list-style-type: none"> • Collocated mode: The device acts as both an AR replicator device and an OISM border leaf device. • Standalone mode: The device is an AR replicator but isn't also an OISM border leaf or server leaf device.
QFX5130-32CD QFX5700	Junos OS Evolved Release 22.2R1	<p>You can configure the AR leaf role on any of these devices that are also acting as OISM border leaf or server leaf devices.</p> <p>You can configure these devices as AR replicators with OISM in standalone mode only. In standalone mode, the AR replicator device doesn't also operate as an OISM border leaf or server leaf.</p>

Table 55: AR Integrated with Regular OISM—Platform-Specific Supported Roles and Behavior
(Continued)

Platforms	Starting Release	Supported Roles and Behavior
EX4400	Junos OS Release 24.4R1	<p>You can configure the AR leaf role on any of these devices that are also acting as OISM border leaf or server leaf devices.</p> <p>These devices don't support the AR replicator role, and support the AR leaf role with other devices configured as standalone AR replicators only.</p> <p>NOTE: We support AR with OISM on these devices only with VLAN-based and VLAN-aware MAC-VRF EVPN instances.</p>



NOTE: ACX Series and PTX Series routers don't support AR with OISM as AR replicator or AR leaf devices.

See these references for more on using AR and OISM together:

- How AR works and how to configure AR—["Assisted Replication Multicast Optimization in EVPN Networks" on page 1010.](#)
- How AR integrates with OISM—["AR with Optimized Intersubnet Multicast \(OISM\)" on page 1029.](#)
- Use case illustrations of AR and OISM working together:
 - ["AR and OISM with an Internal Multicast Source" on page 1106](#)
 - ["AR and OISM with an External Multicast Source" on page 1110](#)

Platform-Specific OISM Behavior with Regular OISM

Use the following table to review platform-specific behaviors when running regular OISM for your platforms.

Table 56: Platform-Specific Regular OISM Behavior

Platform	Difference
PTX10008 with the PTX10K-LC1301-36DD (Express 5) and PTX10K-LC1201-36CD (Express 4) line cards	<p>The device has a limitation in reporting multicast packet statistics when the device operates in interop chassis mode (both Express 5 and Express 4 line cards interoperating on the device), because Express 5 line cards don't support multicast snooping route counters.</p> <p>The multicast snooping route counters record the number of packets that use a snooping route toward the destination. Express 4 line cards update the snooping route counters, but Express 5 line cards don't update those counters. As a result, on devices running in interop mode, the packet statistics reported in the <code>show multicast snooping route extensive</code> CLI command won't match the actual packet count for snooping routes.</p>

Overview of Enhanced OISM

IN THIS SECTION

- [Enhanced OISM Support | 1058](#)
- [How to Enable Enhanced OISM | 1058](#)
- [When to Use Enhanced OISM | 1059](#)
- [Summary of Enhanced OISM Differences | 1060](#)

Enhanced OISM doesn't require you to configure all revenue bridge domains (VLANs) in the network on all OISM devices. On each device, you can configure only the revenue VLANs the device hosts. As a result, we describe this mode as having an *asymmetric bridge domains (VLANs)* model compared to the regular OISM mode where you must configure the revenue VLANs symmetrically on all leaf devices.

However, in enhanced OISM mode, you must still configure revenue VLANs symmetrically on the OISM leaf devices that share any Ethernet segments. In other words, you must configure the same revenue VLANs on OISM leaf devices that are multihoming peers for an attached multihomed host or multihomed customer edge (CE) device.

Enhanced OISM mode enables OISM to scale well when your network has leaf devices that host larger numbers of different VLANs per device.

Enhanced OISM Support

We support enhanced OISM with:

- IGMPv2, IGMPv3, and IGMP snooping.
- MLDv1, MLDv2, and MLD snooping (some platforms).



NOTE: ACX Series and PTX Series routers don't support enhanced OISM with MLD and MLD snooping for IPv6 multicast traffic.

- MAC-VRF EVPN instance type only. See ["MAC-VRF Routing Instance Type Overview" on page 38](#).
- EVPN-VXLAN configurations with an IPv6 underlay for IPv4 or IPv6 multicast traffic (some platforms). See ["Enhanced OISM with an EVPN-VXLAN IPv6 Underlay Configuration " on page 1140](#).
- Seamless data center interconnect (DCI) stitching between EVPN-VXLAN data centers configured with IPv4 or IPv6 underlay peering (some platforms). See ["EVPN-VXLAN DCI Multicast with Enhanced OISM" on page 1188](#).
- Group based policy (GBP). We support GBP unicast flows and enhanced OISM flows seamlessly together across the VXLAN tunnels. You must configure GBP with the `vxlan-gbp-mc-profile` GBP unified forwarding table (UFT) profile setting at the `[edit chassis forwarding-options]` hierarchy level.

See ["GBP Profiles" on page 1279](#) for more on GBP profiles and configurations.



NOTE: We don't assign GBP tags to multicast traffic. Only unicast traffic carries GBP tags in the VXLAN headers.

We don't support enhanced OISM with AR.

See [Feature Explorer](#) for specific enhanced OISM platform and release support.

How to Enable Enhanced OISM

You enable enhanced OISM using the `enhanced-oism` option at the `[edit forwarding-options multicast-replication evpn irb]` hierarchy level. You use this option instead of the regular OISM mode `oism` option at the same hierarchy level. The `enhanced-oism` and `oism` options are mutually exclusive.

Besides the difference in configuring VLANs on the leaf devices and setting the OISM mode to use, the OISM components and configuration elements are the same for enhanced OISM as for regular OISM mode. However, this mode has some operational differences and small configuration differences to support the asymmetric bridge domains model. As a result, you must use the same OISM mode on all OISM devices in the network.

See:

- ["Overview of OISM" on page 1050](#) for a brief introduction to OISM support.
- ["OISM Components" on page 1061](#) for descriptions of all of the components and configuration elements involved in OISM operation.

When to Use Enhanced OISM

You can use enhanced OISM if all OISM devices in the network support this OISM mode. In that case, you might want to use enhanced OISM when:

- Your network has a large number of revenue bridge domains (VLANs), and resources might be strained on some devices to configure all the VLANs there.
- Your network has a large number of disjointed bridge domains (VLANs) in the network (different devices host different sets of VLANs).
- On OISM devices in your network, you don't have policies configured that are based on the source MAC address of the packets. If you do have source MAC address policies, use regular OISM in your network instead.
- Your network uses seamless DCI stitching to span more than one EVPN-VXLAN data center, and you want to optimize multicast traffic flow between the data centers instead of always flooding multicast traffic across the interconnection. The DCI gateway devices and OISM leaf devices in both networks must support seamless DCI multicast with enhanced OISM. (See ["EVPN-VXLAN DCI Multicast with Enhanced OISM" on page 1188](#).)

You might want to use regular OISM instead of enhanced OISM if your network needs to pass multicast packets with stringent requirements for decrementing the time-to-live (TTL) field. The enhanced OISM model inherently has a limitation where packets with TTL=1 will not reach receivers on devices that are not multihoming peers of the source device. Regular OISM forwards source traffic on the source VLAN and doesn't decrement the TTL value for destinations on the same VLAN. Alternatively, if you want to use enhanced OISM but avoid this limitation, we have a solution that enables enhanced OISM devices to forward multicast traffic on the source VLAN like regular OISM devices do. You can enable this solution for particular multicast groups (or groups and sources) using routing policies and a configuration option. See ["Enhanced OISM Exception Policy to Forward on Source VLAN Instead of SBD for Packets with TTL=1" on page 1141](#) for details.

Summary of Enhanced OISM Differences

Where applicable, the sections throughout this document describe any operational or configuration differences when you use enhanced OISM.

We summarize the main differences with enhanced OISM operation and configuration here.

East-West Traffic from Internal Sources

The ingress leaf devices forward east-west multicast source traffic on the source VLAN to their multihoming peer leaf devices with which they share at least one Ethernet segment. For all other OISM leaf devices, they route the source traffic only on the SBD (even if those other devices host the source VLAN). Then each leaf device locally routes the traffic from the SBD to the destination VLAN.

This operation differs from the regular OISM mode, which sends multicast traffic from internal sources only on the source VLAN. Then each leaf device locally forwards the traffic on the source VLAN or routes the traffic from the source VLAN to the destination VLAN.

North-South Traffic from Internal Sources Toward External Receivers

The ingress leaf devices generate EVPN Type 10 Selective P-router Multicast Service Interface (S-PMSI) Auto-Discovery (A-D) routes for internal multicast (S,G) sources and groups.

The OISM border leaf devices act as PIM EVPN gateway (PEG) devices to connect to external multicast sources and receivers. The PEG devices need to perform PIM source registration only for multicast sources inside the EVPN network, so they look for and only do PIM registration for the sources in the advertised S-PMSI A-D routes.

OSPF Area for Server Leaf Device Connectivity on the SBD

On each of the server leaf devices, enhanced OISM requires that you include an OSPF area configuration for the SBD IRB interface in each tenant virtual routing and forwarding (VRF) instance. You configure the SBD IRB interface in OSPF active mode to establish OSPF adjacencies and support routing among the OISM leaf devices on the SBD. However, you set the OSPF interface priority to 0 so the SL devices don't ever assume the OSPF designated router (DR) or backup DR (BDR) role. You configure any other interfaces in the VRF instance in the OSPF area using OSPF passive mode, so they can exchange routing information but don't form OSPF adjacencies and participate in OSPF protocol processing.

OISM Components

IN THIS SECTION

- [OISM Device Roles | 1061](#)
- [PIM Domain with External Multicast Sources and Receivers | 1063](#)
- [Supported Methods for Multicast Data Transfer to or from an External PIM Domain | 1064](#)
- [OISM Bridge Domains \(VLANs\) | 1066](#)
- [Regular OISM Mode—Symmetric Bridge Domains Model | 1070](#)
- [Enhanced OISM Mode—Asymmetric Bridge Domains Model | 1072](#)
- [Configuration Elements for OISM Devices | 1072](#)

The OISM environment includes:

- Leaf devices in the EVPN fabric that function in border roles and server access roles.
- External multicast sources and receivers in an external L3 PIM domain.
- Bridge domain (VLAN) configurations that enable the fabric to route multicast traffic between internal and external devices.

The EVPN-VXLAN ERB overlay design includes lean spine devices that support L3 transit functions for the leaf devices. The lean spine devices don't usually perform any OISM functions.

The following sections describe these OISM components.

OISM Device Roles

[Figure 101 on page 1062](#) shows a simple EVPN-VXLAN ERB overlay fabric and the OISM device roles in the fabric.

Figure 101: EVPN Fabric with OISM

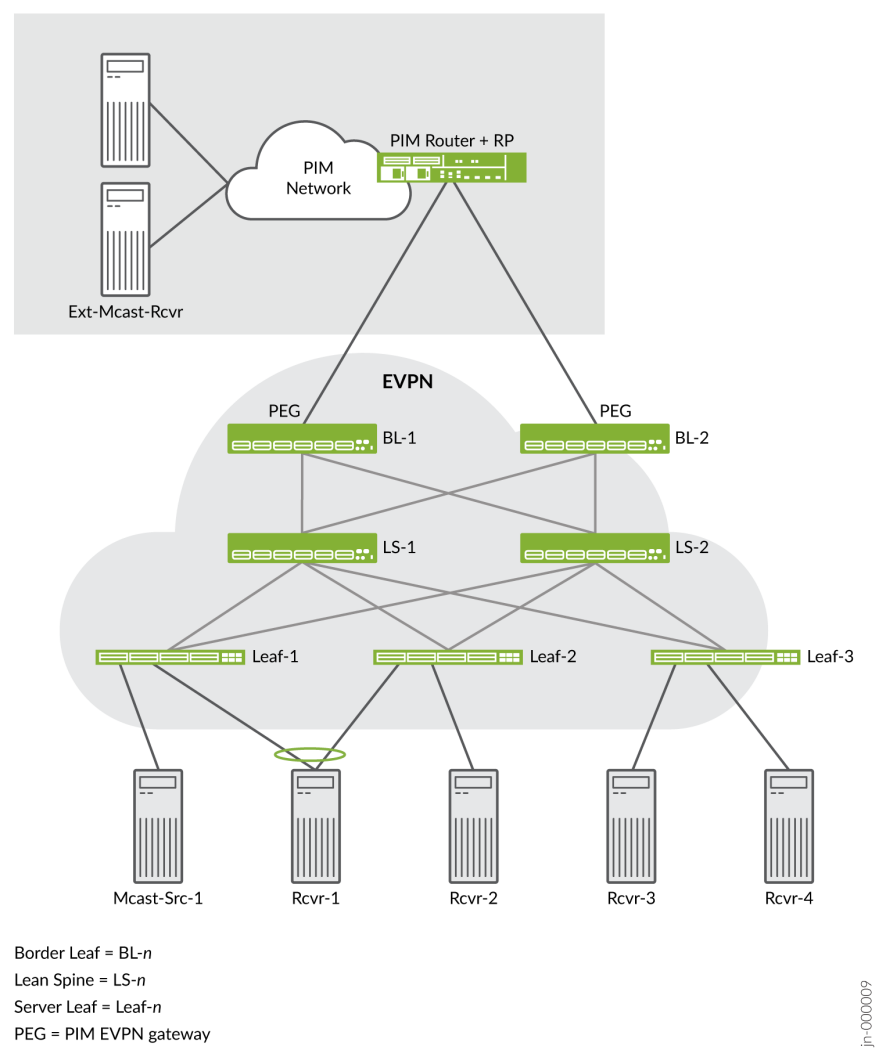


Table 57 on page 1062 summarizes the device roles.

Table 57: EVPN Fabric OISM Device Roles

Device Role	Description
Border leaf (BL)	OISM leaf devices in the EVPN fabric underlay and overlay. Border leaf devices function as gateways interconnecting the EVPN fabric to multicast devices (sources and receivers) outside the fabric in an external PIM domain. These devices serve in the PIM EVPN gateway (PEG) role.

Table 57: EVPN Fabric OISM Device Roles *(Continued)*

Device Role	Description
Lean spine (LS)	<p>Spine devices in the underlay of the EVPN fabric. These devices usually operate as lean spines that support the EVPN underlay as IP transit devices. The lean spines might also act as route reflectors in the fabric.</p> <p>You configure OISM elements on the lean spine devices only in the following use cases:</p> <ul style="list-style-type: none"> • The devices also serve as border devices for external multicast traffic. In this case, you configure the same OISM elements as you configure on border leaf devices. • The lean spine device serves as a standalone AR replicator when you integrate AR with OISM in the fabric. In this case, on the AR replicator spine device, you configure the same common OISM elements that you configure on the border leaf and server leaf devices. You don't need to configure any of the PIM or external multicast elements specific to border leaf or server leaf devices. (See "AR with Optimized Intersubnet Multicast (OISM)" on page 1029.)
Server leaf (Leaf)	<p>OISM leaf devices on the access side in the EVPN fabric underlay and overlay. Server leaf devices are often top-of-rack (ToR) switches. These devices connect the EVPN fabric to multicast sources and multicast receiver hosts on bridge domains or VLANs within the fabric.</p>

See ["Configuration Elements for OISM Devices " on page 1072](#) for details on the configuration elements that are common and those that are different for each device role.

PIM Domain with External Multicast Sources and Receivers

In [Figure 101 on page 1062](#), the OISM border leaf devices connect to multicast sources and receivers outside the EVPN fabric in a representative external PIM domain. The multicast devices in the external PIM domain follow standard PIM protocol procedures; their operation is not specific to OISM. External multicast traffic flows at L3 through the PIM domain.

You can use OISM to route and to forward multicast traffic in an EVPN-VXLAN ERB overlay fabric between devices in the following use cases:

- Internal multicast sources and receivers
- Internal multicast sources and external multicast receivers
- External multicast sources and internal multicast receivers

For simplicity, in this documentation we represent the external PIM domain as:

- A PIM router (a device such as an MX Series router) that doubles as the PIM rendezvous point (RP).
- An external source.
- An external receiver.

Supported Methods for Multicast Data Transfer to or from an External PIM Domain

OISM border leaf devices support one or more methods to route multicast traffic to and from devices outside of the fabric. Supported methods are platform-dependent.

Some platforms don't support the border leaf role. If you don't see a platform listed in [Table 58 on page 1064](#) in the **Supported Platforms** column for any of the external multicast methods, that means the platform doesn't support the border leaf role.

Table 58: External Multicast Connection Methods

Name	Connection Method	Supported Platforms
M-VLAN IRB method	<p>IRB interfaces on a multicast VLAN (M-VLAN) that you extend in the EVPN instance. The fabric uses the M-VLAN and corresponding IRB interfaces only for external multicast traffic flow to and from the external PIM domain.</p> <p>This method supports EVPN Ethernet segment identifier (ESI) multihoming to connect the external PIM router to more than one OISM border leaf device in the fabric.</p> <p>NOTE: We don't support this method with enhanced OISM.</p>	<p>PTX10001-36MR</p> <p>PTX10002-36QDD</p> <p>PTX10004</p> <p>PTX10008</p> <p>PTX10016</p> <p>QFX10002 (except QFX10002-60C)</p> <p>QFX10008</p> <p>QFX10016</p>

Table 58: External Multicast Connection Methods (*Continued*)

Name	Connection Method	Supported Platforms
Classic L3 interface method	<p>Classic physical L3 interfaces on OISM border leaf devices that connect individually to the external PIM domain on different subnets.</p> <p>These interfaces aren't associated with a VLAN. You don't configure these interfaces in the EVPN instances. Instead, you assign IP addresses to these interfaces and include them in the tenant L3 VRF instances.</p> <p>NOTE: The L3 interface connection can be an aggregated Ethernet (AE) interface bundle.</p>	ACX7024 ACX7100 ACX7332 ACX7348 ACX7509 EX4400 EX4650 PTX10001-36MR PTX10002-36QDD PTX10004 PTX10008 PTX10016 QFX5110 QFX5120 QFX5130-32CD QFX5700 QFX10002 (except QFX10002-60C) QFX10008 QFX10016

Table 58: External Multicast Connection Methods (*Continued*)

Name	Connection Method	Supported Platforms
Non-EVPN IRB method	<p>IRB interfaces on an extra VLAN that you don't extend in the EVPN instance. You include these logical interfaces in the tenant L3 VRF instances.</p> <p>On each border leaf device, you assign a unique extra VLAN ID and subnet for the associated IRB interface.</p> <p>We call this type of interface a <i>non-EVPN IRB interface</i> for external multicast.</p>	<p>PTX10001-36MR</p> <p>PTX10002-36QDD</p> <p>PTX10004</p> <p>PTX10008</p> <p>PTX10016</p> <p>QFX5130-32CD</p> <p>QFX5700</p>

See ["Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment" on page 934](#) for a general overview of connecting EVPN-VXLAN fabrics to an external PIM domain using a L2 M-VLAN or L3 links.

OISM Bridge Domains (VLANs)

[Table 59 on page 1067](#) summarizes the OISM bridge domains or VLANs and describes how OISM uses them.



NOTE: References in this document to *all OISM devices* correspond to the border leaf and server leaf devices on which you enable OISM.

Table 59: OISM Bridge Domains or VLANs

Bridge Domain/ VLAN	Description	Configure On:
Multicast VLAN (M-VLAN)	<p>(M-VLAN IRB method for external multicast) A VLAN in the EVPN fabric with associated IRB interfaces that connect the fabric to an external multicast router. This VLAN and IRB interface enable traffic flow between devices inside and outside the fabric. To support IGMP snooping with both IGMPv2 and IGMPv3 traffic, you assign separate M-VLANs to carry traffic for each IGMP version.</p> <p>You extend this VLAN in the EVPN instance. You can also multihome the external multicast router to multiple border leaf device M-VLAN IRB interfaces in the same EVPN ES. The usual EVPN multihoming DF rules apply to send only one copy of the traffic in the ES on the M-VLAN.</p> <p>Configure the M-VLAN as a VLAN that is not the SBD or any of the revenue bridge domains in the EVPN fabric.</p> <p>NOTE: We don't support the M-VLAN IRB method with enhanced OISM.</p> <p>See "Supported Methods for Multicast Data Transfer to or from an External PIM Domain" on page 1064 for other supported methods to connect to external sources and receivers.</p>	Border leaf devices

Table 59: OISM Bridge Domains or VLANs (*Continued*)

Bridge Domain/ VLAN	Description	Configure On:
Extra non-EVPN VLAN	<p>(Non-EVPN IRB method for external multicast) An extra VLAN that isn't in the EVPN instances in the fabric. You configure associated IRB interfaces in the tenant L3 VRF instances. This VLAN and IRB interface enable multicast traffic flow between devices inside the fabric and devices outside the fabric.</p> <p>You must assign distinct extra VLANs and corresponding IRB interface subnets on each border leaf device that are unique across the fabric.</p> <p>Also, to support IGMP snooping with both IGMPv2 and IGMPv3 traffic, use separate distinct extra non-EVPN VLANs to carry traffic for each IGMP version. The same constraints apply if you want to support MLD snooping with both MLDv1 and MLDv2 traffic.</p> <p>NOTE: See "Supported Methods for Multicast Data Transfer to or from an External PIM Domain" on page 1064 for other supported methods to connect to external sources and receivers.</p>	Border leaf devices
Revenue bridge domains (VLANs)	<p>Bridge domains for subscribers to the services that the fabric provides. You configure the revenue bridge domains as VLANs in the fabric.</p> <p>The revenue bridge domains correspond to the <i>customer VLANs</i> in the fabric. These VLANs are not specific to OISM, but the multicast sources and receivers in the fabric are in these bridge domains.</p> <p>For details on how to allocate VLANs for these bridge domains if you want to support IGMP snooping with both IGMPv2 and IGMPv3 receivers in the fabric, see "IGMPv2 and IGMPv3 (or MLDv1 and MLDv2) in the Same EVPN-VXLAN Fabric" on page 1117. Note that the same constraints apply if you want to support MLD snooping with both MLDv1 and MLDv2 receivers.</p>	All OISM devices

Table 59: OISM Bridge Domains or VLANs *(Continued)*

Bridge Domain/ VLAN	Description	Configure On:
Supplemental bridge domain (SBD)	<p>Bridge domain that enables support for external multicast traffic, implements SMET optimization in the EVPN core, and supports the enhanced OISM mode implementation where all devices don't need to know all VLANs in the network.</p> <p>You configure the SBD as a regular VLAN that is different from the revenue bridge domain VLANs, the M-VLAN, or the extra non-EVPN VLANs.</p> <p>The SBD usually serves all OISM leaf devices in the fabric. To support IGMP snooping with both IGMPv2 and IGMPv3 traffic, or to support MLD snooping with both MLDv1 and MLDv2 traffic, you assign separate SBDs to carry traffic for each IGMP or MLD version. The SBD carries:</p> <ul style="list-style-type: none"> • North-south routed multicast data traffic from external multicast sources to EVPN devices in the fabric with interested receivers. • SMET EVPN Type 6 routes from originating leaf devices to other EVPN leaf devices, which enables selective forwarding in the EVPN core. • (Enhanced OISM only) East-west multicast data traffic from internal sources to other leaf devices with interested receivers, when the ingress leaf device isn't a multihoming peer to another leaf device. In enhanced OISM mode, a leaf device routes source traffic on the source VLAN only to its multihoming peer leaf devices. (In contrast, the regular OISM symmetric bridge domains model sends all east-west traffic on the source VLAN.) <p>NOTE: The SBD is central to OISM operation for:</p> <ul style="list-style-type: none"> • External multicast traffic routing and forwarding • (With enhanced OISM) Internal multicast traffic routing and forwarding from internal sources. 	All OISM devices

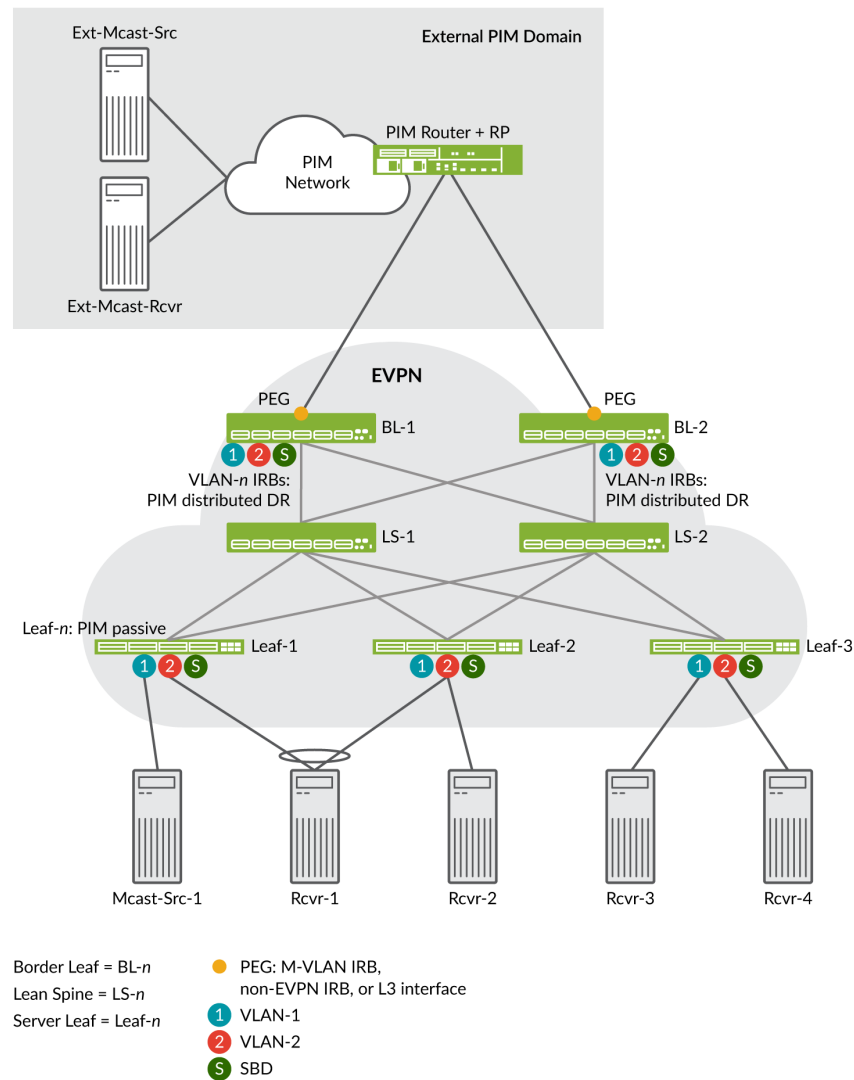
Table 59: OISM Bridge Domains or VLANs (*Continued*)

Bridge Domain/ VLAN	Description	Configure On:
	<p>The SBD IRB interface must always be up for OISM to work.</p> <p>You configure the VLAN that you want to use as the SBD for each L3 VRF instance, and configure a corresponding IRB interface. Use the supplemental-bridge-domain-irb <i>irb-interface-name</i> statement at the [edit routing-instances <i>l3-vrf-instance</i> protocols evpn oism] hierarchy level to associate that VLAN and IRB interface as the SBD for the indicated L3 VRF instance.</p>	

Regular OISM Mode—Symmetric Bridge Domains Model

The regular OISM implementation uses a *symmetric bridge domains* model. We also refer to this OISM mode as the bridge domains everywhere (BDE) model. You configure all of the revenue bridge domains (VLANs) on all of the OISM devices in the network with this model.

Figure 102: EVPN Fabric with Regular OISM



In the symmetric bridge domains model, you must configure the SBD and all revenue bridge domains on all OISM border leaf and server leaf devices.

You also configure the following VLANs uniformly on the border leaf devices that connect to the external PIM domain:

- The M-VLAN, if you use the M-VLAN IRB method for external multicast.
- A unique extra non-EVPN VLAN on each border leaf device, if you use the non-EVPN IRB method for external multicast.

The lean spine devices in the fabric usually serve only as IP transit devices and possibly as route reflectors. As a result, you don't usually need to configure these elements on the lean spine devices. (See the Lean Spine row in [Table 57 on page 1062](#) for some exceptions.)

Enhanced OISM Mode—Asymmetric Bridge Domains Model

The enhanced OISM implementation supports an asymmetric bridge domains model in which on each leaf device, you can configure only the revenue VLANs that the device hosts. As a result, we sometimes refer to enhanced OISM as the *bridge domains NOT everywhere* (BDNE) model.

In general, enhanced OISM uses the same high-level OISM structure and network components you see in [Figure 102 on page 1071](#), but with some operational differences to enable the asymmetric bridge domains model.

See ["Overview of Enhanced OISM" on page 1057](#) for an introduction to the main differences from the regular OISM implementation. Throughout this document, we describe the operational or configuration differences when you use enhanced OISM instead of regular OISM, as applicable.

Configuration Elements for OISM Devices

This section summarizes the elements you need to configure on:

- All OISM devices—Devices in the border leaf role and the server leaf role, and spine devices that also serve as AR replicators in standalone mode when you integrate AR with OISM.

See [Table 60 on page 1073](#).

- Server leaf devices only.

See [Table 61 on page 1079](#).

- Border leaf devices only, based on the method you use for external multicast:

- M-VLAN IRB interface
- Classic L3 interface
- Non-EVPN IRB interface

See [Table 62 on page 1080](#).

Some elements are optional, which the description notes.



NOTE: EX4650, QFX5110, and QFX5120 switches support enterprise style interface configurations for OISM elements, but not service provider style interface

configurations. For more information on these interface configuration styles, see *Flexible Ethernet Services Encapsulation* and ["Understanding Flexible Ethernet Services Support With EVPN-VXLAN" on page 654](#).

[Table 60 on page 1073](#) lists the elements you configure on all OISM devices.

Table 60: Configuration Elements on all OISM Devices

Configuration Element	Description
OISM mode	<p>Enable OISM globally, and enable OISM routing functions in L3 VRF instances. You enable OISM in the regular mode or the enhanced mode, and all devices must run the same OISM mode.</p> <p>A device with OISM enabled advertises EVPN Type 3 Inclusive Multicast Ethernet Tag (IMET) routes as follows:</p> <ul style="list-style-type: none"> • The device advertises an IMET route for each configured revenue bridge domain (VLAN) and the SBD. • The IMET routes include the EVPN multicast flags extended community with a flag indicating OISM support.
Revenue bridge domains (customer VLANs) and corresponding IRB interfaces	<p>Configure revenue bridge domains (customer VLANs) according to your data center services requirements. With regular OISM, you must configure all revenue bridge domain VLANs and corresponding IRB interfaces symmetrically on all OISM devices. With enhanced OISM, on each OISM device you need to configure only the revenue VLANs that the device hosts, along with the corresponding IRB interfaces. However, you must still configure the revenue VLANs symmetrically on any sets of multihoming peer leaf devices. See "OISM Bridge Domains (VLANs)" on page 1066 for more information.</p> <p>See "IGMPv2 and IGMPv3 (or MLDv1 and MLDv2) in the Same EVPN-VXLAN Fabric" on page 1117 for special considerations to configure the revenue bridge domains if you want to support:</p> <ul style="list-style-type: none"> • IGMP snooping with traffic for both IGMPv2 and IGMPv3 receivers. • MLD snooping with traffic for both MLDv1 and MLDv2 receivers,

Table 60: Configuration Elements on all OISM Devices (*Continued*)

Configuration Element	Description
SBD (VLAN) and corresponding IRB interface	<p>On all OISM devices, in each L3 VRF that supports OISM routing, configure a VLAN as the SBD and a corresponding IRB interface. The SBD can be any VLAN that is distinct from the M-VLAN, any of the non-EVPN VLANs or any revenue bridge domain VLANs in the EVPN fabric. See "OISM Bridge Domains (VLANs)" on page 1066 for more information.</p> <p>To identify this VLAN as the SBD in the L3 VRF instance, configure the supplemental-bridge-domain-irb <i>SBD-irb-interface-name</i> statement at the [edit routing-instances <i>l3-vrf-instance</i> protocols evpn oism] hierarchy level.</p> <p>Starting in Junos OS and Junos OS Evolved 24.1R1, for interoperability with other vendors and compliance with the OISM draft standard, the EVPN Type 3 IMET routes for the SBD IRB interface include the OISM SBD flag in the multicast flags extended community.</p>
L3 multicast protocol—IGMPv2 or IGMPv3	<p>Enable IGMPv2 or IGMPv3 L3 multicast protocols. Receivers send IGMP reports to express interest in receiving traffic for a multicast group.</p> <p>You can use IGMPv2 or IGMPv3 in any-source multicast (ASM) mode, or IGMPv3 in source-specific multicast (SSM) mode.</p> <p>Note that you can't enable IGMP snooping for both IGMP versions together for the same VLAN or in the same VRF instance with OISM enabled. However, to support IGMP snooping with IGMPv2 and IGMPv3 receivers in the same fabric, you can enable IGMP snooping with IGMPv2 for specific VLANs in one VRF instance, and enable IGMP snooping with IGMPv3 for other VLANs in another VRF instance. See "IGMPv2 and IGMPv3 (or MLDv1 and MLDv2) in the Same EVPN-VXLAN Fabric" on page 1117 for details.</p> <p>IGMPv2 is the default IGMP version. To configure IGMPv3, you must specify the version 3 option.</p>

Table 60: Configuration Elements on all OISM Devices *(Continued)*

Configuration Element	Description
L3 multicast protocol— MLDv1 or MLDv2	<p data-bbox="537 352 1406 447">Enable MLDv1 or MLDv2 L3 multicast protocols if you have IPv6 multicast traffic in your fabric. Receivers send MLD reports to express interest in receiving traffic for a multicast group.</p> <p data-bbox="537 485 1390 548">You can use MLDv1 or MLDv2 in any-source multicast (ASM) mode, or MLDv2 in source-specific multicast (SSM) mode.</p> <p data-bbox="537 583 1406 716">Note that with OISM enabled, you can't enable MLD snooping for both MLD versions together for the same VLAN or in the same VRF instance. However, you can support MLD snooping with MLDv1 and MLDv2 receivers in the same fabric if you:</p> <ul data-bbox="537 751 1382 846" style="list-style-type: none"> <li data-bbox="537 751 1357 783">• Enable MLD snooping with MLDv1 for specific VLANs in one VRF instance. <li data-bbox="537 814 1382 846">• Enable MLD snooping with MLDv2 for other VLANs in another VRF instance. <p data-bbox="537 882 1357 945">See "IGMPv2 and IGMPv3 (or MLDv1 and MLDv2) in the Same EVPN-VXLAN Fabric" on page 1117 for details.</p> <p data-bbox="537 980 1357 1043">MLDv1 is the default MLD version. To configure MLDv2, you must specify the version 2 option.</p>

Table 60: Configuration Elements on all OISM Devices *(Continued)*

Configuration Element	Description
L2 multicast optimizations—IGMP snooping with SMET	<p>Enable IGMP snooping at L2 with IGMPv2 or IGMPv3 protocols as part of the optimizations OISM provides. With IGMP snooping, the device routes or forwards multicast traffic only toward interested access-side receivers. Receivers send IGMP reports to express interest in receiving traffic for a multicast group.</p> <p>When you enable IGMP snooping, the device also automatically advertises SMET Type 6 routes. With SMET, the device sends copies of the traffic into the EVPN core only toward other devices that have interested receivers.</p> <p>Configure IGMP snooping as follows:</p> <ul style="list-style-type: none"> • In the EVPN instance(s) for each OISM revenue VLAN and the SBD. • On border leaf devices for external multicast only with the M-VLAN IRB method or the non-EVPN IRB method, as follows: <ul style="list-style-type: none"> • M-VLAN IRB method: In the EVPN instance(s) for the M-VLAN IRB multicast router interface. Include the <code>evpn-ssm-reports-only</code> option only with IGMPv3. • Non-EVPN IRB method: Globally for the non-EVPN IRB interface configured as a multicast router interface. <p>With IGMPv3, you don't need the IGMP snooping <code>evpn-ssm-reports-only</code> option because the external multicast interface isn't extended in the EVPN instance.</p> <p>NOTE: With the classic L3 interface method for external multicast, you don't configure IGMP snooping, an L2 optimization, on the pure L3 interfaces.</p>

Table 60: Configuration Elements on all OISM Devices (*Continued*)

Configuration Element	Description
L2 multicast optimizations—MLD snooping with SMET for IPv6 multicast traffic	<p>Enable MLD snooping at L2 with MLDv1 or MLDv2 protocols if you have IPv6 multicast traffic in the fabric. With MLD snooping, the device routes or forwards multicast traffic only toward interested access-side receivers. Receivers send MLD reports to express interest in receiving traffic for a multicast group.</p> <p>When you enable MLD snooping, the device also automatically advertises SMET Type 6 routes. With SMET, the device sends copies of the traffic into the EVPN core only toward other devices that have interested receivers.</p> <p>Configure MLD snooping as follows:</p> <ul style="list-style-type: none"> • In the EVPN instance(s) for each OISM revenue VLAN and the SBD. • On border leaf devices for external multicast only with the non-EVPN IRB method with these guidelines: <ul style="list-style-type: none"> • Configure MLD snooping globally with the non-EVPN IRB interface configured as a multicast router interface. • With MLDv2 in SSM mode, you don't need the MLD snooping evpn-ssm-reports-only option because the external multicast interface isn't extended in the EVPN instance. <p>NOTE: With the classic L3 interface method for external multicast, you don't configure MLD snooping, an L2 optimization, on the pure L3 interfaces.</p>
L3 VRF instance (or the default IRB routing instance)	<p>Configure one or more routing instances (instance type vrf) to support the L3 OISM routing functions. Or, on devices that support OISM in a default IRB routing instance, you can define the L3 routing-related parameters in the default routing instance instead.</p> <p>If your configuration has user-specified VRF instances, then you configure the relevant L3 OISM routing functions at the [edit routing-instances vrf-instance-name ...] hierarchy level. If your configuration uses the default L3 routing instance, then you configure the VRF instance elements at the global level instead. See "OISM Support with Tenant L3 VRF Instances or the Default L3 Routing Instance" on page 1052 for details.</p>

Table 60: Configuration Elements on all OISM Devices *(Continued)*

Configuration Element	Description
originate-smet-on-revenue-vlan-too	<p>(Optional) Enable the device to originate SMET Type 6 routes for revenue bridge domains (as well as on the SBD) upon receiving local IGMP or MLD reports. By default, OISM devices originate Type 6 routes only on the SBD.</p> <p>Use this option for compatibility with other vendor devices that don't support OISM. Those devices can't create the right states for the revenue bridge domain VLANs upon receiving Type 6 routes on the SBD.</p>
install-star-g-routes	<p>Enable the Routing Engine (RE) on the device to install (*,G) multicast routes on the Packet Forwarding Engine (PFE) for all of the revenue bridge domain VLANs in the routing instance immediately upon receiving an EVPN Type 6 route. Setting this option helps minimize traffic loss when multicast traffic first arrives.</p> <p>This option is mutually exclusive with the <code>conserve-mcast-routes-in-pfe</code> option, so you can't set both options together in a routing instance.</p> <p>We require this option on:</p> <ul style="list-style-type: none"> • (Regular OISM mode only) The QFX10000 line of switches, QFX5130-32CD switches, and QFX5700 switches when you configure those devices in the AR replicator role. • In releases prior to Junos OS and Junos OS Evolved Release 23.4R1, on the QFX10000 line of switches and the PTX10000 line of routers when you configure those devices as OISM server leaf or border leaf devices. <p>We no longer require this option in this case starting in Junos OS and Junos OS Evolved Release 23.4R1.</p> <p>Setting this option is generally not recommended in any use cases other than those listed above where it is required.</p> <p>See "Latency and Scaling Trade-Offs for Installing Multicast Routes with OISM (install-star-g-routes Option)" on page 1122 for details on how and when to set this option.</p>

Table 60: Configuration Elements on all OISM Devices *(Continued)*

Configuration Element	Description
conserve-mcast-routes-in-pfe	<p>(Required on ACX Series routers, QFX5130-32CD switches, and QFX5700 switches when you configure those devices as OISM server leaf or OISM border leaf devices)</p> <p>Configure this option with OISM to conserve PFE table space. The device installs only the L3 multicast routes and avoids installing L2 multicast snooping routes.</p> <p>Don't set this option on QFX5130-32CD and QFX5700 switches when you configure those devices as standalone AR replicator devices with OISM. This option is mutually exclusive with the install-star-g-routes option, so you can't set both options together in a routing instance.</p> <p>See "ACX Series Routers, QFX5130-32CD Switches, and QFX5700 Switches as Server Leaf and Border Leaf Devices with OISM" on page 1125 for details.</p>

[Table 61 on page 1079](#) lists the elements you configure on the server leaf devices.

Table 61: Configuration Elements on OISM Server Leaf Devices

Configuration Element	Description
PIM in passive mode on all revenue bridge domains and the SBD	<p>Configure this mode to facilitate local routing without all of the traditional PIM protocol functions. The server leaf device:</p> <ul style="list-style-type: none"> • Doesn't form PIM neighbor relationships with any other devices (avoids sending or receiving PIM hello messages). • Acts as a PIM local RP. The device creates the PIM state locally from IGMP or MLD reports. The device also doesn't do source registration. Only the border leaf devices perform source registration toward the external PIM RP.
PIM with the accept-remote-source option on SBD IRB interfaces	<p>This option enables an SBD IRB interface to accept multicast traffic from a source that isn't on the same subnet. The server leaf devices require this setting because:</p> <ul style="list-style-type: none"> • The border leaf devices route the multicast source traffic from the external multicast interfaces to the SBD toward the server leaf devices. • The traffic arrives at the server leaf device on the SBD IRB interface, which is not a PIM neighbor. The source is not a local source, so without this setting, the device would otherwise drop the traffic.

Table 61: Configuration Elements on OISM Server Leaf Devices *(Continued)*

Configuration Element	Description
<p>OSPF in tenant VRFs for peer connectivity to support:</p> <ul style="list-style-type: none"> • External multicast traffic on SBD • (Enhanced OISM) East-west traffic on SBD 	<p>Configure OSPF in each tenant VRF so the server leaf devices learn routes:</p> <ul style="list-style-type: none"> • To external sources when multicast traffic from outside the fabric arrives on the SBD. • (Enhanced OISM) For east-west traffic arriving from other leaf devices on the SBD. <p>The device creates the PIM (S,G) entries it needs to forward the traffic from the SBD to the revenue bridge domains.</p> <p>With regular OISM, on server leaf devices, you configure all interfaces in the L3 VRF instance in OSPF passive mode so these devices can share internal routes without forming OSPF adjacencies.</p> <p>With enhanced OISM only, on server leaf devices, you configure the SBD IRB interface in the L3 VRF instance in OSPF active mode. The SBD IRB interfaces need to establish OSPF adjacencies in this case because the server leaf devices exchange multicast traffic among themselves mostly on the SBD. You configure all other interfaces in the L3 VRF instance in OSPF passive mode.</p>

Table 62 on page 1080 lists the elements you configure on the border leaf devices based on the external multicast method you use.

Table 62: Configuration Elements on OISM Border Leaf Devices

Configuration Element	External Multicast Method	Description
M-VLAN and corresponding IRB interface (in EVPN instance)	M-VLAN IRB method	<p>Configure a VLAN to serve as the M-VLAN, and extend this VLAN in the EVPN instance. This VLAN must be distinct from the SBD or any revenue bridge domain VLANs in the EVPN fabric. Also configure an M-VLAN IRB interface in the EVPN instance. See "OISM Bridge Domains (VLANs)" on page 1066 for more information about the M-VLAN.</p> <p>You can link multiple border leaf device M-VLAN IRB interfaces to the external multicast router in the same EVPN ES. The usual EVPN multihoming DF rules apply to prevent sending duplicate traffic on the M-VLAN.</p>

Table 62: Configuration Elements on OISM Border Leaf Devices (*Continued*)

Configuration Element	External Multicast Method	Description
L2 multicast router interface on external multicast ports	M-VLAN IRB method or non-EVPN IRB method	<p>Configure the <code>multicast-router-interface</code> option with IGMP snooping or MLD snooping on the L2 ports that link the border leaf device to the external PIM domain at L2.</p> <p>With the M-VLAN IRB method, these interfaces support multicast traffic when the external domain router is multihomed to the border leaf devices. As a result, multihomed M-VLAN use cases require this configuration. This setting is also required with the non-EVPN IRB method.</p>
PIM on M-VLAN IRB interface	M-VLAN IRB method	<p>Configure PIM in distributed designated router (DR) mode (<i>distributed-dr</i>) or standard PIM mode on an M-VLAN IRB interface. We recommend using distributed DR mode in most cases, especially on border leaf devices where the external PIM router is multihomed to multiple border leaf devices.</p> <p>The device uses PIM to:</p> <ul style="list-style-type: none"> Form PIM neighbor relationships with the other border leaf devices and the external PIM router. The external PIM router might be multihomed in an ES. As a result, EVPN-forwarded traffic might come to a different peer border leaf device. Elect a single last-hop router (LHR) designated router (DR) for the M-VLAN, so only one device does source registration for an internal source toward the PIM RP. <p>You configure PIM on the M-VLAN IRB interface in the tenant VRF instances, similar to how you configure PIM on the revenue bridge domains.</p>

Table 62: Configuration Elements on OISM Border Leaf Devices *(Continued)*

Configuration Element	External Multicast Method	Description
L3 physical interface with IP address	Classic L3 interface	<p>Configure a physical L3 interface with an IP address for external multicast that connects the border leaf device to the external PIM domain at L3.</p> <p>Define the external multicast L3 interface in a different subnet on each border leaf device.</p> <p>NOTE: The L3 interface connection can be an AE interface bundle.</p>
PIM on the logical interface for the external multicast physical L3 interface	Classic L3 interface	<p>Configure the logical interface (unit 0) for the external multicast L3 interface in the tenant VRF instances. Configure standard PIM mode on the logical interface.</p> <p>With this setting, the border leaf device forms a PIM neighbor relationship with the external PIM router to send join messages and transmit or receive external multicast traffic.</p>
Extra VLAN and corresponding IRB interface (not in EVPN instance)	Non-EVPN IRB method	<p>Configure an extra VLAN and IRB interface globally for external multicast without EVPN signaling. This VLAN and the IRB interface subnet must be distinct from the SBD, any revenue bridge domain VLANs, or the extra VLAN on any other border leaf device in the EVPN fabric. See "OISM Bridge Domains (VLANs)" on page 1066 for more about this extra VLAN and external multicast method.</p>
PIM on non-EVPN IRB interface	Non-EVPN IRB method	<p>Configure PIM on the non-EVPN IRB interface in the tenant VRF instances.</p> <p>With this setting, the border leaf device forms a PIM neighbor relationship with the external PIM router to send join messages and transmit or receive external multicast traffic.</p>

Table 62: Configuration Elements on OISM Border Leaf Devices *(Continued)*

Configuration Element	External Multicast Method	Description
PIM on SBD IRB interface	All	<p>Configure standard PIM mode on the SBD IRB interface in the tenant VRF instances for SBD routing and forwarding. With this setting, the border leaf device:</p> <ul style="list-style-type: none"> • Routes external multicast source traffic from the external multicast interfaces to the SBD, and forwards copies toward server leaf devices with multicast receivers. • Forms PIM neighbor relationships with the other border leaf devices. • Elects a single LHR DR on the SBD among the peer border leaf devices. Only this elected PIM DR forwards the external multicast source traffic on the SBD. This election prevents peer border leaf devices from forwarding duplicate traffic into the EVPN core.
PIM with the accept-remote-source option on SBD IRB interfaces	<p>Methods (platform-specific) supported with enhanced OISM:</p> <ul style="list-style-type: none"> • Classic L3 interface • Non-EVPN IRB method 	<p>(Enhanced OISM only) With this option, the border leaf device accepts multicast traffic from a source that isn't on the same subnet. We need this option because with enhanced OISM, you might not have configured all of the revenue VLANs on all of the OISM devices. Including this option enables border devices to have routes to a multicast source located behind other OISM leaf devices when the source is on a VLAN that isn't also configured on the border leaf device.</p>

Table 62: Configuration Elements on OISM Border Leaf Devices (*Continued*)

Configuration Element	External Multicast Method	Description
PIM EVPN gateway (PEG) role	All (include the external IRB option for EVPN, external-irb <i>interface-name</i> , only with the M-VLAN IRB method)	<p>Configure the <i>pim-evpn-gateway</i> role on the border leaf device to connect to the external PIM router. In this role, the border leaf device uses traditional PIM routing behavior and does local routing, as follows:</p> <p>For externally-sourced traffic:</p> <ul style="list-style-type: none"> • Routes external source traffic from the M-VLAN IRB interface, L3 interface, or non-EVPN IRB interface to any local receivers on the revenue bridge domains on the border leaf device. • Routes external source traffic from the M-VLAN IRB interface, L3 interface, or non-EVPN IRB interface to the SBD to reach any internal receivers in the fabric. (The device forwards the traffic to the EVPN core only on the SBD to other OISM leaf devices.) <p>For internally-sourced traffic:</p> <ul style="list-style-type: none"> • Locally routes traffic that arrives on a revenue bridge domain to other revenue bridge domains. • Routes the traffic to external multicast receivers through the M-VLAN IRB interface, L3 interface, or non-EVPN IRB interface. The device doesn't forward the traffic back into the EVPN core. <p>EVPN IMET routes for PEG interfaces include the OISM PEG flag in the multicast flags extended community field of the route.</p>

Table 62: Configuration Elements on OISM Border Leaf Devices *(Continued)*

Configuration Element	External Multicast Method	Description
<p>OSPF for:</p> <ul style="list-style-type: none"> • External multicast interface peer connectivity • (Enhanced OISM) East-west traffic on SBD 	All	<p>Configure an OSPF area in the tenant L3 VRF instance so the border leaf device learns routes to the multicast sources. The device requires these routes to support forwarding multicast traffic:</p> <ul style="list-style-type: none"> • From sources inside the fabric toward receivers outside the fabric. • From sources outside the fabric toward receivers inside the fabric. <p>The device needs this route information to create the PIM (S,G) entries to forward the traffic on the external multicast interfaces, the SBD, and the revenue bridge domains.</p> <p>(Enhanced OISM) The border leaf devices also need to learn the routes for east-west traffic on the SBD among the leaf devices that aren't multihoming peers.</p> <p>As a result, with either regular OISM or enhanced OISM, on border leaf devices, you configure OSPF in active mode on:</p> <ul style="list-style-type: none"> • The SBD IRB interface • The external multicast interface—the M-VLAN IRB interface, the L3 interface, or the non-EVPN IRB interface <p>You configure any other interfaces in the L3 VRF instance in OSPF passive mode.</p>

Table 62: Configuration Elements on OISM Border Leaf Devices (*Continued*)

Configuration Element	External Multicast Method	Description
PIM distributed DR mode on revenue bridge domain IRB interfaces	All	<p>Configure PIM in distributed DR mode (<i>distributed-dr</i>) on the revenue bridge domain IRB interfaces in the tenant VRF instances. In this mode, the border leaf device:</p> <ul style="list-style-type: none"> • Forms PIM neighbor relationships with other PIM devices to support multihomed external PIM router connections. • Acts as the LHR DR on its revenue bridge domain IRB interfaces, and creates the PIM state locally from received IGMP or MLD reports. As a result: <ul style="list-style-type: none"> • The device can do local multicast routing between the revenue bridge domains. • One peer border leaf device becomes the PIM DR that performs source registration toward the PIM RP. PIM hello messages and the PIM DR election process determine the PIM DR.

Table 62: Configuration Elements on OISM Border Leaf Devices (*Continued*)

Configuration Element	External Multicast Method	Description
PIM accept-join-always-from option and policy on M-VLAN IRB interface	M-VLAN IRB method	<p>Set this option on the M-VLAN IRB interface in the tenant VRF instances when the external PIM router is multihomed to more than one EVPN border leaf device. With this option, the device can accept and install the same PIM (S,G) join states on multihoming peer border leaf devices. This option supports sending multicast traffic from sources inside the fabric to receivers in the external PIM domain.</p> <p>With multihoming on the M-VLAN, the usual EVPN multihoming DF rules apply in an ES to prevent sending duplicate traffic. If peer border leaf devices have the same valid join states in place, any device that is the EVPN DF can forward the multicast traffic.</p> <p>Configure this statement with policies that specify the interface should always install PIM joins from upstream neighbor addresses that correspond to the external PIM router.</p> <p>NOTE: You don't use this option with the classic L3 interface and non-EVPN IRB methods. Those methods don't extend the external multicast interfaces in the EVPN instance.</p>

See the following sections for more details on configuring OISM devices:

- ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices " on page 1158](#)
- ["Configure Server Leaf Device OISM Elements" on page 1169](#)
- ["Configure Border Leaf Device OISM Elements with M-VLAN IRB Method \(Symmetric Bridge Domains Model Only\)" on page 1170](#)
- ["Configure Border Leaf Device OISM Elements with Classic L3 Interface Method" on page 1175](#)
- ["Configure Border Leaf Device OISM Elements with Non-EVPN IRB Method" on page 1177](#)

For a full OISM configuration example of a data center fabric use case that includes classic L3 interface connections to the external PIM domain, see [Optimized Intersubnet Multicast \(OISM\) with Assisted Replication \(AR\) for Edge-Routed Bridging Overlays](#).

How OISM Works

IN THIS SECTION

- [Local Routing on OISM Devices | 1088](#)
- [Multicast Traffic Forwarding and Routing with Source and Receivers Inside the EVPN Data Center | 1090](#)
- [Multicast Traffic From an Internal Source to Receivers Outside the EVPN Data Center—M-VLAN IRB Method | 1091](#)
- [Multicast Traffic from an Internal Source to Receivers Outside the EVPN Data Center—L3 Interface Method or Non-EVPN IRB Method | 1094](#)
- [Multicast Traffic from an External Source to Receivers Inside the EVPN Data Center—M-VLAN IRB Method | 1097](#)
- [Multicast Traffic from an External Source to Receivers Inside the EVPN Data Center—L3 Interface Method or Non-EVPN IRB Method | 1102](#)
- [AR and OISM with an Internal Multicast Source | 1106](#)
- [AR and OISM with an Internal Multicast Source and Multihomed Receiver | 1108](#)
- [AR and OISM with an External Multicast Source | 1110](#)

The following sections describe how OISM works and show how the multicast traffic flows in several common use cases with the symmetric bridge domains OISM model.

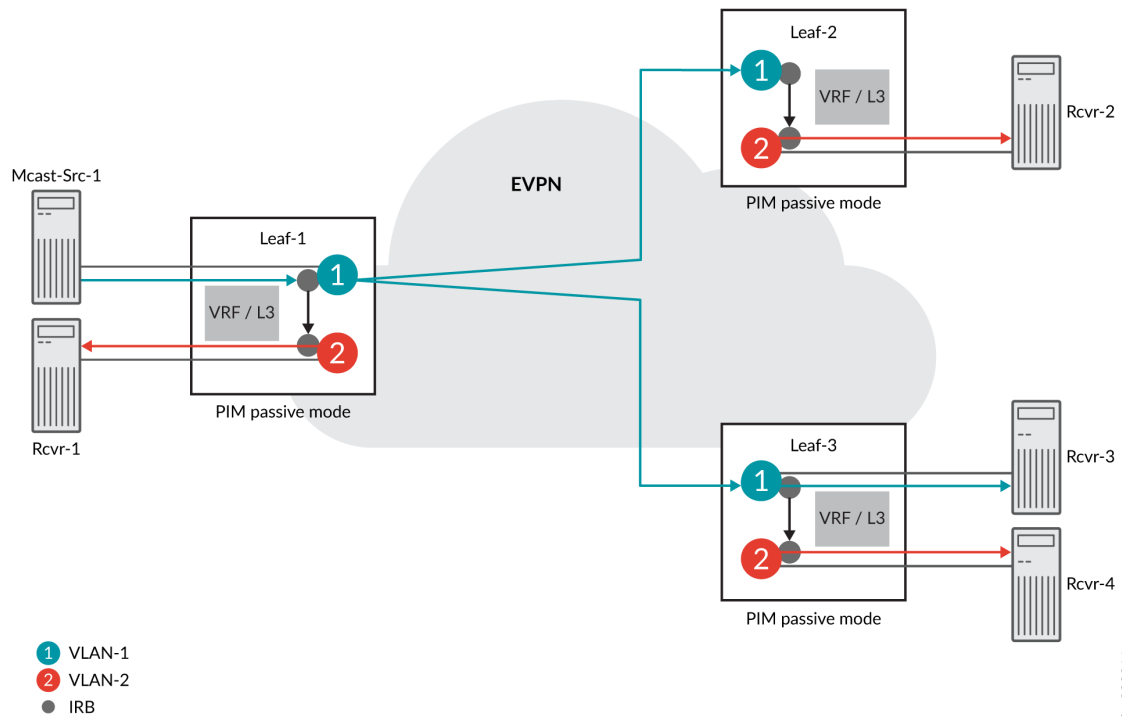
The use cases we support with enhanced OISM (the asymmetric bridge domains model) are similar to those in this section, but with a few operational differences. Also, as mentioned previously, you don't need to configure all VLANs on all leaf devices as the figures in this sections show.

For an overview of the differences with enhanced OISM, see ["Overview of Enhanced OISM" on page 1057](#). For more details on the operational differences, see ["How Enhanced OISM Works" on page 1112](#).

Local Routing on OISM Devices

In [Figure 103 on page 1089](#), we illustrate how local routing and forwarding works in general on OISM devices. As the figure shows, OISM local routing forwards the traffic on the source VLAN. Each leaf device routes the traffic locally to its receivers on other VLANs, which avoids hairpinning for intersubnet routing on the same device.

Figure 103: Local Routing with OISM



In this case, the source traffic comes from Mcast-Src-1 on VLAN-1, the blue VLAN. Server leaf devices use IRB interfaces and PIM in passive mode to route traffic between VLANs. With PIM in passive mode, server leaf devices:

- Don't become PIM neighbors with the other leaf devices.
- Act as a local PIM RP, create local PIM state upon receiving IGMP or MLD reports, and avoid doing source registration.

As a result, the server leaf devices forward and route multicast traffic within the fabric as follows to receivers interested in the multicast group:

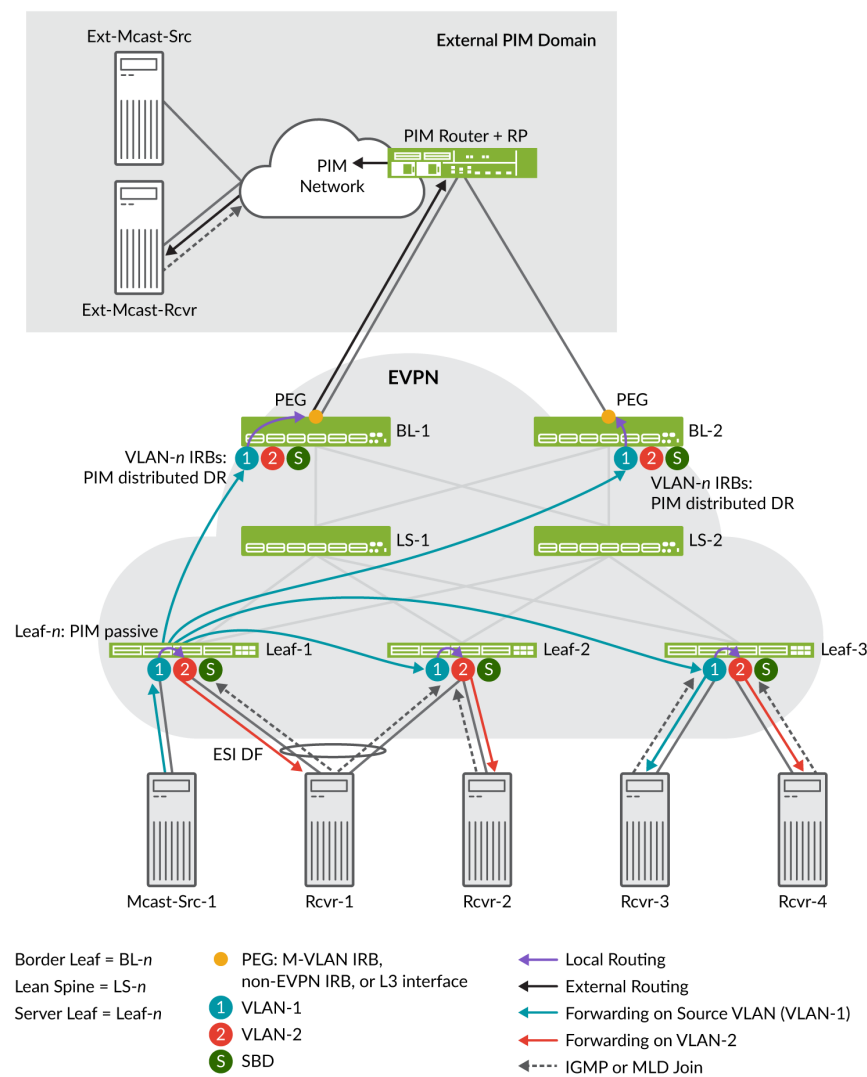
- The ingress leaf device (Leaf-1) forwards the traffic on the source VLAN into the EVPN fabric toward the other leaf devices with interested receivers.
- All of the server leaf devices don't need to forward the traffic back into the EVPN core to another device that is a designated router. Server leaf devices can locally:
 - Forward the traffic on the source VLAN toward local interested receivers on the source VLAN.
 - Route the traffic from the source VLAN through the IRB interfaces toward local interested receivers in other VLANs.

Multicast Traffic Forwarding and Routing with Source and Receivers Inside the EVPN Data Center

When the multicast source is inside the EVPN fabric, the server leaf devices receive the multicast traffic on the source VLAN. Then they locally route or forward the traffic as described in ["Local Routing on OISM Devices" on page 1088](#).

The following figure illustrates OISM local routing and forwarding within an EVPN fabric in detail. The figure also shows how local routing works with EVPN multihoming for a multicast receiver.

Figure 104: OISM with an Internal Multicast Source and Internal Multicast Receivers



In [Figure 104 on page 1090](#), the multicast source, Mcast-Src-1, is single-homed to Leaf-1. The source VLAN is VLAN-1 (the blue VLAN). Multicast control and data traffic flow proceeds as follows:

1. Receivers on all three server leaf devices sent IGMP or MLD reports (join messages) expressing interest in receiving the traffic for a multicast group.
2. Leaf-1 forwards the traffic on the source VLAN to both Leaf-2 and Leaf-3 because both leaf devices have interested receivers. In this case, the receivers on Leaf-2 and Leaf-3 use single-homing.
3. Leaf-2 and Leaf-3 forward or locally route the traffic to their interested receivers (Rcvr-2, Rcvr-3, and Rcvr-4) as described in ["Local Routing on OISM Devices" on page 1088](#).
4. Rcvr-1 on VLAN-2 is multihomed to Leaf-1 and Leaf-2 in an EVPN ES. Rcvr-1 has expressed interest in receiving the multicast traffic, so:
 - Both server leaf devices, Leaf-1 and Leaf-2, receive the IGMP or MLD report.
 - Both Leaf-1 and Leaf-2 locally route the traffic from the source VLAN (VLAN-1) because each device has the PIM passive mode configuration.
 - However, because Leaf-1 is the DF for the EVPN ES, only Leaf-1 forwards the traffic to Rcvr-1.
5. The border leaf devices receive the multicast traffic through the EVPN fabric on the source VLAN. Note that the border leaf devices could have local receivers, although we don't show that case. With local receivers, the device also locally routes or forwards the traffic to those receivers the same way the server leaf devices do.

[Figure 104 on page 1090](#) also shows that the border leaf devices locally route the traffic from the source VLAN toward any external multicast receivers in the external PIM domain. See ["Supported Methods for Multicast Data Transfer to or from an External PIM Domain" on page 1064](#) for the available external multicast methods by platform. Later sections describe the multicast control and data traffic flow for external source and external receiver use cases.

Multicast Traffic From an Internal Source to Receivers Outside the EVPN Data Center—M-VLAN IRB Method

In [Figure 105 on page 1092](#), we illustrate the OISM use case where a multicast source inside the EVPN fabric sends multicast traffic to an interested receiver outside the fabric using the M-VLAN IRB method for external multicast. (["Supported Methods for Multicast Data Transfer to or from an External PIM Domain" on page 1064](#) lists external multicast method support by platform.)

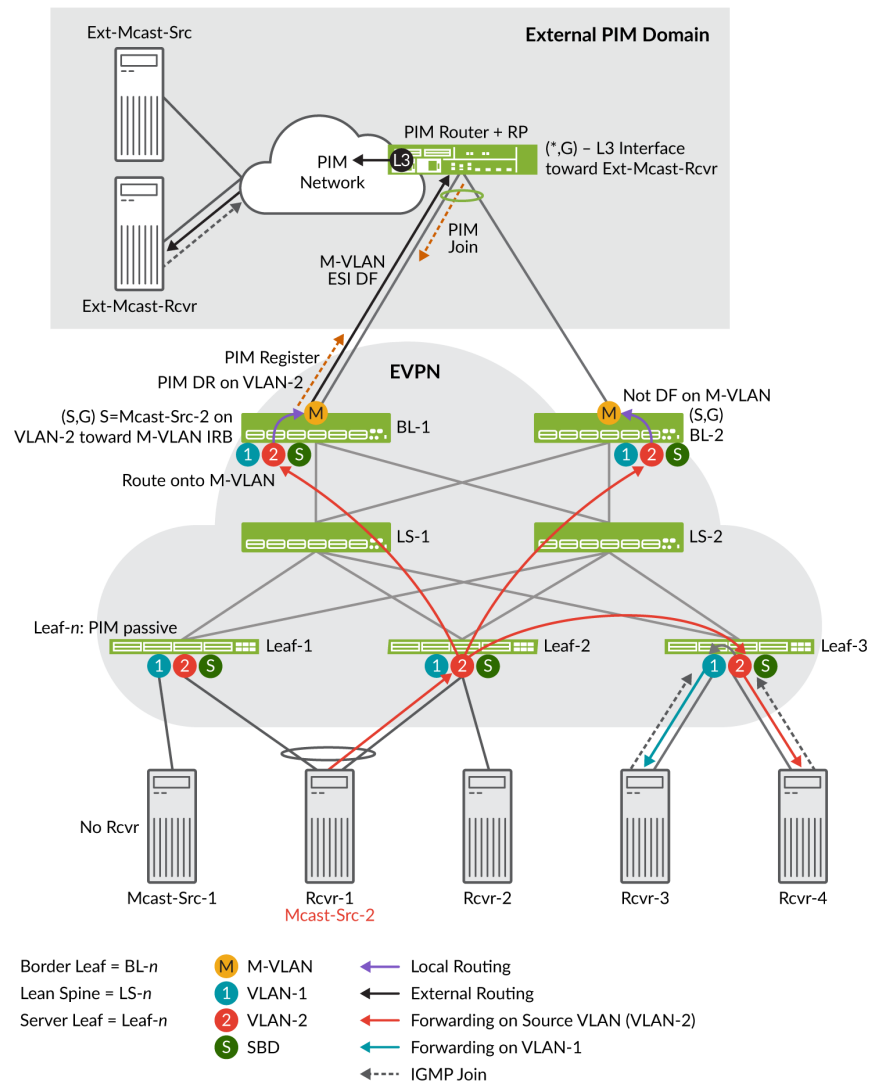
The border leaf devices you configure in the OISM PEG role receive the multicast traffic on the source VLAN through the EVPN core. Then the border leaf devices replicate the traffic and route it onto the M-VLAN toward the external PIM domain to reach the external receiver.



NOTE: PEG border leaf devices only send multicast source traffic received on the revenue bridge domains to the M-VLAN. These devices don't forward the traffic back into the EVPN core toward the other border leaf devices.

This use case also shows an internal multihomed multicast source and local routing to single-homed receivers inside the fabric.

Figure 105: OISM with an Internal Multicast Source and an External Multicast Receiver—M-VLAN IRB Method



In [Figure 105 on page 1092](#), the internal source for a multicast group is Mcast-Src-2, the same device as Rcvr-1, which is multihomed to Leaf-1 and Leaf 2. The source sends the multicast traffic on VLAN-2. The external receiver, Ext-Mcast-Rcvr, expresses interest in receiving the multicast traffic for that multicast group (sends a join message). Internal receivers Rcvr-3 (on VLAN-1) and Rcvr-4 (on VLAN-2) also request to join the multicast group and receive the traffic.

Note that the PIM router is multihomed to both BL-1 and BL-2, the PEG devices, in the EVPN fabric. Those connections are in the same ES; the DF election process chooses one of these devices as the DF for the ES. Only the DF will forward traffic (on the M-VLAN) toward external receivers.

The source traffic reaches the interested internal and external receivers as follows:

Traffic Flow from Multihomed Source to Internal Receivers

These steps summarize the multicast control and data traffic flow from the multihomed source to the internal receivers:

1. Mcast-Src-2 (also labeled Rcvr-1) originates the traffic on VLAN-2, the red VLAN. Because the device is multihomed to Leaf-1 and Leaf-2, the device hashes the traffic on VLAN-2 to one of those server leaf devices. In this case, Leaf-2 receives the traffic.
2. The red arrows show that Leaf-2 forwards the traffic on the source VLAN, VLAN-2, only to:
 - The other server leaf devices with interested receivers—In this case, only Leaf-3.
 - The border leaf devices, which both act in the OISM PEG role.

Note that no receivers behind Leaf-1 or Leaf-2 sent an IGMP report to join the multicast group. With IGMP snooping and SMET forwarding enabled, Leaf-2 doesn't forward the traffic to Leaf-1 because Leaf-1 has no interested receivers. Leaf-2 also doesn't locally route the traffic to Rcvr-2 for the same reason.

3. Leaf-3 receives the source traffic on VLAN-2. Then Leaf-3 routes the traffic locally to VLAN-1 to Rcvr-3. Leaf-3 also forwards the traffic to Rcvr-4 on VLAN-2.
4. Both border leaf devices BL-1 and BL-2 also receive the source traffic from the EVPN core. We describe the external multicast flow next.

Traffic Flow to External Receiver—M-VLAN IRB Method

These steps summarize the multicast control and data traffic flow in [Figure 105 on page 1092](#) from the border leaf devices toward the external receiver using the M-VLAN IRB method:

1. In the external PIM domain, the PIM RP enters a PIM (*,G) multicast routing table entry. The entry includes the L3 interface toward Ext-Mcast-Rcvr as the downstream interface.

2. Both border leaf devices BL-1 and BL-2 receive the source traffic from the EVPN core. The IRB interface on VLAN-2 on one of these border leaf devices is the PIM DR for VLAN-2. In this case, the PIM DR is on BL-1, so BL-1 sends a PIM Register message toward the PIM RP on the M-VLAN IRB interface.
3. The PIM RP sends a PIM Join message back toward BL-1. BL-1 creates an (S,G) multicast routing table entry as follows:
 - The source address is the IP address of Mcast-Src-2 in VLAN-2.
 - The downstream interface is the M-VLAN IRB interface.
4. Both BL-1 and BL-2 are PEG devices and configured in PIM distributed DR mode for the revenue bridge domain (VLAN-1 and VLAN-2) IRB interfaces. As a result, both BL-1 and BL-2 receive the PIM Join and create a similar (S,G) state. Both devices route the traffic locally from VLAN-2 to the M-VLAN.

However, only the DF for the M-VLAN ES actually forwards the data on the M-VLAN to the external PIM domain. In this case, BL-1 is the DF and sends the traffic toward the external receiver. (See the label "M-VLAN ESI DF" and the black arrow between BL-1 and the PIM router in [Figure 105 on page 1092](#).)

5. The PIM RP receives the traffic from the OISM M-VLAN IRB interface connection. The PIM router sends the traffic to an L3 interface toward the external receiver.

Multicast Traffic from an Internal Source to Receivers Outside the EVPN Data Center—L3 Interface Method or Non-EVPN IRB Method

In [Figure 106 on page 1096](#), we illustrate the OISM use case where a multicast source inside the EVPN fabric sends multicast traffic to a receiver outside the fabric using either of the following methods for external multicast:

- Classic L3 interface external multicast method:

On each border leaf device, you configure a classic L3 interface with family `inet` that connects to the external PIM router. You assign an IP address to that interface on a different subnet than the L3 interface subnets on other border leaf devices in the fabric.

You enable PIM on the interface and include the interface in the tenant VRF instances that have multicast data receivers. This method differs from the M-VLAN IRB method because you don't extend this interface in the EVPN instance.



NOTE: The L3 interface connection here can be an individual physical interface or an AE interface bundle that includes multiple physical L3 interfaces.

- Non-EVPN IRB external multicast method:

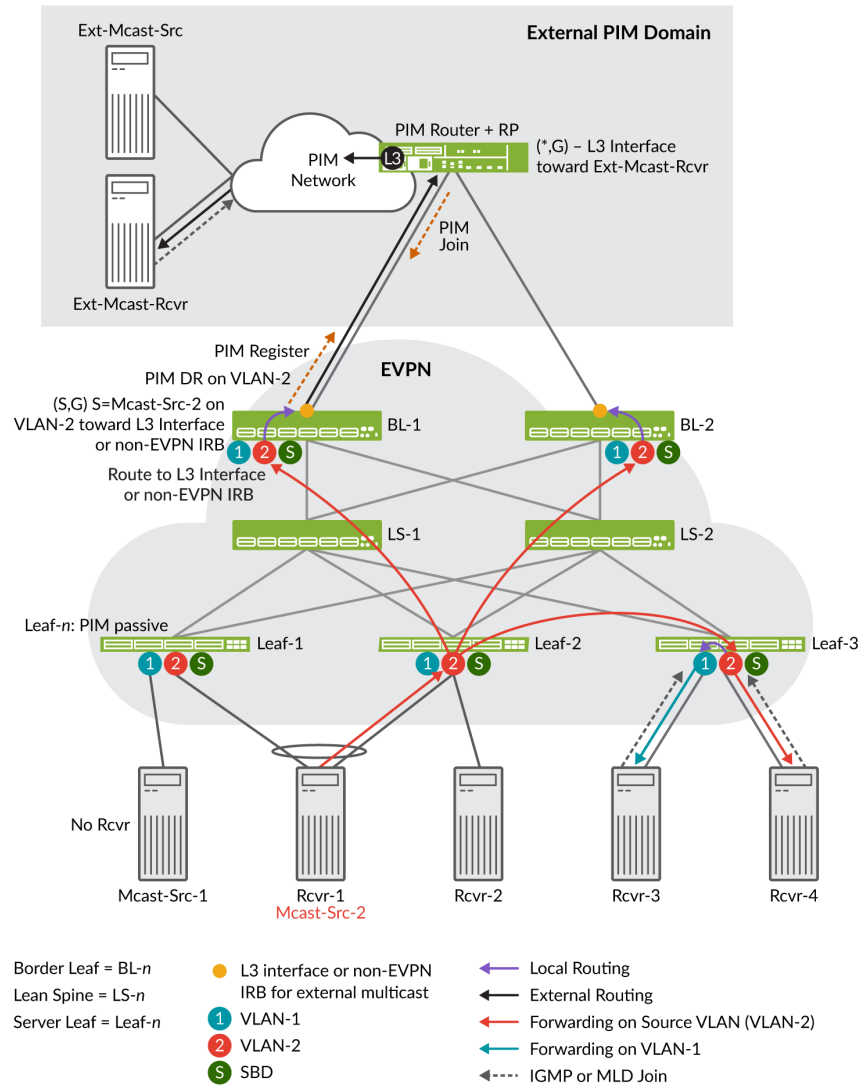
On each border leaf device, you configure a unique extra VLAN that is only for external multicast. You also configure a corresponding L3 IRB interface with an IP address that connects to the external PIM router. The extra VLAN ID can't be the same as the VLAN ID of the revenue bridge domains, SBD, or extra VLAN on any other border leaf device in the fabric. In addition, similar to the L3 interface method, the non-EVPN IRB interfaces on different border leaf devices should connect to the PIM router on different subnets in the fabric.

You enable PIM on the IRB interface and include the interface in the tenant VRF instances that have multicast data receivers. This method differs from the M-VLAN IRB method because you don't extend this VLAN or IRB interface in the EVPN instance.

["Supported Methods for Multicast Data Transfer to or from an External PIM Domain" on page 1064](#) lists the platforms that support these external multicast methods.

[Figure 106 on page 1096](#) includes the same internal multihomed source, internal receivers, and an external receiver as in ["Multicast Traffic From an Internal Source to Receivers Outside the EVPN Data Center—M-VLAN IRB Method" on page 1091](#). See the steps in ["Traffic Flow from Multihomed Source to Internal Receivers" on page 1093](#) for details on the internal multicast traffic flow. In this section we describe only what's different in this case, which is the multicast traffic flow from the border leaf devices to the external receiver.

Figure 106: OISM with an Internal Multicast Source and an External Multicast Receiver—L3 Interface or Non-EVPN IRB Method



In [Figure 106 on page 1096](#), the internal source for a multicast group is Mcast-Src-2, which is multihomed to Leaf-1 and Leaf-2. The source sends the multicast traffic on VLAN-2, the red VLAN. The external receiver, Ext-Mcast-Rcvr, expresses interest in receiving the multicast traffic for that multicast group (sends a join message).

The external multicast flow in this use case is very similar to the M-VLAN IRB use case. The border leaf devices in the OISM PEG role receive the multicast traffic on the source VLAN through the EVPN core. However, the main difference in this case is that the external multicast interfaces don't use EVPN signaling and don't share an ESI across the border leaf devices. The external multicast interfaces on each border leaf device are distinct, and each have L3 reachability to the external PIM gateway router. The

border leaf device that establishes the PIM join state replicates and sends the traffic on the L3 interface or non-EVPN IRB interface to the external PIM domain with the external receiver.



NOTE: PEG border leaf devices only send multicast source traffic received on the revenue bridge domains to the external PIM domain. These devices don't forward the traffic back into the EVPN core toward the other border leaf devices.

The following section explains how the source traffic reaches the interested external receiver.

Traffic Flow to External Receiver—L3 Interface or Non-EVPN IRB Method

These steps summarize the multicast control and data traffic flow in [Figure 106 on page 1096](#) from the border leaf devices toward the external receiver using the classic L3 interface method or the non-EVPN IRB method:

1. In the external PIM domain, the PIM RP enters a PIM (*,G) multicast routing table entry. The entry includes the L3 interface toward Ext-Mcast-Rcvr as the downstream interface.
2. Both border leaf devices BL-1 and BL-2 receive the source traffic from the EVPN core on VLAN-2. The IRB interface on VLAN-2 on one of these border leaf devices is the PIM DR for VLAN-2. In this case, the PIM DR is on BL-1, so BL-1 sends a PIM Register message toward the PIM RP on its external multicast L3 interface or non-EVPN IRB interface.
3. The PIM RP sends a PIM Join message back toward BL-1. BL-1 receives the PIM join and creates an (S,G) multicast routing table entry as follows:
 - The source address is the IP address of Mcast-Src-2 in VLAN-2.
 - The downstream interface is the external multicast L3 interface or the non-EVPN IRB interface.
4. BL-1 routes the traffic from VLAN-2 to its external multicast L3 interface or non-EVPN IRB interface.
5. The PIM RP receives the traffic from BL-1 on the external multicast interface. The PIM router sends the traffic to an L3 interface toward the external receiver. The external receiver receives the multicast traffic.

Multicast Traffic from an External Source to Receivers Inside the EVPN Data Center—M-VLAN IRB Method

[Figure 107 on page 1099](#) illustrates the OISM use case where a multicast source outside the EVPN fabric sends multicast traffic to receivers inside the fabric. This use case exhibits the two main ways OISM uses the SBD in the EVPN core:

- To carry external multicast source traffic.

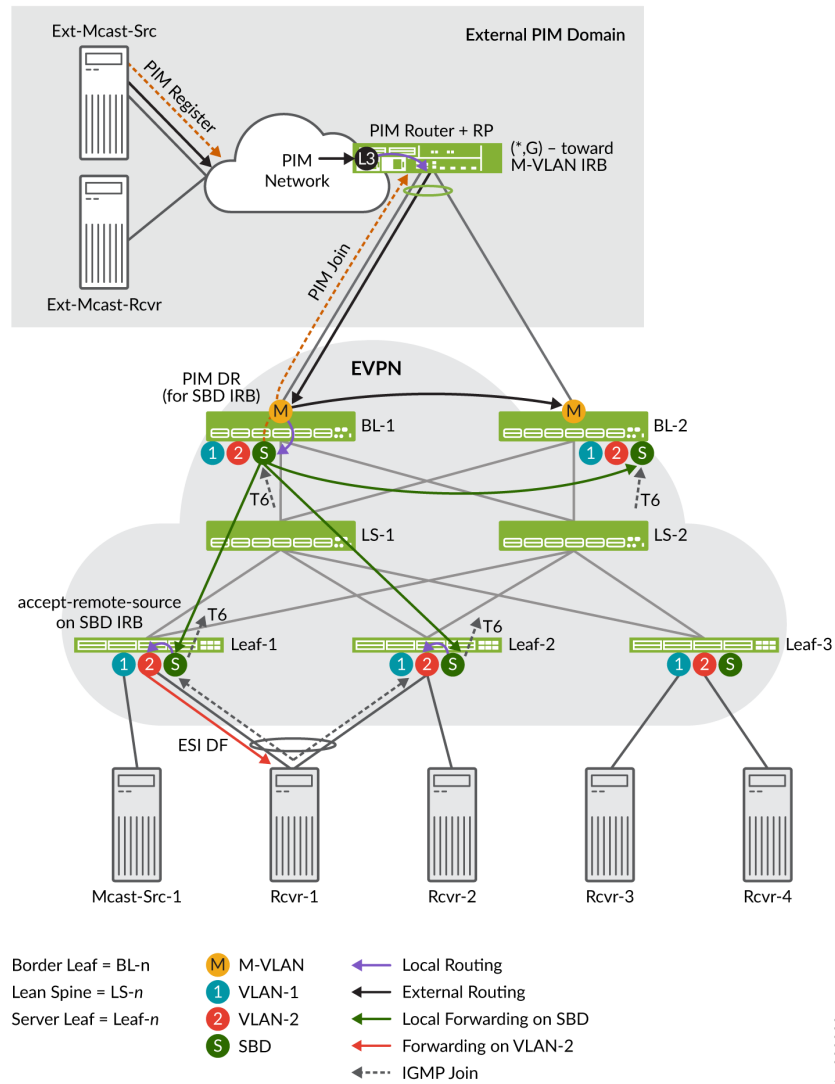
- To advertise SMET Type 6 routes.

The Type 6 routes ensure that the border leaf devices only forward the traffic toward the EVPN devices with interested receivers.

OISM border leaf devices receive external multicast source traffic through the M-VLAN IRB interfaces. OISM devices use the SBD to forward traffic toward EVPN server leaf devices with interested receivers on the revenue bridge domains. Each leaf device then locally forwards or routes the traffic on the revenue bridge domains to its local receivers.

This use case has an internal receiver that is multihomed to two server leaf devices.

Figure 107: OISM with an External Multicast Source and an Internal Multihomed Multicast Receiver—M-VLAN IRB Method



In [Figure 107 on page 1099](#), Rcvr-1 within the EVPN fabric is multihomed to server leaf devices Leaf-1 and Leaf-2. Rcvr-1 expresses interest in receiving traffic from a multicast group. The source for the multicast traffic for the group is Ext-Mcast-Src in the external PIM domain.

The external source traffic reaches the interested multihomed receiver, Rcvr-1, as follows:

Multicast Control Flow between the Internal Multihomed Receiver and the External Source—M-VLAN IRB Method

These steps summarize the multicast control flow in this use case:

1. Rcvr-1 sends an IGMP join message to both multihoming peers Leaf-1 and Leaf-2.
2. Both Leaf-1 and Leaf-2 generate an EVPN Type 6 route toward the EVPN core on the SBD. The Type 6 (SMET) route advertises that Rcvr-1 is interested in the multicast data.
3. Both border leaf devices BL-1 and BL-2 receive the Type 6 route on the SBD.
4. The Type 6 route (on the SBD) signals the border leaf devices to create a PIM join toward the PIM RP (reachable through the M-VLAN). However, to avoid duplicate join messages, only the border leaf device that is the PIM DR for the SBD generates the PIM join message. In this case, the figure shows the PIM DR for the SBD is BL-1. BL-1 sends the PIM join message toward the PIM RP by way of its neighbor, the M-VLAN IRB interface.
5. The PIM RP receives the join message. Then the PIM RP creates a PIM (*,G) entry in the multicast routing table with the M-VLAN IRB interface as the downstream interface.
6. The external source Ext-Mcast-Src registers with the PIM RP. The PIM RP has a multicast route for the group with the M-VLAN IRB interface as the downstream interface. As a result, the PIM RP routes the multicast traffic coming in at L3 onto its connection to the M-VLAN IRB toward BL-1 or BL-2. In this case, BL-1 sent the PIM join, so BL-1 receives the traffic on its M-VLAN IRB interface.

Traffic Flow from the Border Leaf Devices to Internal Receivers—M-VLAN IRB Method

In [Figure 107 on page 1099](#), BL-1 is the PIM DR for the SBD and sent the PIM join toward the external PIM domain. BL-1 receives and routes (or forwards) the external source traffic as follows:

1. BL-1 routes the traffic locally from the M-VLAN to the SBD on its SBD IRB interface because BL-1 is the PIM DR for the SBD. See the small gray arrow from the M-VLAN to the SBD on BL-1.
2. BL-1 forwards a copy of the traffic *on the M-VLAN* toward BL-2 because both border leaf devices are in the PEG role. See the black arrow from BL-1 toward BL-2.

As a PEG device using the M-VLAN IRB method, BL-2 expects to receive external multicast traffic only on the M-VLAN IRB interface. If BL-2 has any local receivers, BL-2 can receive the traffic and route it locally to those receivers.

3. BL-1 *also* forwards a copy of the traffic *on the SBD* into the EVPN core to BL-2. See the green arrow from BL-1 toward BL-2.

BL-2 drops the traffic because again, as a PEG device using the M-VLAN IRB method, BL-2 expects to receive external source traffic only on the M-VLAN IRB interface. BL-2 doesn't expect external source traffic on the SBD IRB interface from BL-1. In other words, BL-2 sees this case as a source interface mismatch (a reverse path forwarding [RFP] failure).



NOTE: One reason why the ingress border leaf device also forwards a copy on the SBD to other border leaf devices is to ensure that another border leaf device can receive external source traffic if its M-VLAN interface is down. Then any interested local receivers on the other border leaf device can still get the traffic.

4. BL-1 selectively forwards copies of the traffic on the SBD to the server leaf devices with interested receivers, based on the advertised Type 6 routes.

In this case, Leaf-1 and Leaf-2 have a multihomed interested receiver, Rcvr-1, on VLAN-2. As a result, BL-1 sends the traffic toward *both* leaf devices. See the green arrows from BL-1 toward Leaf-1 and Leaf-2.



NOTE: In a similar use case with the PIM router multihomed to BL-1 and BL-2, BL-1 might receive the external multicast source traffic, but BL-2 is the PIM DR on the SBD. One reason why BL-1 forwards the incoming external multicast traffic toward BL-2 on the M-VLAN is so that BL-2 can handle this use case. See the black arrow from BL-1 toward BL-2 on the M-VLAN in [Figure 107 on page 1099](#). If BL-2 is the PIM DR on the SBD, upon receiving the traffic on the M-VLAN from BL-1, BL-2 forwards the traffic on the SBD toward Leaf-1 and Leaf-2. In this case, the green arrows in the figure would flow from BL-2 toward the other EVPN devices instead of flowing from BL-1.

5. Leaf-1 and Leaf-2 locally route the traffic from the SBD IRB interface to the revenue bridge domain IRB interface for VLAN-2 toward the interested (multihomed) receiver. However, with EVPN multihoming, only the EVPN DF in the ES forwards the traffic toward Rcvr-1 so Rcvr-1 doesn't get duplicate traffic.

In this case, Leaf-1 is the EVPN DF, so only Leaf-1 forwards the traffic to Rcvr-1.

What Happens When a Multihomed External PIM Router Load-Balances the Traffic—M-VLAN IRB Method

In [Figure 107 on page 1099](#), the external PIM gateway router is multihomed to BL-1 and BL-2 on an ES in the EVPN fabric. If the pair of connections on the PIM router side is an AE interface bundle, the PIM router does load balancing among the interfaces in the bundle. In that case, both BL-1 and BL-2 will each receive some of the multicast traffic flow from the external source. However, all receivers should receive all of that traffic. For simplicity, the figure doesn't show traffic arrows for this load balancing, but we'll describe the flow here.

BL-1 and BL-2 each receive part of the external multicast source traffic on their M-VLAN IRB interfaces. However, because BL-1 is the PIM DR on the SBD, only BL-1 will route the traffic onto the SBD into the EVPN fabric, as follows:

1. BL-1 routes the traffic it receives onto the SBD toward the server leaf devices.

BL-1 also forwards that traffic on the M-VLAN and routes it on the SBD toward BL-2, in case BL-2 has any local receivers (as described in ["Traffic Flow from the Border Leaf Devices to Internal Receivers—M-VLAN IRB Method" on page 1100](#)).

2. BL-2 forwards the traffic it receives from the external PIM domain on the M-VLAN toward BL-1, because BL-1 only expects to receive external source traffic on the M-VLAN.

Due to DF and split horizon rules, BL-2 won't forward any traffic it receives on the M-VLAN from BL-1 into the EVPN core or back toward the source, BL-1.

3. BL-1 routes the traffic it receives on the M-VLAN from BL-2 onto the SBD toward the server leaf devices.

What Happens with Local Receivers on Border Leaf Devices—M-VLAN IRB Method

[Figure 107 on page 1099](#) doesn't show local receivers attached to the border leaf devices. However, let's look briefly at PIM join message flow and how external source traffic reaches local receivers on border leaf devices.

Consider that BL-1 or BL-2 has an interested receiver on a revenue bridge domain in the fabric. In that case:

1. *Both* devices generate a PIM join on the IRB interfaces for the *revenue bridge domain* toward the PIM RP.
2. You configure the border leaf devices with PIM in distributed DR mode on the revenue bridge domain IRB interfaces. That way, neither BL-1 nor BL-2 acts as the PIM DR alone. Both devices locally route external multicast source traffic coming in on the M-VLAN IRB interface to the appropriate revenue bridge domain IRB interface.

Multicast Traffic from an External Source to Receivers Inside the EVPN Data Center—L3 Interface Method or Non-EVPN IRB Method

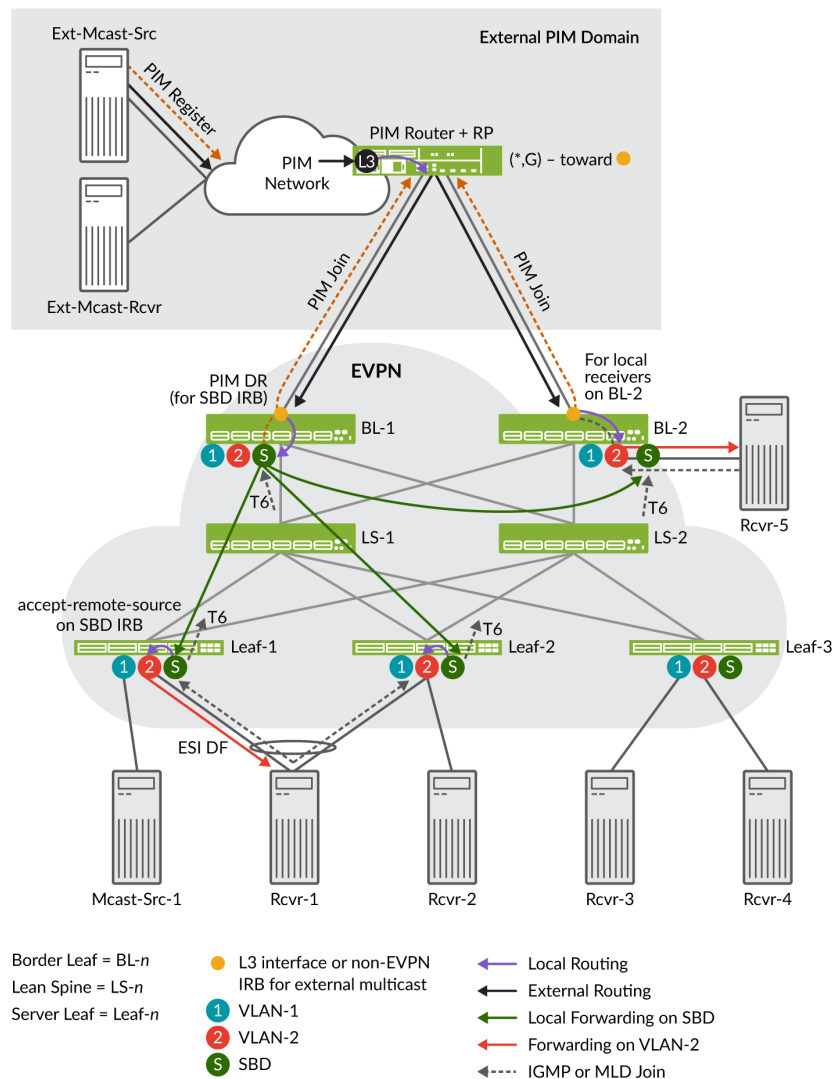
[Figure 108 on page 1103](#) illustrates the OISM use case where a multicast source outside the EVPN fabric sends multicast traffic to receivers inside the fabric. In this case, the fabric uses the classic L3 interface or non-EVPN IRB external multicast method to connect to the external PIM domain. Also, this case includes the following internal receivers:

- A receiver that is multihomed to two server leaf devices.
- A local receiver on one of the border leaf devices.

This use case, like the M-VLAN IRB external source use case in [Figure 107 on page 1099](#), shows the two main ways OISM uses the SBD in the EVPN core—To carry external multicast source traffic and to

advertise SMET Type 6 routes. The Type 6 routes ensure that the border leaf devices only forward the traffic toward the EVPN devices with interested receivers.

Figure 108: OISM with an External Multicast Source and an Internal Multihomed Multicast Receiver—L3 Interface or Non-EVPN IRB Method



In [Figure 108 on page 1103](#):

- Rcvr-1 is multihomed to server leaf devices Leaf-1 and Leaf-2 in the EVPN fabric, and expresses interest in receiving traffic from a multicast group.
- Rcvr-5 on BL-2 in the EVPN fabric is also interested in receiving the multicast traffic.

- Ext-Mcast-Src in the external PIM domain is the source for the traffic for the multicast group.

The multicast control flow and data traffic flow for external multicast are similar for the classic L3 interface and the non-EVPN IRB interface methods. As a result, in this section we commonly say *external multicast interface* when we refer to the border leaf device external connection points.

The external source traffic reaches the interested receivers (Rcvr-1 and Rcvr-5) as follows:

Multicast Control Flow between the Internal Receivers and the External Source—L3 Interface or Non-EVPN IRB Method

These steps summarize the multicast control flow in this use case:

1. Rcvr-1 sends an IGMP or MLD join message on VLAN-2 to both multihoming peers Leaf-1 and Leaf-2.
2. Both Leaf-1 and Leaf-2 generate an EVPN Type 6 route toward the EVPN core on the SBD. The Type 6 (SMET) route advertises that Rcvr-1 is interested in the multicast data.
3. Both border leaf devices BL-1 and BL-2 receive the Type 6 route on the SBD.
4. The Type 6 route (on the SBD) signals the border leaf devices to create a PIM join toward the PIM RP (reachable through the external multicast interface). However, to avoid duplicate join messages *for server leaf devices on the SBD*, only the border leaf device that is the PIM DR for the SBD generates the PIM join message. In this case, the figure shows the PIM DR for the SBD is BL-1. BL-1 sends the PIM join message toward the PIM RP by way of its PIM neighbor, the external multicast interface.
5. The local receiver on BL-2, Rcvr-5, also sends an IGMP or MLD join message on VLAN-2 to BL-2. Note that in this case, BL-1 and BL-2 are not multihoming peers in an EVPN ES. As a result, BL-2 sends a separate PIM join message on its external multicast interface because it has a local interested receiver (Rcvr-5).
6. The PIM RP receives the join messages. The PIM RP creates PIM (*,G) entries in the multicast routing table with the BL-1 and BL-2 external multicast interfaces as downstream interfaces.
7. The external source Ext-Mcast-Src registers with the PIM RP. The PIM RP has multicast routes for the group with the BL-1 and BL-2 external multicast interfaces as downstream interfaces. As a result, the PIM RP routes the multicast traffic coming in at L3 toward BL-1 and BL-2.

Both BL-1 and BL-2 receive the multicast traffic. The next section explains how the border leaf devices forward or route the traffic in the EVPN fabric.

Traffic Flow from the Border Leaf Devices to Internal Receivers—L3 Interface or Non-EVPN IRB Method

In [Figure 108 on page 1103](#), BL-1 is the PIM DR for the SBD and sent the PIM join message toward the external PIM domain *for server leaf devices on the SBD*. BL-2 also sent a PIM join message toward the external PIM domain *for its interested local receiver*.

BL-1 and BL-2 receive the external source traffic, and route (or forward) it as follows:

1. BL-1 routes the traffic locally from the external multicast interface to the SBD IRB interface because BL-1 is the PIM DR for the SBD. See the small gray arrow from the external multicast interface to the SBD on BL-1.
2. BL-1 forwards a copy of the traffic *on the SBD* into the EVPN core to BL-2. See the green arrow from BL-1 toward BL-2.

However, BL-2 drops the traffic from the SBD because, as a PEG device using the classic L3 interface or non-EVPN IRB method, BL-2 doesn't expect external source traffic on the SBD IRB interface from BL-1. If BL-2 has interested receivers, it would have sent a PIM join message and should receive the same traffic from its external multicast connection.



NOTE: One reason why the ingress border leaf device also forwards a copy on the SBD to other border leaf devices is to ensure that another border leaf device can receive external source traffic if its external multicast interface is down. Then any interested local receivers on the other border leaf device can still get the traffic.

3. BL-2 routes the external multicast traffic to its local receiver, Rcvr-5, on VLAN-2. See the small gray arrow on BL-2 from the external multicast interface to VLAN-2.



NOTE: The border leaf devices configured in PEG mode that are not the PIM DR on the SBD will still locally route the traffic received from the external multicast interface. These devices don't send traffic from external multicast sources to other PEG border leaf devices on the SBD. These devices also don't forward the traffic on the SBD into the EVPN core.

4. BL-1 (the PIM DR on the SBD) selectively forwards copies of the traffic on the SBD to the server leaf devices with interested receivers (based on the advertised Type 6 routes). See the green arrows from BL-1 toward Leaf-1 and Leaf-2.

In this case, Leaf-1 and Leaf-2 have a multihomed interested receiver, Rcvr-1, on VLAN-2. As a result, BL-1 sends the traffic on the SBD toward *both* leaf devices.

5. Leaf-1 and Leaf-2 locally route the traffic from the SBD IRB interface to the revenue bridge domain IRB interface for VLAN-2 toward the interested (multihomed) receiver. However, with EVPN multihoming, only the EVPN DF in the ES forwards the traffic toward Rcvr-1 so Rcvr-1 doesn't get duplicate traffic.

In this case, Leaf-1 is the EVPN DF, so only Leaf-1 forwards the traffic to Rcvr-1. See the red arrow from Leaf-1 toward Rcvr-1.

AR and OISM with an Internal Multicast Source

In [Figure 109 on page 1107](#), we show an OISM use case where you configure the spine devices as standalone AR replicator devices. The OISM server leaf and border leaf devices are AR leaf devices. The AR replicator devices handle replicating the multicast traffic for the OISM server leaf and border leaf devices. This case shows a multicast source and single-homed receivers behind server leaf devices inside the EVPN fabric.

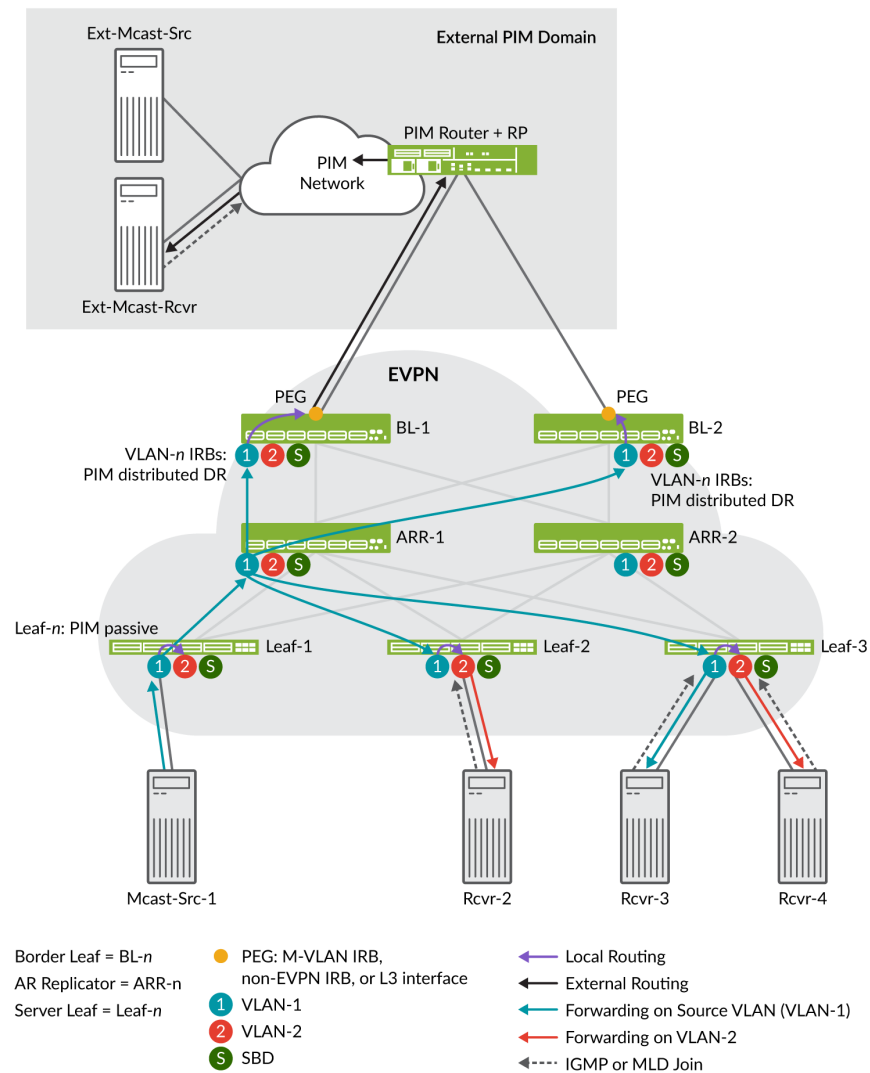


NOTE: AR behavior is different when the multicast source is behind a server leaf device that also has a multihomed receiver; see ["AR and OISM with an Internal Multicast Source and Multihomed Receiver" on page 1108](#) for more on the behavior with that use case.

With OISM traffic from an internal source, the ingress device forwards the traffic in the EVPN fabric on the source VLAN. When you also enable AR, the ingress leaf device forwards one copy of the traffic to an AR replicator. The AR replicator replicates the traffic and sends the copies on the source VLAN to the other leaf devices with interested receivers. Then each leaf device:

- Locally forwards the traffic toward its receivers on the source VLAN.
- Locally routes the traffic toward its receivers on the other revenue VLANs.

Figure 109: AR with OISM—Internal Multicast Source and Single-Homed Internal Receivers



In the use case in [Figure 109 on page 1107](#):

1. Rcvr-2, Rcvr-3 and Rcvr-4 send IGMP or MLD reports to join the multicast group.
2. The traffic source for the multicast group, Mcast-Src-1, forwards the traffic on the source VLAN, VLAN-1, to Leaf-1.
3. Leaf-1 forwards the traffic to one of the available AR replicators on VLAN-1 to replicate the traffic to the other leaf devices with interested receivers. In this case, Leaf-1 forwards the traffic to ARR-1.



NOTE: See ["AR Leaf Device Load Balancing with Multiple Replicators"](#) on page 1018 for details on how AR leaf devices load-balance among multiple available AR replicators.

4. ARR-1 replicates the traffic and sends copies on VLAN-1 toward all the leaf devices with interested receivers.
5. Each server leaf device:

- Forwards the traffic toward the interested receivers on VLAN-1.
- Locally routes the traffic to VLAN-2 and forwards it toward the interested receivers on VLAN-2.

Also note that in [Figure 109 on page 1107](#), Rcvr-1 is multihomed to Leaf-1 and Leaf-2, with Leaf-1 as the ESI DF. As a result, only Leaf-1 forwards the traffic to Rcvr-1 on VLAN-2.

6. If any external receivers expressed interest in receiving the traffic, the border leaf devices locally route the traffic to the external multicast interface. The external multicast interface sends the traffic toward any interested external receivers based on the external multicast method you configure.

See ["Configure Assisted Replication"](#) on page 1038 for details on how to configure AR.

AR and OISM with an Internal Multicast Source and Multihomed Receiver

In [Figure 110 on page 1109](#), we show an OISM use case similar to the setup in ["AR and OISM with an Internal Multicast Source"](#) on page 1106. However, in this case, the multicast source is behind a server leaf device that also has a multihomed receiver. In this case, AR operates in extended AR mode by default to efficiently support the multihomed receiver. See ["Extended AR Mode for Multihomed Ethernet Segments"](#) on page 1017 for full details on this mode.

Here is a summary of how multicast traffic ingressing on a server leaf device reaches a multihomed receiver in this case:

- The ingress server leaf device with an ESI for a multihomed receiver maintains a list of its multihoming peer leaf devices on the ES.

The AR replicator device also knows which AR leaf devices have multihoming peers.

- The ingress server leaf device takes care of replicating and forwarding the multicast traffic to any of its multihoming peers that are interested in the traffic.

The ingress leaf device also sends one copy to an AR replicator device to handle replication and forwarding to any other leaf devices.

Other than this difference in handling replication to the multihoming peers, the traffic flow to an AR replicator and then to the interested receivers is the same as we describe in ["AR and OISM with an Internal Multicast Source"](#) on page 1106.

4. Leaf-1 also forwards the traffic to one of the available AR replicators, ARR-1 in this case, on the source VLAN, VLAN-1.



NOTE: See ["AR Leaf Device Load Balancing with Multiple Replicators" on page 1018](#) for details on how AR leaf devices load-balance among multiple available AR replicators.

5. ARR-1 replicates the traffic and sends copies on VLAN-1 only toward the other leaf devices with interested receivers besides Leaf-2. Due to the default extended AR mode behavior (see Step "3" on [page 1109](#) above), ARR-1 skips sending the traffic to Leaf-2, the multihoming peer of the ingress leaf device Leaf-1.
6. Each server leaf device then forwards or routes the traffic to its interested receivers.

Note that in [Figure 110 on page 1109](#), Leaf-1 is the ESI DF for multihomed receiver Rcvr-2. As a result, only Leaf-1 forwards the traffic to Rcvr-1 on VLAN-2.

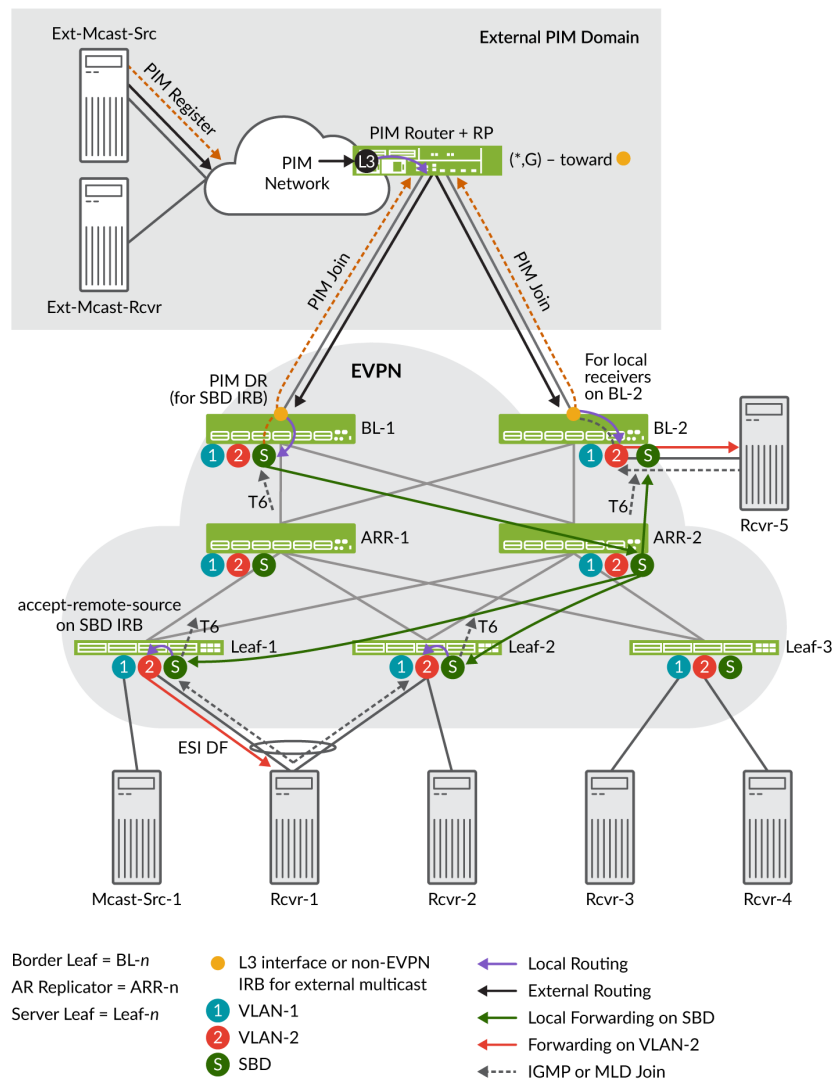
See ["Configure Assisted Replication" on page 1038](#) for details on how to configure AR.

AR and OISM with an External Multicast Source

In [Figure 111 on page 1111](#), we show an OISM use case where you configure the spine devices as standalone AR replicator devices. The OISM server leaf and border leaf devices are AR leaf devices. The multicast source is outside the EVPN fabric in an external PIM domain. The border leaf devices in this case use the classic L3 interface method to connect to the PIM router and PIM RP.

With OISM traffic from an external source, the ingress border leaf device forwards the traffic in the EVPN fabric on the SBD VLAN. When you also enable AR, the ingress border leaf device forwards one copy of the traffic to an AR replicator. The AR replicator replicates the traffic and sends the copies on the SBD VLAN to the other leaf devices with interested receivers. Then each device locally routes the traffic it receives on the SBD toward its receivers on the revenue bridge domain VLANs.

Figure 111: AR with OISM-External Multicast Source



In the use case in [Figure 111 on page 1111](#):

1. Rcvr-1 (multihomed to Leaf-1 and Leaf-2) and Rcvr-5 (a local host behind BL-2) send IGMP or MLD reports to join the multicast group.
2. The external source, Ext-Mcast-Src, sends multicast traffic through the external PIM domain. In this case we use the classic L3 interface external multicast method, and both devices sent a PIM join message, so the PIM router sends the traffic to both BL-1 and BL-2. (See ["Multicast Traffic from an External Source to Receivers Inside the EVPN Data Center—L3 Interface Method or Non-EVPN IRB Method" on page 1102](#) for a full explanation of this behavior.)



NOTE: As [Figure 111 on page 1111](#) shows, in this use case, because BL-2 has a local receiver, BL-2 routes the incoming externally sourced traffic directly toward its receiver on VLAN-2. BL-2 doesn't route traffic to the local receiver that it receives on the SBD from ARR-2 because its reverse-path forwarding toward the external source refers to the L3 interface.

BL-2 also doesn't route the traffic to the SBD because BL-1 is the PIM DR on the SBD (see the next step).

3. BL-1 is the PIM DR for the SBD, so BL-1 is the border leaf device that routes the externally sourced traffic into the EVPN fabric. With AR enabled, BL-1 forwards the traffic on the SBD to one of the available AR replicators. In this case, BL-1 forwards the traffic to ARR-2.



NOTE: See ["AR Leaf Device Load Balancing with Multiple Replicators" on page 1018](#) for details on how AR leaf devices load-balance among multiple AR replicators.

4. ARR-2 replicates the traffic and sends copies on the SBD toward the leaf devices with interested receivers—in this case, Leaf-1, Leaf-2, and BL-2.
5. Each leaf device that receives the traffic on the SBD locally routes the traffic toward the interested receivers on the revenue VLANs. In this case:
 - BL-2 routes the traffic toward its receiver on VLAN-2.
 - Both Leaf 1 and Leaf-2 receive the traffic on the SBD. Rcvr-1 is multihomed to Leaf-1 and Leaf-2, and Leaf-1 is the ESI DF. As a result, only Leaf-1 forwards the traffic toward Rcvr-1 on VLAN-2.

See ["Configure Assisted Replication" on page 1038](#) for details on how to configure AR.

How Enhanced OISM Works

IN THIS SECTION

- [Local Routing and East-West Traffic Differences with Enhanced OISM | 1113](#)
- [PIM Registration with Enhanced OISM for Internal Sources Based on EVPN Type 10 S-PMSI A-D Routes | 1115](#)

The use cases we support with enhanced OISM (the asymmetric bridge domains model) are similar to those we describe in ["How OISM Works" on page 1088](#), but with a few operational differences. Also, as

mentioned previously, you don't need to configure all VLANs on all leaf devices the way you do with regular OISM.

See ["Overview of Enhanced OISM" on page 1057](#) for a brief introduction to enhanced OISM mode differences compared to regular OISM mode. This section describes the main operational differences in more detail.

Local Routing and East-West Traffic Differences with Enhanced OISM

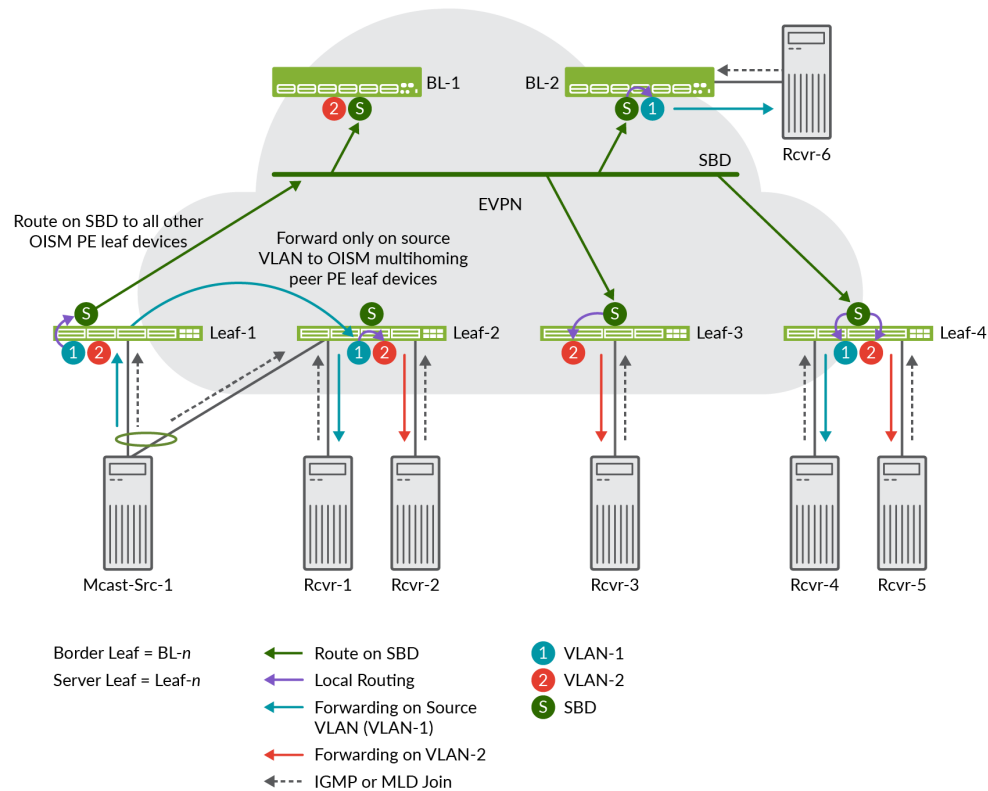
With enhanced OISM, the OISM leaf devices perform local routing the same way as we describe for regular OISM in ["Local Routing on OISM Devices" on page 1088](#). However, to send the traffic to other OISM leaf devices that are not its multihoming peers, an enhanced OISM ingress leaf device routes source traffic on the SBD instead of forwarding it on the source VLAN. The receiving leaf devices then locally route the traffic from the SBD to the destination VLAN.

For an ingress leaf device that has multihoming peers (other OISM leaf devices with which the device shares at least one Ethernet segment), in that case only, the device forwards east-west multicast source traffic on the source VLAN to the multihoming peers instead of using the SBD. Then the receiving leaf devices forward or locally route the traffic to the destination VLAN.

See [Figure 112 on page 1114](#). We need to configure the SBD on all devices for OISM to work. We don't need to configure VLAN-1 and VLAN-2 on leaf devices that don't have receivers on those VLANs.

Routing east-west traffic mostly on the SBD supports the asymmetric bridge domains model—all leaf devices don't need to host all of the source VLANs in the network. You only need to configure the SBD in common on all of the OISM leaf devices. In the case of multihoming peers, however, you must configure the revenue VLANs symmetrically on the devices that are multihoming peers.

Figure 112: Enhanced OISM—Forward on Source VLAN Only to Multihoming Peers and Otherwise Route Only on SBD



In [Figure 112 on page 1114](#):

- Receivers send IGMP or MLD join messages to express interest in receiving multicast traffic for a multicast group (*,G) or multicast source and group (S,G) on a particular VLAN.
- Leaf-1 and Leaf-2 share an Ethernet segment for multihomed host Mcast-Src-1. As a result, we configure the same VLANs, VLAN-1 and VLAN-2, symmetrically on both of those devices, even though Leaf-1 might not have any receivers that use VLAN-2.
- Leaf-1 receives the multicast traffic on the source VLAN, VLAN-1, and:
 - Forwards the traffic to Leaf-2, its multihoming peer, on the source VLAN, VLAN-2

Leaf-2 then forwards the traffic to interested receivers on the source VLAN, VLAN-1, or locally routes the traffic to interested receivers on destination VLAN VLAN-2.
- Routes the traffic onto the SBD to the other OISM leaf devices that are not its multihoming peers and have interested receivers.

The OISM leaf devices receive the traffic on the SBD, and locally route the traffic to interested receivers on the destination VLAN, VLAN-1 or VLAN-2.

PIM Registration with Enhanced OISM for Internal Sources Based on EVPN Type 10 S-PMSI A-D Routes

Enhanced OISM requires some differences in handling PIM source registration for north-south traffic from internal sources to receivers outside of the EVPN network.

With regular OISM, the border leaf devices running as OISM PEG devices receive traffic from external multicast sources only on the supplemental bridge domain (SBD). The PEG devices receive traffic from internal multicast sources on the source VLAN. OISM PEG devices should only perform PIM registration for internal sources, so with the regular OISM design, the PEG devices can easily distinguish the internal sources, and do PIM source registration only for those sources.

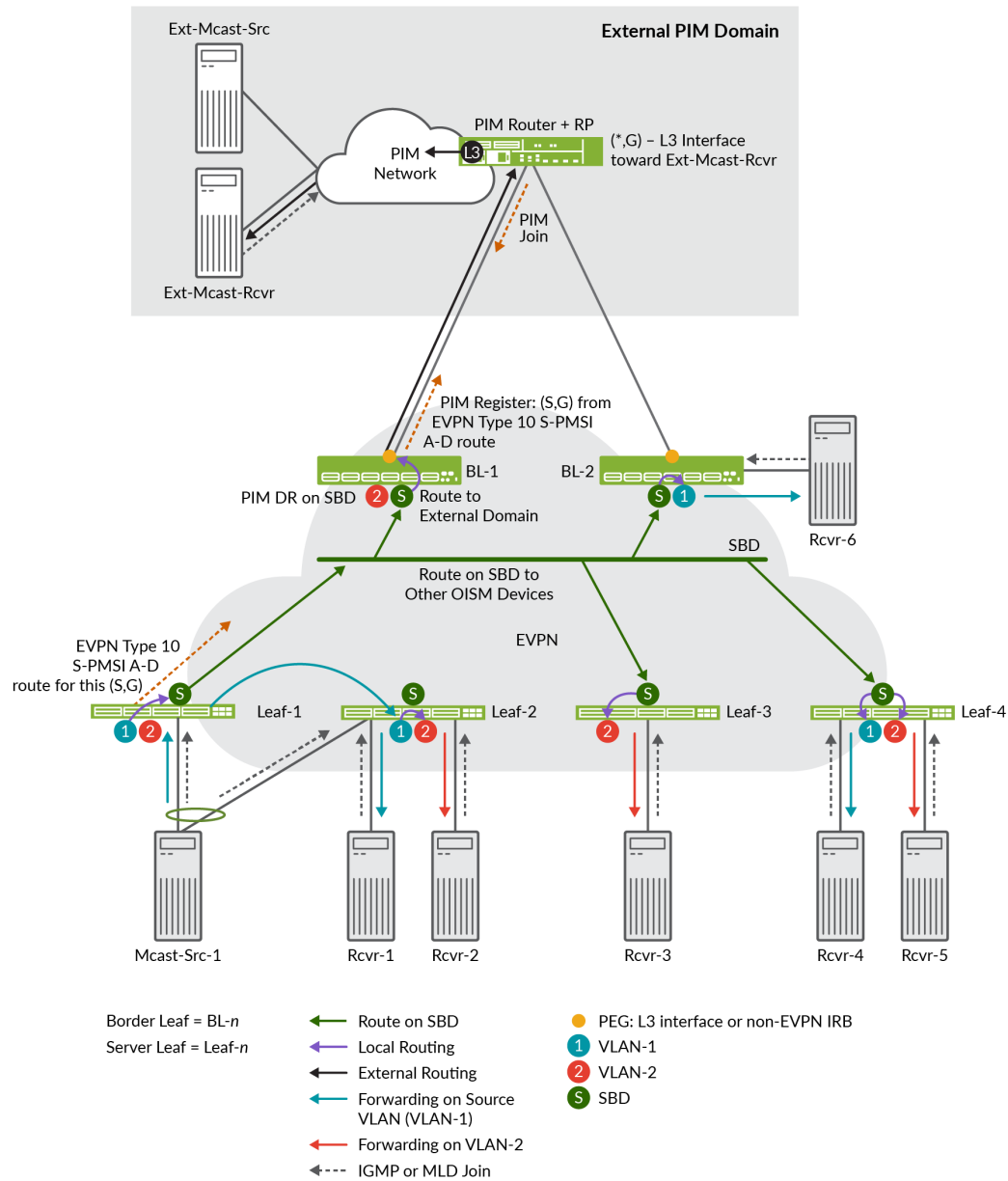
With enhanced OISM, the PEG devices receive traffic on the SBD from both external and internal multicast sources. Because the PEG devices should only perform PIM registration to the PIM RP for internal sources, the PEG devices running enhanced OISM must be able to distinguish between internal and external multicast sources.

The enhanced OISM design employs EVPN Type 10 Selective P-router Multicast Service Interface (S-PMSI) Auto-Discovery (A-D) routes to make this distinction, as follows (and see [Figure 113 on page 1116](#)):

- The ingress OISM leaf devices that receive traffic from internal multicast sources advertise S-PMSI A-D routes for those multicast (S,G) sources and groups.
- If a PEG device receives traffic on an SBD IRB interface and doesn't see an S-PMSI A-D route for that source, the device interprets that source as an external source.
- The PEG device only sends a PIM register to the PIM RP for the sources that correspond to received S-PMSI A-D routes.

This design ensures that the PEG devices perform PIM source registration only for the multicast sources inside the EVPN network.

Figure 113: Enhanced OISM—Internal Source PIM Registration Using EVPN Type 10 S-PMSI A-D Routes



For example, [Figure 113 on page 1116](#) shows the same enhanced OISM internal traffic flow as [Figure 112 on page 1114](#) with the addition of the external PIM domain serving external receivers. In the figure:

1. When Leaf-1 receives multicast traffic from Mcast-Src-1, Leaf-1 generates an S-PMSI A-D route and sends it into the EVPN network.

2. PEG device BL-1 receives the S-PMSI A-D route. BL-2 also receives the S-PMSI A-D route. However, BL-1 is the PIM DR on the SBD, so BL-1 sends the PIM Register message on its external multicast interface toward the PIM RP for that (S,G).
3. The PIM RP sends a PIM Join message back toward BL-1. BL-1 receives the PIM join and creates an (S,G) multicast routing table entry for the external receiver.
4. When BL-1 receives multicast traffic for that (S,G) on the SBD, it locally routes the traffic to the external multicast interface toward the external receivers.

You can use commands such as the following to see details on the EVPN Type 10 S-PMSI A-D routes on the OISM leaf devices:

- `show evpn oism spmsi-ad extensive`
- `show route table evpn-instance-name.evpn-mcsn.1 match 10* extensive`

Considerations for OISM Configurations

IN THIS SECTION

- [IGMPv2 and IGMPv3 \(or MLDv1 and MLDv2\) in the Same EVPN-VXLAN Fabric | 1117](#)
- [Latency and Scaling Trade-Offs for Installing Multicast Routes with OISM \(install-star-g-routes Option\) | 1122](#)
- [OISM and AR Scaling with Many VLANs | 1124](#)
- [PEG DF Election | 1129](#)
- [Statically Identify Multihoming Peers With Enhanced OISM To Improve Convergence | 1139](#)
- [Enhanced OISM with an EVPN-VXLAN IPv6 Underlay Configuration | 1140](#)
- [Enhanced OISM Exception Policy to Forward on Source VLAN Instead of SBD for Packets with TTL=1 | 1141](#)

Before you begin to set up your OISM installation, here are a few considerations in specific use cases. These considerations apply to both regular OISM and enhanced OISM modes unless the section specifies otherwise.

IGMPv2 and IGMPv3 (or MLDv1 and MLDv2) in the Same EVPN-VXLAN Fabric

You have several options to configure IGMP snooping with IGMPv2, IGMPv3, or both IGMP versions together in an EVPN-VXLAN fabric with OISM. The same is true of MLD snooping with MLDv1, MLDv2,

or both MLD versions together. You might also want to mix configuring IGMP and MLD together in the same fabric. This section includes configuration considerations for a few of these options.

If you have traffic for either IGMPv2 or IGMPv3 on a device with OISM enabled, you can enable that IGMP version globally on the device with IGMP snooping. You can alternatively enable that IGMP version only for the interfaces that will handle the multicast traffic. You can enable IGMP snooping with that version of IGMP on all VLANs or on specific VLANs, as needed.

You have the same options for either MDLv1 or MLDv2 with MLD snooping (on the platforms that support MLD with OISM).

You can also enable one version of IGMP with IGMP snooping and one version of MLD with MLD snooping together on the device with OISM.

However, OISM supports IGMP snooping with both IGMPv2 and IGMPv3 traffic together on a device only within the following constraints:

- You can't enable IGMP snooping with both IGMPv2 and IGMPv3 for interfaces in the same VLAN.
- You can't enable IGMP snooping with both IGMPv2 and IGMPv3 for VLANs that are part of the same L3 VRF instance with OISM enabled.

The constraints above also apply if you want to enable MLD snooping with both MLDv1 and MLDv2 traffic together on a device.

These constraints don't apply if you use one version of IGMP with one version of MLD together on a device.

To support IGMP snooping with both IGMP versions, or MLD snooping with both MLD versions, you must configure:

- One tenant VRF instance to support the IGMPv2 or MLDv1 receivers.
- Another tenant VRF instance to support the IGMPv3 or MLDv2 receivers.

as follows:

1. In your configuration, define VLANs for the IGMPv2 receivers, and define different VLANs for the IGMPv3 receivers.

Similarly, for MLD, define VLANs for the MLDv1 receivers, and define different VLANs for the MLDv2 receivers.

2. Include the IRB interfaces that support IGMPv2 in one VRF instance, and enable IGMPv2 on those IRB interfaces. Enable IGMP snooping on the corresponding VLANs.

Similarly, for MLD, include the IRB interfaces that support MLDv1 in one VRF instance, and enable MLDv1 on those IRB interfaces. Enable MLD snooping on the corresponding VLANs.

3. Include the IRB interfaces that support IGMPv3 in another VRF instance, and enable IGMPv3 on those IRB interfaces. Enable IGMP snooping with the `evpn-ssm-reports-only` option on the corresponding VLANs.

Similarly, for MLD, include the IRB interfaces that support MLDv2 in another VRF instance, and enable MLDv2 on those IRB interfaces. Enable MLD snooping with the `evpn-ssm-reports-only` option on the corresponding VLANs.

In this use case, for each IGMP or MLD version, allocate a set of VLANs and IRB interfaces for:

- The OISM revenue bridge domains.
- The SBD.
- Any external multicast VLANs and interfaces (depending on the external multicast method you use).

You also define two L3 VRF instances for each tenant instance you need in your installation, one for each IGMP version or MLD version. If you use MAC-VRF routing instances at L2, you might want to allocate different MAC-VRF EVPN instances for the IGMP snooping or MLD snooping traffic for each IGMP or MLD version.

The next sections show example configurations with both versions of IGMP or both versions of MLD together. You can scale these simple scenarios to support different tenants with different combinations of IGMP or MLD versions.

See ["Supported IGMP or MLD Versions and Group Membership Report Modes" on page 936](#) for more information on IGMP any-source multicast (ASM) mode and source-specific multicast (SSM) mode support with IGMPv2, IGMPv3, MLDv1, and MLDv2 in EVPN-VXLAN fabrics.

Example Configuration with IGMPv2 and IGMPv3 Together

Consider a use case with both IGMP versions in a fabric you set up with the M-VLAN IRB method for external multicast. You want to support IGMP snooping with both IGMPv2 and IGMPv3 traffic. In that case, you might configure the following MAC-VRF instances, L3 VRF instances, VLANs, and corresponding IRB interfaces:

- MAC-VRF2 and L3VRF-A to support IGMPv2 receivers:
 - Revenue bridge domain VLAN-100 with `irb.100`
 - SBD VLAN-302 with `irb.302`
 - (Border leaf devices only) M-VLAN VLAN-902 with `irb.902`
- MAC-VRF3 and L3VRF-B to support IGMPv3 receivers:
 - Revenue bridge domain VLAN-200 with `irb.200`

- SBD VLAN-303 with irb.303
- (Border leaf devices only) M-VLAN VLAN-903 with irb.903

Then include the IGMPv2 IRB interfaces in L3VRF-A and enable IGMPv2 for those IRB interfaces. Include the IGMPv3 IRB interfaces in L3VRF-B and enable IGMPv3 for those IRB interfaces.

For example:

```
set routing-instances L3VRF-A interface irb.100      # revenue bridge domain for IGMPv2 receivers
set routing-instances L3VRF-A interface irb.302      # SBD for IGMPv2 receivers
set routing-instances L3VRF-A interface irb.902      # M-VLAN for IGMPv2 (border leaf only)

set routing-instances L3VRF-B interface irb.200      # revenue bridge domain for IGMPv3 receivers
set routing-instances L3VRF-B interface irb.303      # SBD for IGMPv3 receivers
set routing-instances L3VRF-B interface irb.903      # M-VLAN for IGMPv3 (border leaf only)

# version 2 option isn't required for IGMPv2 because that's the default IGMP version
set protocols igmp interface irb.100 <version 2>    # revenue bridge domain for IGMPv2
receivers
set protocols igmp interface irb.302 <version 2>    # SBD for IGMPv2 receivers
set protocols igmp interface irb.902 <version 2>    # M-VLAN for IGMPv2 (border leaf only)

# version 3 option is required to enable IGMPv3
set protocols igmp interface irb.200 version 3      # revenue bridge domain for IGMPv3
receivers
set protocols igmp interface irb.303 version 3      # SBD for IGMPv3 receivers
set protocols igmp interface irb.903 version 3      # M-VLAN for IGMPv3 (border leaf only)
```

Finally, enable IGMP snooping at L2 in the EVPN instances as follows:

- Configure `igmp-snooping` in MAC-VRF2 for the VLANs corresponding to the IGMPv2 IRB interfaces.
- Configure `igmp-snooping` in MAC-VRF3 for the VLANs corresponding to the IGMPv3 IRB interfaces.

Include the `evpn-ssm-reports-only` option only when you enable IGMP snooping for IGMPv3 traffic.

For example:

```
set routing-instances MAC-VRF2 protocols igmp-snooping vlan VLAN-100 # IGMPv2-enabled VLAN
set routing-instances MAC-VRF2 protocols igmp-snooping vlan VLAN-302 # IGMPv2-enabled VLAN
set routing-instances MAC-VRF2 protocols igmp-snooping vlan VLAN-902 # IGMPv2-enabled VLAN

set routing-instances MAC-VRF3 protocols igmp-snooping vlan VLAN-200 evpn-ssm-reports-only #
```

IGMPv3-enabled VLAN

```
set routing-instances MAC-VRF3 protocols igmp-snooping vlan VLAN-303 evpn-ssm-reports-only #
IGMPv3-enabled VLAN
```

```
set routing-instances MAC-VRF3 protocols igmp-snooping vlan VLAN-903 evpn-ssm-reports-only #
IGMPv3-enabled VLAN
```



NOTE: With the non-EVPN IRB method for external multicast, you don't include the `evpn-ssm-reports-only` option on the non-EVPN IRB interface. You don't need this option because with the non-EVPN IRB method, you don't extend the external multicast interface in the EVPN instance.

When you use the L3 interface method for external multicast, you don't enable IGMP snooping at all on the L3 interface to the external PIM domain. That interface operates at L3, while IGMP snooping operates at L2.

Example Configuration with MLDv1 and MLDv2 Together

Consider this use case with both MLDv1 and MLDv2 with MLD snooping in a fabric with OISM:

- MAC-VRF1 and L3VRF-A to support MLDv1 receivers:
 - Revenue bridge domain VLAN-100 with `irb.100`
 - SBD VLAN-301 with `irb.301`
- MAC-VRF2 and L3VRF-B to support MLDv2 receivers:
 - Revenue bridge domain VLAN-200 with `irb.200`
 - SBD VLAN-302 with `irb.302`



NOTE: In this use case, we don't use the M-VLAN IRB method for external multicast, so we don't configure an M-VLAN IRB interface like we do in the IGMP use case above.

In this case, you configure:

- MLDv1 on the IRB interfaces for MLDv1 receivers (VLANs 100 and 301).
- MLDv2 on the IRB interfaces for MLDv2 receivers (VLANs 200 and 302)
- MLD snooping for MLDv1 VLANs in MAC-VRF1.
- MLD snooping for MLDv2 VLANs with the `evpn-ssm-reports-only` option in MAC-VRF2.

For example:

```

set routing-instances L3VRF-A interface irb.100    # revenue bridge domain for MLDv1 receivers
set routing-instances L3VRF-A interface irb.301    # SBD for MLDv1 receivers

set routing-instances L3VRF-B interface irb.200    # revenue bridge domain for MLDv2 receivers
set routing-instances L3VRF-B interface irb.302    # SBD for MLDv2 receivers

# version 1 option isn't required for MLDv1 because that's the default MLD version
set protocols mld interface irb.100                # revenue bridge domain for MLDv1 receivers
set protocols mld interface irb.301                # SBD for MLDv1 receivers

set protocols mld interface irb.200 version 2      # revenue bridge domain for MLDv2 receivers
set protocols mld interface irb.302 version 2      # SBD for MLDv2 receivers

set routing-instances MAC-VRF1 protocols mld-snooping vlan VLAN-100 # MLDv1-enabled VLAN
set routing-instances MAC-VRF1 protocols mld-snooping vlan VLAN-301 # MLDv1-enabled VLAN

set routing-instances MAC-VRF2 protocols mld-snooping vlan VLAN-200 evpn-ssm-reports-only #
MLDv2-enabled VLAN
set routing-instances MAC-VRF2 protocols mld-snooping vlan VLAN-302 evpn-ssm-reports-only #
MLDv2-enabled VLAN

```

Latency and Scaling Trade-Offs for Installing Multicast Routes with OISM (install-star-g-routes Option)

Devices in an OISM-enabled fabric send EVPN Type 6 routes so other EVPN devices learn about receivers that are interested in the traffic for a multicast group. The receivers are in different OISM revenue bridge domains in an L3 VRF instance. To save bandwidth in the EVPN fabric core, OISM devices send and receive the Type 6 routes only on the OISM SBD in the routing instance.

To help minimize packet loss at the onset of a multicast flow, we provide the `install-star-g-routes` option at the `[edit <routing-instances name> multicast-snooping-options oism]` hierarchy level (see *oism (Multicast Snooping Options)*). When you configure this option, upon receiving a Type 6 route, the RE on the device immediately installs corresponding (*,G) multicast routes on the PFE for all of the revenue bridge domain VLANs in the routing instance.

With this option, you trade off taking up extra PFE resources to improve network latency. Lower-scale deployments might have fewer multicast flows but have strict network latency requirements. To improve network latency in that case, the device installs the (*,G) routes in the data plane in advance of any incoming multicast traffic.

Configure this option:

- Globally if you configure EVPN in the default-switch instance, at the [edit multicast-snooping-options oism] hierarchy level.
- In the MAC-VRF instances if you configure EVPN in instances of type mac-vrf, at the [edit routing-instances *instance-name* multicast-snooping-options oism] .

We require that you configure `install-star-g-routes` with OISM on the QFX10000 line of switches, QFX5130-32CD switches, and QFX5700 switches when you configure those devices with AR in the AR replicator role.

In releases prior to Junos OS and Junos OS Evolved Release 23.4R1, you must also configure the `install-star-g-routes` option on the following devices when you configure them as OISM server leaf or border leaf devices:

- Switches in the QFX10000 line.
- PTX10001-36MR, PTX10004, PTX10008, and PTX10016 routers.

Starting in Junos OS and Junos OS Evolved Release 23.4R1, we no longer require you to set this option when you configure those devices as OISM server leaf or border leaf devices.

We don't recommend setting this option other than in the use cases mentioned above.

Consider this option only if you have very stringent latency requirements and can trade off higher scaling to achieve better network latency.



NOTE: The functions of the `install-star-g-routes` option and the `conserve-mcast-routes-in-pfe` option are mutually exclusive, so you can use only one or the other of these options in a routing instance. See ["ACX Series Routers, QFX5130-32CD Switches, and QFX5700 Switches as Server Leaf and Border Leaf Devices with OISM" on page 1125](#) for more on when to use the `conserve-mcast-routes-in-pfe` option.

Default Behavior without the `install-star-g-routes` Option

By default, without this option, the device prioritizes saving resources on the PFE by not installing multicast routes until the multicast traffic arrives. In this default case:

1. The PFE receives multicast traffic from source S for multicast group G.
2. The PFE doesn't have forwarding next-hop information for the traffic, so it signals the RE to get that information.



NOTE: The PFE drops multicast traffic until it gets the routing information.

3. The RE learns about the multicast flow for (S,G) from the PFE, and installs that route on the PFE.
4. The PFE sends the traffic on the next hop in the installed (S,G) route.

Behavior with the install-star-g-routes Option

With the `install-star-g-routes` option, the device prioritizes having multicast routing information available on the PFE before any traffic arrives. The device consumes extra PFE resources for routes that it isn't using yet (and might never be used). With this option:

1. The RE receives an EVPN Type 6 route for a receiver subscribing to traffic for a multicast group G on the OISM SBD in a routing instance.
2. The RE installs corresponding (*,G) routes on the PFE for all of the revenue bridge domains in the L3 VRF instance.
3. At some later time, the PFE receives multicast traffic from source S for multicast group G.
4. The PFE has forwarding next hop information for traffic for (*,G). So it forwards the traffic to receivers on any revenue bridge domains using the (*,G) route next hop.
5. The PFE also signals the RE that it has received multicast traffic from source S for multicast group G.
6. The RE learns about the multicast flow for (S,G) from the PFE. The RE installs the (S,G) route on the PFE.
7. The PFE continues sending the traffic, but now uses the (S,G) route and the next hop in that more specific route.



NOTE: The PFE still retains the (*,G) routes per revenue bridge domain that the RE installed after receiving the Type 6 route.

OISM and AR Scaling with Many VLANs

With OISM and IGMP snooping or MLD snooping enabled in an EVPN-VXLAN fabric, OISM server leaf and border leaf devices send EVPN Type 6 SMET routes into the EVPN core when their receivers join a multicast group.

When an OISM-enabled device receives Type 6 routes on the SBD, the device:

- Derives multicast states from the Type 6 routes as follows:
 - (*,G) states for IGMPv2 or MLDv1
 - (S,G) states for IGMPv3 or MLDv2

- Installs the derived states on the OISM SBD and revenue bridge domain VLANs in the MAC-VRF instance for *a//*VLANs that are part of OISM-enabled L3 tenant VRF instances.
- Uses the derived multicast routes to optimize multicast forwarding by selectively sending the traffic for a group only to other EVPN devices that have receivers subscribed to that group.

On some devices that support OISM, you can also configure the assisted replication (AR) multicast optimization feature with OISM enabled. AR replicator devices use the Type 6 routes the same way OISM devices do.

QFX5130-32CD and QFX5700 switches can serve as OISM server leaf or border leaf devices. They can act as AR replicators only on devices that are not also OISM server leaf or border leaf devices. In that case, the device operates in the standalone AR replicator role.

The next sections describe configuration considerations on these devices when you configure them as OISM server leaf or border leaf devices, or as standalone AR replicators with OISM.



NOTE: The use cases and sample configurations in the next sections show IGMP configurations for IPv4 multicast, but also apply in the same ways to MLD configurations for IPv6 multicast.

ACX Series Routers, QFX5130-32CD Switches, and QFX5700 Switches as Server Leaf and Border Leaf Devices with OISM

When you configure ACX Series routers, QFX5130-32CD switches, and QFX5700 switches as server leaf or border leaf devices with OISM, as soon as these devices receive multicast traffic, they use the L3 multicast routes from PIM to forward the traffic. They use the derived multicast snooping states only to learn which receivers are interested in a multicast stream. They don't need to save the multicast snooping derived states in the forwarding plane for forwarding traffic.

Starting in Junos OS Evolved Releases 22.4R2 and 23.1R1, when you configure these devices as OISM server leaf and border leaf devices, we require you to also configure the `conserve-mcast-routes-in-pfe` option at the `[edit routing-instances name multicast-snooping-options oism]` hierarchy level. (See *oism (Multicast Snooping Options)*.) With this option, these devices conserve PFE table space by installing only the L3 multicast routes; they avoid installing L2 multicast snooping routes.

Use the following guidelines for setting the `conserve-mcast-routes-in-pfe` option:

- You must set this option on ACX Series routers, QFX5130-32CD switches, and QFX5700 switches when you configure them as server leaf or border leaf devices with OISM enabled.
- Set this option in all OISM-enabled MAC-VRF EVPN routing instances on the device.
- Don't configure this option if you did not enable OISM on the device.

- When you disable OISM on a device, you must also delete this setting.



NOTE: The functions of the `conserve-mcast-routes-in-pfe` option and the `install-star-g-routes` option are mutually exclusive, so you can use only one or the other of these options in a routing instance. See ["Latency and Scaling Trade-Offs for Installing Multicast Routes with OISM \(install-star-g-routes Option\)"](#) on page 1122 for more on when to use the `install-star-g-routes` option.

QFX5130-32CD and QFX5700 Switches as Standalone AR Replicators with OISM

QFX5130-32CD and QFX5700 switches can serve as standalone AR replicators in a fabric with OISM. However, in fabrics with many VLANs, QFX5130-32CD and QFX5700 switches might have scaling issues when installing the multicast states on all the OISM VLANs.

As a result, starting in Junos OS Evolved 22.2R1, when you configure these switches as standalone AR replicators with OISM enabled, by default these switches only install multicast states on the SBD VLAN. (This includes multicast (*,G) states for IGMPv2 and multicast (S,G) states for IGMPv3.) These switches don't install the multicast states on all the revenue bridge domain VLANs.

For example, consider a QFX5130-32CD device where you have a MAC-VRF instance `evpn-vxlan-A` with 3 VLANs—`VLAN_2`, `VLAN_3`, and `VLAN_4`. The `show igmp snooping evpn status detail` command shows that you configured `VLAN_4` as the SBD (the Supplementary BD output field is Yes), and the other two VLANs are OISM revenue bridge domain VLANs:

```
user@device> show igmp snooping evpn status detail
Instance: evpn-vxlan-A
  Bridge-Domain: VLAN_2, VN Identifier: 2
    OISM          : Enabled
    Supplementary BD: No
    External VLAN  : No
  Bridge-Domain: VLAN_3, VN Identifier: 3
    OISM          : Enabled
    Supplementary BD: No
    External VLAN  : No
  Bridge-Domain: VLAN_4, VN Identifier: 4
    OISM          : Enabled
    Supplementary BD: Yes
    External VLAN  : No
```

The device received Type 6 routes from remote devices for multicast groups 233.252.0.1 and 233.252.0.2:

```

user@device> show route table bgp.evpn.0 match-prefix 6*233.252.0.1*

bgp.evpn.0: 269 destinations, 269 routes (269 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.0.1:9::4::233.252.0.1::192.168.0.1/520
    *[BGP/170] 00:10:28, localpref 100, from 192.168.0.1
    AS path: I, validation-state: unverified
    > to 10.1.1.2 via et-0/0/3:0.0
6: 192.168.0.2:9::4::233.252.0.1:: 192.168.0.2/520
    *[BGP/170] 00:09:30, localpref 100, from 192.168.0.2
    AS path: I, validation-state: unverified
    > to 10.1.2.2 via et-0/0/3:2.0
6: 192.168.0.3:9::4::233.252.0.1::192.168.0.3/520
    *[BGP/170] 00:12:14, localpref 100, from 192.168.0.3
    AS path: I, validation-state: unverified
    > to 10.1.3.2 via ae1.0

user@device> show route table bgp.evpn.0 match-prefix 6*233.252.0.2*

bgp.evpn.0: 269 destinations, 269 routes (269 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.0.1:9::4::233.252.0.2::192.168.0.1/520
    *[BGP/170] 00:10:34, localpref 100, from 192.168.0.1
    AS path: I, validation-state: unverified
    > to 10.1.1.2 via et-0/0/3:0.0
6: 192.168.0.2:9::4::233.252.0.2:: 192.168.0.2/520
    *[BGP/170] 00:09:36, localpref 100, from 192.168.0.2
    AS path: I, validation-state: unverified
    > to 10.1.2.2 via et-0/0/3:2.0
6: 192.168.0.3:9::4::233.252.0.2:: 192.168.0.3/520
    *[BGP/170] 00:12:20, localpref 100, from 192.168.0.3
    AS path: I, validation-state: unverified
    > to 10.1.3.2 via ae1.0

```

Due to the scaling behavior difference on QFX5130-32CD or QFX5700 switches, if you run the `show multicast snooping route` command on these devices, the output shows multicast group entries only on the

SBD, and not on any of the revenue bridge domains. For example, with our multicast groups 233.252.0.1 and 233.252.0.2:

```
user@device> show multicast snooping route extensive instance evpn-vxlan-A
```

```
Nexthop Bulking: OFF
```

```
Family: INET
```

```
Group: 224.0.0.0/4
```

```
Source: *
```

```
Vlan: VLAN_2
```

```
Mesh-group: __ar_flood__
```

```
Downstream interface list:
```

```
evpn-core-nh -(639026)
```

```
Statistics: 0 kBps, 0 pps, 0 packets
```

```
Next-hop ID: 639104
```

```
Route state: Active
```

```
Forwarding state: Forwarding
```

```
Sensor ID: 4026531845
```

```
Group: 224.0.0.0/4
```

```
Source: *
```

```
Vlan: VLAN_3
```

```
Mesh-group: __ar_flood__
```

```
Downstream interface list:
```

```
evpn-core-nh -(639022)
```

```
Statistics: 0 kBps, 0 pps, 0 packets
```

```
Next-hop ID: 639106
```

```
Route state: Active
```

```
Forwarding state: Forwarding
```

```
Sensor ID: 4026531846
```

```
Group: 224.0.0.0/4
```

```
Source: *
```

```
Vlan: VLAN_4
```

```
Mesh-group: __ar_flood__
```

```
Downstream interface list:
```

```
evpn-core-nh -(639018)
```

```
Statistics: 0 kBps, 0 pps, 0 packets
```

```
Next-hop ID: 639097
```

```
Route state: Active
```

```
Forwarding state: Forwarding
```

Sensor ID: 4026531844

Group: 233.252.0.1/32

Source: *

Vlan: VLAN_4

Mesh-group: __ar_flood__

Downstream interface list:

evpn-core-nh -(165537)

Statistics: 523 kbps, 500 pps, 290630 packets

Next-hop ID: 220045

Route state: Active

Forwarding state: Forwarding

Sensor ID: 4026531843

Group: 233.252.0.2/32

Source: *

Vlan: VLAN_4

Mesh-group: __ar_flood__

Downstream interface list:

evpn-core-nh -(165537)

Statistics: 0 kbps, 0 pps, 238 packets

Next-hop ID: 220045

Route state: Active

Forwarding state: Forwarding

Sensor ID: 4026531842

QFX5130-32CD or QFX5700 switches that are not running as AR replicators with OISM will install the multicast group entries on the revenue bridge domain VLANs and the SBD. When you run the `show multicast snooping route` command in that case, you see the revenue bridge domain VLANs and the SBD. This behavior also applies to all other platforms running OISM whether the device is an AR replicator or not.



NOTE: On QFX5130-32CD or QFX5700 switches acting as AR replicators, you should not configure the `conserve-mcast-routes-in-pfe` option we describe in ["ACX Series Routers, QFX5130-32CD Switches, and QFX5700 Switches as Server Leaf and Border Leaf Devices with OISM"](#) on page 1125.

PEG DF Election

By default, peer OISM PEG devices use PIM-based DF election—the devices discover their PIM neighbors on the OISM revenue VLANs and the SBD in each L3 VRF, and elect the DF from among those neighbors. Starting in Junos OS Release 23.4R1 and Junos OS Evolved Release 23.4R1, you can

instead configure mod-based or preference-based PEG DF election using the *peg-df-election* statement at the [edit routing-instances *name* protocols evpn oism pim-evpn-gateway] hierarchy level, as follows:

```
peg-df-election {
  delay-time  num;
  mod;          # default method if you don't configure either the mod or preference option
  preference {
    use-least;
    value  preference-value;
  };
}
```

When you configure PEG DF election, the device maintains an ordinal list of the PEG devices in the fabric that host each revenue VLAN (bridge domain) or the SBD. PEG devices use the EVPN multicast flags extended community in EVPN Type 3 IMET routes to advertise OISM, IGMP snooping, MLD snooping, and PEG device support. In addition, PEG devices include the DF election extended community to communicate the configured DF election method parameters (as defined in the [RFC 8584](#) standard).

The peer PEG DF candidates are the devices that advertise IMET routes for a revenue VLAN or the SBD with the following EVPN multicast flags extended community values:

- For IPv4 multicast—igmp-snooping-enabled:oism:peg
- For IPv6 multicast—mld-snooping-enabled:oism:peg



NOTE: The devices elect a DF for IPv4 multicast traffic and a DF for IPv6 multicast traffic separately based on those advertised multicast flags extended community values.

When the PEG devices use PEG DF election, they don't use the PIM protocol (they don't exchange PIM protocol packets) within the data center. As a result, we recommend you have external L3 redundancy on the PEG devices.

You can configure the PEG devices to use either of the following PEG DF election methods:

- **Mod-based**—The default method when you enable PEG DF election at the [edit routing-instances *name* protocols evpn oism pim-evpn-gateway peg-df-election] hierarchy level without including the *mod* option or the preference option. You can also explicitly configure the *mod* option to use this method.

The algorithm to choose the device in the ordinal list that will be the DF is:

(mapped VNI for the VLAN) mod (number of entries in the list)

For example, if you have three peer PEG devices BL1, BL2, and BL3 configured with mod-based PEG DF election for VLAN 1, which is mapped to VNI 100:

- The three devices each maintain an ordinal list of PEG DF candidates for VLAN 1 and VNI 100, such as:

Table 63: Sample Mod-based PEG DF Election Candidates List

Index	Device
0	BL1
1	BL2
2	BL3

- In this case, (*mapped VNI for the VLAN*) mod (*number of entries in the list*) is $(100) \bmod (3) = 1$, so the devices elect the device at index 1 as the PEG DF, which is BL2.
- **Preference-based with a customized preference value**—Configure the value *preference-value* option at the [edit routing-instances *name* protocols evpn oism pim-evpn-gateway peg-df-election preference] hierarchy level. We recommend you set a unique preference value on each peer PEG device. With preference-based PEG DF election:
 - Each device communicates its preference value in the EVPN Type 3 IMET route advertisement.
 - The peer PEG devices elect the device with the highest preference value (by default) as the DF for the VLAN.
 - You can customize the preference-based method to elect the device with the *lowest* preference value instead of the highest value. To do this, set the use-least option at the [edit routing-instances *name* protocols evpn oism pim-evpn-gateway peg-df-election preference] hierarchy level.

With either PEG DF election method, you can also:

- Specify a time period to wait before the devices elect a DF. To do this, set the delay-time *num* option at the [edit routing-instances *name* protocols evpn oism pim-evpn-gateway peg-df-election] hierarchy level.
- Set the maximum number (0-255) of PEG DF election event entries the device maintains in a database for DF election history per VLAN (VNI). To do this, set the peg-df-election-history *num* option at the [edit <routing-instances *name*> protocols evpn] hierarchy level.

The show evpn oism peg-df-status extensive command displays DF election history details.

All peer PEG devices must use the same DF election mechanism. As a result, if you enable PEG DF election, configure the same DF election method symmetrically on all peer PEG devices. If the configured PEG DF election methods don't match, the peer PEG devices all fall back to using the default mod-based PEG DF election method. If you enable PEG DF election on only some of the peer PEG devices, all of the devices fall back to using PIM-based DF election.

How to Check PEG DF Election Status

Use the following commands to see PEG DF election status for a PEG device:

1. `show evpn oism`—See if PEG DF election is enabled for each L3 routing instance. Include the extensive option to see the configured PEG DF election method and preference value if you selected the preference option. For example:

Mod-based PEG DF Election:

```
user@BL1> show evpn oism
L3 context      SBD      PEG-DF-ELECTION
VRF-1           irb.4    Enabled

user@BL1> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Enabled
  PEG DF Algorithm: MOD
  OISM Mode: Regular OISM
```

Preference-based PEG DF Election—uses the highest preference value by default, so BL1 would be the elected PEG DF here:

```
user@BL1> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Enabled
  PEG DF Algorithm: PREFERENCE(200), use-least-preference: F
  OISM Mode: Regular OISM
```

```
user@BL2> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Enabled
```

```
PEG DF Algorithm: PREFERENCE(100), use-least-preference: F
OISM Mode: Regular OISM
```

Preference-based PEG DF Election with the use-least option—uses the lowest preference value, so BL2 would be the elected PEG DF here:

```
user@BL1> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Enabled
  PEG DF Algorithm: PREFERENCE(200), use-least-preference: T
  OISM Mode: Regular OISM
```

```
user@BL2> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Enabled
  PEG DF Algorithm: PREFERENCE(100), use-least-preference: T
  OISM Mode: Regular OISM
```

2. `show evpn oism peg-df-status`—See PEG DF election status and the DF IP address for all or selected L3 VRF instances per VLAN-mapped VNI. This command displays information only for routing instances and VNIs where you configured PEG DF election. Use the `extensive` option to see more details such as the DF candidates list and the DF election history information. For example:

Mod-based PEG DF Election:

```
user@BL1> show evpn oism peg-df-status
EVPN L3 context: VRF-1

  VN Identifier: 100
  IPv4 Multicast DF Status: DF, DF Address: 192.168.1.1

  VN Identifier: 200
  IPv4 Multicast DF Status: NDF, DF Address: 192.168.2.1

  VN Identifier: 300
  IPv4 Multicast DF Status: DF, DF Address: 192.168.1.1
.
.
```

```

.
user@BL1> show evpn oism peg-df-status extensive
EVPN L3 context: VRF-1

VN Identifier: 100
  IPv4 Multicast DF Status: DF, DF Address: 192.168.1.1
    DF Candidates List[Total: 2]
      0: 192.168.1.1
      1: 192.168.2.1
    PIM Update TS: Oct 14 11:25:17, DF Compute TS: Oct 14 11:25:17
History db:
  Time                               Event
  Oct 14 11:25:17.358 2023  Added DF candidate 192.168.1.1 df type: 0, pref 32767 dp: 0
gran: 0 ord_num 0 total 1
  Oct 14 11:25:17.358 2023  Added DF candidate 192.168.2.1 df type: 0, pref 0 dp: 0 gran: 0
ord_num 1 total 2
  Oct 14 11:25:17.359 2023  Completed DF election, 2 total candidate(s), max_candidate_count
2, mod 0, l2_domain_ID 2, new DF PE address 192.168.1.1
.
.
.

```

Preference-based PEG DF Election—BL1 has the highest preference value, 200, and is the PEG DF here for VNIs 100 and 200; BL2 with preference value 100 is not elected as the DF (nDF):

```

user@BL1> show evpn oism peg-df-status
EVPN L3 context: VRF-1

VN Identifier: 100
  IPv4 Multicast DF Status: DF, DF Address: 192.168.1.1

VN Identifier: 200
  IPv4 Multicast DF Status: DF, DF Address: 192.168.1.1
.
.
.

user@BL1> show evpn oism peg-df-status extensive
EVPN L3 context: VRF-1

VN Identifier: 100
  IPv4 Multicast DF Status: DF, DF Address: 192.168.1.1

```

```

    DF Candidates List[Total: 2]
      0: 192.168.1.1
      1: 192.168.2.1
    PIM Update TS: Oct 14 09:04:38, DF Compute TS: Oct 14 09:05:02
History db:
  Time                Event
  Oct 14 09:04:38.410 2023 Added DF candidate 192.168.1.1 df type: 2, pref 200 dp: 0
gran: 0 ord_num 0 total 1
  Oct 14 09:05:02.518 2023 Added DF candidate 192.168.2.1 df type: 2, pref 100 dp: 0
gran: 0 ord_num 1 total 2
  Oct 14 09:05:02.518 2023 Completed DF election, 2 total candidate(s),
max_candidate_count 2, mod 0, l2_domain_ID 0, new DF PE address 192.168.1.1

VN Identifier: 200
  IPv4 Multicast DF Status: DF, DF Address: 192.168.1.1
  DF Candidates List[Total: 2]
    0: 192.168.1.1
    1: 192.168.2.1
  PIM Update TS: Oct 14 09:04:38, DF Compute TS: Oct 14 09:05:02
History db:
  Time                Event
  Oct 14 09:04:38.412 2023 Added DF candidate 192.168.1.1 df type: 2, pref 200 dp: 0
gran: 0 ord_num 0 total 1
  Oct 14 09:05:02.518 2023 Added DF candidate 192.168.2.1 df type: 2, pref 100 dp: 0
gran: 0 ord_num 1 total 2
  Oct 14 09:05:02.518 2023 Completed DF election, 2 total candidate(s),
max_candidate_count 2, mod 0, l2_domain_ID 0, new DF PE address 192.168.1.1
.
.
.

```

```

user@BL2> show evpn oism peg-df-status

```

```

EVPN L3 context: VRF-1

```

```

VN Identifier: 200

```

```

IPv4 Multicast DF Status: NDF, DF Address: 192.168.1.1

```

```

VN Identifier: 300

```

```

IPv4 Multicast DF Status: NDF, DF Address: 192.168.1.1
.

```


Preference-based PEG DF Election with the use-least option—BL2 has the lowest preference value, 100, so in that case BL2 is the elected PEG DF here for VNIs 100 and 200:

```
user@BL2> show evpn oism peg-df-status
EVPN L3 context: VRF-1

VN Identifier: 100
IPv4 Multicast DF Status: DF, DF Address: 192.168.2.1

VN Identifier: 200
IPv4 Multicast DF Status: DF, DF Address: 192.168.2.1
```

3. `show route table <evpn-instance-name>.evpn.0 match-prefix 3:* extensive`—Verify that the PEG devices (the PEG DF election candidates) have the DF election extended community, in addition to the multicast flags extended community, in the EVPN Type 3 IMET routes in the EVPN instance routing table. For example:

Mod-based PEG DF Election:

```
user@BL1> show route table evpnA.evpn.0 match-prefix 3:192.168.1.1* extensive
3:192.168.1.1:9::2::192.168.1.1/248 IM (1 entry, 1 announced)
  *EVPN   Preference: 170
    Next hop type: Indirect, Next hop index: 0
    Address: 0x7647a14
    Next-hop reference count: 98
    Kernel Table Id: 0
    Protocol next hop: 192.168.1.1
    Indirect next hop: 0x0 - INH Session ID: 0
    Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
    State: <Active Int Ext>
    Age: 6:45
    Validation State: unverified
    Task: evpn-vxlan-A-evpn
    Announcement bits (1): 2-rt-export
    AS path: I
    Communities: encapsulation:vxlan(0x8) DF election:mod(0x0):0 evpn-mcast-
flags:0x19:igmp-snooping-enabled:oism:peg
    Route Label: 2
```

```
PMSI: Flags 0x0: Label 2: Type INGRESS-REPLICATION 192.168.1.1
Thread: junos-main
```

Preference-based PEG DF Election (truncated to show only the extended community values):

```
.
.
.

Communities: encapsulation:vxlan(0x8) DF election:preference(0x2):200 evpn-mcast-
flags:0x19:igmp-snooping-enabled:oism:peg
```

4. `show pim interfaces instance vrf-instance-name`—Verify the DF election status of the PIM IRB interfaces for each L3 routing instance matches the DF election results from the PEG DF election process. The device relays the PEG DF election status to the PIM protocol processes because OISM relies on PIM to create the multicast routes. For example:

```
user@BL1> show pim interfaces instance VRF-1
Stat = Status, V = Version, NbrCnt = Neighbor Count,
S = Sparse, D = Dense, B = Bidirectional,
DR = Designated Router, DDR = Dual DR, DistDR = Distributed DR,
P2P = Point-to-point link, P2MP = Point-to-Multipoint,
Active = Bidirectional is active, NotCap = Not Bidirectional Capable,
EVPN = EVPN Driven DR state
```

Name	Stat	Mode	IP	V	State	NbrCnt	JoinCnt(sg/*g)	DR address
irb.2	Up	S	4	2	EVPN,DR,NotCap	0	0/0	192.168.1.1
irb.3	Up	S	4	2	EVPN,DistDR,NotCap	0	0/0	192.168.2.1
irb.4	Up	S	4	2	EVPN,DR,NotCap	0	0/0	192.168.1.1
lo0.1	Up	S	4	2	DR,NotCap	0	0/0	192.168.1.2
lsi.0	Up	SD	4	2	P2P,NotCap	0	0/0	
pime.32769	Up	S	4	2	P2P,NotCap	0	0/0	
lsi.0	Up	SD	6	2	P2P,NotCap	0	0/0	

5. `show pim join instance vrf-instance-name extensive`—Verify that only the PEG device that is elected as the DF on the SBD sends PIM Join messages to an external PIM router (the PIM RP). The device that sends the PIM Join pulls in externally sourced traffic that is destined for multicast receivers within the EVPN data center. For example:

```
user@BL1> show pim join instance VRF-1 extensive
Instance: PIM.VRF-1 Family: INET
```

R = Rendezvous Point Tree, S = Sparse, W = Wildcard

```
Group: 233.252.0.1
  Source: *
  RP: 172.30.7.7
  Flags: sparse,rptree,wildcard
  Upstream interface: irb.1001
  Upstream neighbor: 192.168.1.103
  Upstream state: Join to RP
  Uptime: 00:00:32
  Downstream neighbors:
    Interface: irb.4
      172.16.1.1 State: Join Flags: SRW Timeout: Infinity
      Uptime: 00:00:32 Time since last Join: 00:00:32
  Number of downstream interfaces: 1
  Number of downstream neighbors: 1
```

6. `show pim rps instance vrf-instance-name extensive`—Verify that only the PEG device that is elected as the DF for an OISM revenue VLAN or the SBD sends the PIM Register messages to the external PIM RP for multicast sources inside the EVPN data center.

```
user@BL1> show pim rps instance VRF-1 extensive
```

```
Instance: PIM.VRF-1
```

```
address-family INET
```

```
RP: 172.30.7.7
```

```
Learned via: static configuration
```

```
Mode: Sparse
```

```
Time Active: 6d 17:20:07
```

```
Holdtime: 0
```

```
Device Index: 25
```

```
Subunit: 32769
```

```
Interface: pime.32769
```

```
Static RP Override: Off
```

```
Group Ranges:
```

```
  233.252.0.0/4
```

```
Active groups using RP:
```

```
  233.252.0.1
```

```
    total 1 groups active
```

```
Register State for RP:
```

Group	Source	FirstHop	RP Address	State	Timeout
233.252.0.1	10.1.1.11	192.168.1.2	172.30.7.7	Suppress	47

Look for the elected DF that sends the PIM Register messages for:

- The source VLAN when you configure regular OISM. In this case, the PEG devices receive source traffic from inside the data center on the source revenue VLAN.
- The SBD when you configure enhanced OISM (where you don't need to configure all revenue VLANs on all devices in the fabric). In this case, the PEG devices receive source traffic from inside the data center on the SBD.

Statically Identify Multihoming Peers With Enhanced OISM To Improve Convergence

OISM leaf devices are multihoming peers when they share an Ethernet segment (ES) for an attached multihomed client host or CE device. With enhanced OISM, the ingress leaf devices send east-west traffic:

- On the source VLAN to their multihoming peer leaf devices.
- On the SBD to any other OISM leaf devices.

If one of a pair of multihoming peer OISM leaf devices receives multicast source traffic, the device forwards the traffic to its multihoming peer on the source VLAN. However, if one of the multihomed client connections fails, those two OISM leaf devices are not multihoming peers anymore. As a result, the ingress OISM leaf device starts routing the traffic to the SBD instead. When the multihomed client connection comes up again, the ingress OISM leaf device switches back to forwarding the traffic on the source VLAN.

When multihomed connections go up and down, the multihoming peer devices need to repeatedly converge on the new core next hops to use either the source VLAN or the SBD. When this happens, the devices can lose some multicast traffic.

To avoid this situation, starting in Junos OS Release 24.2R1 on supported devices, you can statically identify the device's multihoming peer OISM leaf devices by their device loopback IP addresses.

In Junos OS Release 24.2R1, use the `multihoming-peer-gateways` statement at the `[edit protocols evpn]` hierarchy level to perform this function. Starting in Junos OS and Junos OS Evolved Release 24.4R1, you must instead use the `static-multihoming-peer` statement at the `[edit protocols evpn]` hierarchy level for this function. The `multihoming-peer-gateways` statement isn't available in the Junos OS CLI anymore after Junos OS Release 24.2R1.

With this setting, the device always forwards multicast traffic to its multihoming peers on the source VLAN, even when a multihomed client connection to one of the peers might be down. When multihomed client connections are flapping, the device doesn't need to keep switching between forwarding on the source VLAN and routing on the SBD.

For example, on a OISM leaf device SL-1 with loopback address 192.168.1.1 that has a multihoming peer SL-2 with loopback address 192.168.1.2, configure the following:

- On SL-1, statically identify SL-2 as the multihoming peer of SL-1:

```
set protocols evpn static-multihoming-peer 192.168.1.2
```

- On SL-2, statically identify SL-1 as the multihoming peer of SL-2:

```
set protocols evpn static-multihoming-peer 192.168.1.1
```

This statement applies only when you configure enhanced OISM mode.

Enhanced OISM with an EVPN-VXLAN IPv6 Underlay Configuration

On supported platforms you can enable enhanced OISM with IPv6 underlay peering for IPv4 and IPv6 multicast data traffic. An IPv6 underlay EVPN-VXLAN configuration enables the expanded addressing capabilities and efficient packet processing that the IPv6 protocol offers. See ["EVPN-VXLAN with an IPv6 Underlay" on page 897](#).

To configure enhanced OISM in an EVPN-VXLAN network with an IPv6 underlay:

1. Configure the EVPN-VXLAN IPv6 underlay:

Configure the IPv6 underlay the same way you would configure it without OISM. See ["Configure an IPv6 Underlay with EVPN-VXLAN" on page 903](#).

Note that:

- With enhanced OISM and an IPv6 underlay, we support only EBGp or OSPFv3 for the IPv6 underlay peering.
- You must use the `mac-vrf` instance type for the EVPN instances.
- We support both IPv4 and IPv6 multicast data traffic over an IPv6 underlay with enhanced OISM.

2. Configure enhanced OISM:

Configure the enhanced OISM elements for your multicast EVPN-VXLAN environment in the same way you would configure these elements in an EVPN-VXLAN network with an IPv4 underlay.

Note that:

- We don't support regular OISM with an IPv6 underlay, only enhanced OISM.
- You can use enhanced OISM with an IPv6 underlay for:

- IPv4 multicast data traffic with IGMPv1, IGMPv2, and IGMP snooping.
- IPv6 multicast data traffic with MLDv1, MLDv2, and MLD snooping.



NOTE: You can configure EX4100 and EX4400 switches as enhanced OISM server leaf devices. You can configure other devices that support enhanced OISM as border leaf devices or server leaf devices.

On supported platforms, you can also use enhanced OISM with seamless DCI stitching for EVPN-VXLAN data centers that are configured with IPv6 underlay peering. In each EVPN-VXLAN data center, you configure an IPv6 underlay, configure OISM, and configure DCI the same way you would configure each of these feature individually. See ["EVPN-VXLAN DCI Multicast with Enhanced OISM" on page 1188](#) for details on how the combination of these features works.

Enhanced OISM Exception Policy to Forward on Source VLAN Instead of SBD for Packets with TTL=1

Enhanced OISM routes most traffic coming from a multicast source on the SBD rather than on the source VLAN, even if the destination OISM device hosts the source VLAN. When OISM devices route multicast data packets to the SBD and then to the destination VLAN, they decrement the packet TTL more than once. As a result, packets with TTL=1 won't reach the receivers. This problem applies to traffic for any multicast groups other than 224.0.0.0/24 (for IPv4 multicast) and ff02::/16 (for IPv6 multicast).

Regular OISM mode forwards ingress multicast traffic on the source VLAN, so you don't see the same problem when running OISM in regular mode. For sites that want to run OISM in enhanced mode with TTL=1 traffic, we provide a solution that enables the regular OISM forwarding model on devices running enhanced OISM.

With this solution, you can configure an exception policy on enhanced OISM devices to use the source VLAN instead of the SBD for remote receivers that subscribe to specific multicast groups (or sources and groups), as follows:

1. Configure one or more routing policies to match the multicast groups (*,G), or multicast sources and groups (S,G), for which you want to send multicast traffic on the source VLAN instead of on the SBD.

Use the policy-statement *policy-name* statement at the [edit policy-options] hierarchy level. Define the *term* parameters in the *from* clause of the policy as follows:

- Match on a multicast group address using the *route-filter* policy statement option.
- Match on a multicast source address using the *source-address-filter* policy statement option.
- To match (S,G) SSM traffic for a particular source, include both the *source-address-filter* and *route-filter* options in a policy *from* term.



NOTE: A policy that matches only on a multicast group can also apply to SSM traffic for that group and any source.

2. In the relevant tenant L3 VRF instance, enable the policy using the *forward-on-source-bridge-domain* statement at the [edit routing-instances *L3-VRF-instance-name* protocols evpn oism enhanced] hierarchy level.

With the *forward-on-source-bridge-domain* statement, include the *forward-policy* option with one or more of the policy names you set in Step "1" on page 1141.

You can apply multiple policies with the *forward-policy* option.

See "[Sample TTL=1 Exception Policy Configuration](#)" on page 1143 for a simple example configuration. This TTL=1 exception policy feature works only with enhanced OISM.

The PFE logs a message (in syslog) when the device receives TTL=1 EVPN multicast packets, so you can see if there are multicast group you might want to include in a TTL=1 exception policy.

TTL=1 Exception Policy Support and Limitations

We support configuring this enhanced OISM exception policy feature with:

- EVPN instances of instance type `mac-vrf` only.
- IGMPv2, IGMPv3, and IGMP snooping in proxy mode.
- MLDv1, MLDv2, and MLD snooping in proxy mode.
- IPv4 or IPv6 underlay peering in the EVPN network.
- Seamless DCI stitching across EVPN-VXLAN data centers (see "[EVPN-VXLAN DCI Multicast with Enhanced OISM](#)" on page 1188) with either IPv4 underlay peering or IPv6 underlay peering.



NOTE: With DCI stitching, be sure to apply the same exception policies uniformly across all enhanced OISM devices in the interconnected data centers. The ingress OISM leaf devices in a data center send the multicast traffic on the source VLAN to the DCI gateway devices for groups that match an exception policy. Then the DCI gateway devices also send the matching traffic on the source VLAN across the interconnection to the other DCI gateway devices.

Keep the following behaviors and limitations in mind when you configure TTL=1 exception policies:

- You must configure the same exception policies uniformly on all enhanced OISM devices in the network.

- Remote devices must host the source VLAN to ensure multicast streams that match the configured exception policies reach those devices.
- Traffic that doesn't match any enabled TTL=1 exception policies follows the usual enhanced OISM model and forwards that traffic only on the SBD.
- We don't support multicast flows that send packets with TTL=1 and with TTL>1 for the same multicast group and source.

Sample TTL=1 Exception Policy Configuration

See the following sample exception policy configurations:

- A sample policy pol1 to match traffic with multicast group addresses in the 233.252.0.1/24 (or longer) range. This policy can match either ASM (*,G) traffic or SSM (S,G) traffic with any source.

```
user@leaf3> show configuration policy-options pol1
term 1 {
    from {
        route-filter 233.252.0.1/24 orlonger;
    }
    then accept;
}
term 2 {
    then reject;
}
```

- A sample policy pol2 to match SSM (S,G) traffic with group range 233.252.1.1/24 (or longer) and only a specific multicast source address, 172.16.1.1.

```
user@leaf3> show configuration policy-options pol2
term 1 {
    from {
        route-filter 233.252.1.1/24 orlonger;
        source-address-filter 172.16.1.1/32 exact;
    }
    then accept;
}
term 2 {
    then reject;
}
```


The configuration below enables enhanced OISM forwarding on the source VLAN in tenant L3 VRF instance VRF-1 with policy pol1 for ASM (*,G) flows. If you have SSM (S,G) flows in this VRF instance, this policy can also match flows for that group with any source.

```
user@leaf3> show configuration routing-instances L3VRF-1
instance-type vrf;
routing-options {
  router-id 192.168.3.2;
}
protocols {
  evpn {
    oism {
      supplemental-bridge-domain-irb irb.4;
      enhanced {
        forward-on-source-bridge-domain {
          forward-policy [pol1]
        }
      }
    }
  }
}
```

Verify TTL=1 Exception Policies on an Ingress Leaf Device

Use the following CLI commands to verify that an enhanced OISM ingress device forwards multicast traffic on the source VLAN instead of routing the traffic to the SBD. The ingress device is the leaf device that initially receives the traffic from the multicast source. In this case, the ingress device is an enhanced OISM leaf device called leaf3 with device address 192.168.3.1, with VTEPs to other OISM leaf devices 192.168.1.1, 192.168.2.1, 192.168.4.1, and 192.168.5.1.

The outputs here are based on this exception policy configuration:

```
set policy-options policy-statement pol1 term 1 from route-filter 233.252.0.1/24 orlonger
set policy-options policy-statement pol1 term 1 then accept
set policy-options policy-statement pol1 term 2 then reject
set routing-instances VRF-1 protocols evpn oism enhanced forward-on-source-bridge-domain forward-policy pol1
```

- `show igmp snooping evpn status`—Verify the EVPN OISM mode, the VLANs associated with the EVPN instance, and which VLAN is the OISM SBD. For example:

```
user@leaf> show igmp snooping evpn status detail
```

```
Mode: Enhanced OISM
```

```
SMET enabled: Yes
```

```
Instance: evpn-vxlan-A
```

```
Bridge-Domain: VLAN_2, VN Identifier: 2
```

```
OISM           : Enabled
```

```
Supplementary BD: No
```

```
External VLAN  : No
```

```
DCI status      : Disabled
```

```
Bridge-Domain: VLAN_4, VN Identifier: 4
```

```
OISM           : Enabled
```

```
Supplementary BD: Yes
```

```
External VLAN  : No
```

```
DCI status      : Disabled
```

```
Bridge-Domain: VLAN_5, VN Identifier: 5
```

```
OISM           : Enabled
```

```
Supplementary BD: No
```

```
External VLAN  : No
```

```
DCI status      : Disabled
```

You see similar information for IPv6 multicast traffic with enhanced OISM, MLD, and MLD snooping when you use the `show mld snooping evpn status detail` command.

- `show evpn igmp-snooping proxy`—Verify the device creates snooping proxy entries for the groups in the forward-on-source-bridge-domain policy. Those entries display **1** in the **Enhanced OISM Forward on Source BD** output field. For example, for VLAN_2 (with VNI or L2 domain ID 2):

```
user@leaf> show evpn igmp-snooping proxy internal l2-domain-id 2
```

```
Instance: evpn-vxlan-A
```

```
VN Identifier: 2
```

```
Group IP: 233.252.0.1, Source IP: 0.0.0.0
```

```
SMET Flood: 0, Enhanced OISM Local Source: 0
```

```
L3-MGM Notified: 1, Local Refreshed: 1
```

```
Enhanced-OISM Forward on SRC BD: 1
```

```
MCSN Added: 1
```

```
In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
```

```
Peer IP: 192.168.4.1, Flags: 0x0
```

```
Peer IP: 192.168.1.1, Flags: 0x2
```

```

    Peer IP: 192.168.2.1, Flags: 0x2
Group IP: 233.252.0.2, Source IP: 0.0.0.0
    SMET Flood: 0, Enhanced OISM Local Source: 0
    L3-MGM Notified: 1, Local Refreshed: 1
Enhanced-OISM Forward on SRC BD: 1
MCSN Added: 1
In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
    Peer IP: 192.168.4.1, Flags: 0x0
    Peer IP: 192.168.1.1, Flags: 0x2
    Peer IP: 192.168.2.1, Flags: 0x2

```

- `show igmp snooping evpn proxy`—Check that the snooping proxy state on the device for the multicast groups (or multicast groups and sources) in the policy also show **S** in the **Flags** field to indicate forwarding on the source VLAN is enabled. For example:

```

user@leaf> show igmp snooping evpn proxy
Legend for Flags:
    S - Enhanced-OISM Forward on source bridge domain
Instance: evpn-vxlan-A
Bridge-Domain: VLAN_2, VN Identifier: 2

```

Group	Source	Local	Remote	Flags
233.252.0.1	0.0.0.0	1	1	S
233.252.0.2	0.0.0.0	1	1	S

```

Bridge-Domain: VLAN_4, VN Identifier: 4

```

Group	Source	Local	Remote	Flags
0.0.0.0	0.0.0.0	0	1	
233.252.0.1	0.0.0.0	0	1	S
233.252.0.2	0.0.0.0	0	1	S

```

Bridge-Domain: VLAN_5, VN Identifier: 5

```

Group	Source	Local	Remote	Flags
233.252.0.1	0.0.0.0	0	1	S
233.252.0.2	0.0.0.0	0	1	S

The command displays similar output but with IPv6 multicast groups and source addresses when you use the `show mld snooping evpn proxy` command with MLD traffic.

- `show evpn igmp-snooping proxy extensive`—Verify the source device creates IGMP snooping proxy entries on all VLANs for the groups that match the enabled policy or policies. The core next hop IDs in the **Corenh** column are 0 in the proxy entries for the SBD (which is VLAN_4 in this case) to indicate that the device doesn't forward the matching traffic on the SBD. The non-zero **Corenh** values for the revenue VLANs are the remote VTEP next hop IDs for that VLAN. The device forwards the traffic on the source VLAN using those next hops based on received SMET EVPN Type 6 routes.

For example:

```

user@leaf> show evpn igmp-snooping proxy extensive
Instance: evpn-vxlan-A
  VN Identifier: 2
    Group      Source   Local  Remote  Remote DC  Corenh  Flood/Lclsrc/L3-MGM/Lcl-
Refresh
    233.252.0.1 0.0.0.0   1      3       0          4403   0/0/1/1
    233.252.0.2 0.0.0.0   1      3       0          4403   0/0/1/1
  VN Identifier: 4
    Group      Source   Local  Remote  Remote DC  Corenh  Flood/Lclsrc/L3-MGM/Lcl-
Refresh
    0.0.0.0     0.0.0.0   0      2       0          0       0/0/0/0
    233.252.0.1 0.0.0.0   1      3       0          0       0/0/1/1
    233.252.0.2 0.0.0.0   1      3       0          0       0/0/1/1
  VN Identifier: 5
    Group      Source   Local  Remote  Remote DC  Corenh  Flood/Lclsrc/L3-MGM/Lcl-
Refresh
    233.252.0.1 0.0.0.0   0      3       0          4403   0/0/0/0
    233.252.0.2 0.0.0.0   0      3       0          4403   0/0/0/0

```

The command displays similar output but with IPv6 multicast groups and source addresses when you use the `show evpn mld-snooping proxy` command with MLD traffic.

- `show evpn multicast-snooping next-hops next-hop-id detail`, `show interfaces vtep.vtep-id`, and `show route table evpn-instance-name.evpn.0 match-prefix 6:...`—Check that the core next hops in the output from the `show igmp snooping evpn proxy extensive` command (see above) include the VTEPs of the remote enhanced OISM devices with interested receivers. The remote devices with interested receivers are those that advertised SMET EVPN Type 6 routes. The list should also have next hops to the OISM PEG devices to ensure traffic reaches any interested external receivers—look for each PEG device's address in the "VXLAN Endpoint Address" output field when you run the `show interfaces vtep.vtep-id` command for all the VTEPs in the next hop.



NOTE: SMET Type 6 routes are still advertised on the SBD. However, the device creates snooping proxy entries on the revenue VLANs for multicast groups (or multicast groups and sources) that match the configured policy or policies.

For example:

```

user@leaf3> show evpn multicast-snooping next-hops 4403 detail
Family: INET

```

```

ID          Refcount KRefCount Downstream interface Addr
4403        16        5 vtep.32773-(11414)
                vtep.32774-(11415)
                vtep.32775-(11416)
                Flags 0x2100 type 0x18 members 0/0/0/3/0
                Address 0x5582e89cca44

user@leaf3> show interfaces vtep.32773 | grep Addr
    VXLAN Endpoint Type: Shared Remote, VXLAN Endpoint Address: 192.168.1.1, L3 Routing
Instance: default

user@leaf3> show interfaces vtep.32774 | grep Addr
    VXLAN Endpoint Type: Shared Remote, VXLAN Endpoint Address: 192.168.2.1, L3 Routing
Instance: default

user@leaf3> show interfaces vtep.32775 | grep Addr
    VXLAN Endpoint Type: Shared Remote, VXLAN Endpoint Address: 192.168.4.1, L3 Routing
Instance: default

user@leaf3> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*233.252.0.1* | grep "6:" |
except 192.168.3.1
6:192.168.1.1:9::4::233.252.0.1::192.168.1.1/520
    *[BGP/170] 19:16:53, localpref 100, from 192.168.1.1
    [BGP/170] 19:16:52, localpref 100, from 192.168.2.1
    [BGP/170] 19:16:53, localpref 100, from 192.168.4.1
6:192.168.2.1:9::4::233.252.0.1::192.168.2.1/520
    *[BGP/170] 19:16:53, localpref 100, from 192.168.2.1
    [BGP/170] 19:16:53, localpref 100, from 192.168.1.1
    [BGP/170] 19:16:53, localpref 100, from 192.168.4.1
6:192.168.4.1:9::4::233.252.0.1::192.168.4.1/520
    *[BGP/170] 19:16:40, localpref 100, from 192.168.4.1
    [BGP/170] 19:16:40, localpref 100, from 192.168.1.1
    [BGP/170] 19:16:40, localpref 100, from 192.168.2.1

```

Use the same commands to verify operation with MLD, MLD snooping, and IPv6 multicast traffic.

Note that these commands are also applicable with DCI stitching and enhanced OISM. In that use case, the OISM PEG devices advertise EVPN Type 3 IMET routes in the stitched network that include the `peg` flag in the EVPN multicast flags extended community. Using these IMET routes, the non-PEG OISM devices in one data center can send multicast traffic across the interconnection to the remote PEG devices in another data center to reach subscribed external users.

Run the `show route table` command with options such as the following to see the received IMET routes that have the `peg` flag, which identifies the PEG devices in the stitched network. Then look for those device addresses when you run the `show interfaces vtep.vtep-id` command to make sure the device has next hops to the PEG devices.

```
user@leaf3> show route table evpn-vxlan-A.evpn.0 match-prefix 3* extensive | match "IM|peg" |
except "Import|Primary"
3:192.168.1.1:9::2::192.168.1.1/248 IM (1 entry, 1 announced)
      Communities: target:9:9 encapsulation:vxlan(0x8) evpn-mcast-flags:0x1b:igmp-
snooping-enabled:mld-snooping-enabled:oism:peg
3:192.168.1.1:9::3::192.168.1.1/248 IM (1 entry, 1 announced)
      Communities: target:9:9 encapsulation:vxlan(0x8) evpn-mcast-flags:0x1b:igmp-
snooping-enabled:mld-snooping-enabled:oism:peg
```

See ["Verify TTL=1 Exception Policies on a DCI Gateway Device" on page 1149](#) next for more on how to verify forwarding on the source VLAN is enabled with DCI stitching and enhanced OISM.

Verify TTL=1 Exception Policies on a DCI Gateway Device

You can use these CLI commands on DCI gateway devices for the same verification steps we describe for OISM ingress leaf devices in ["Verify TTL=1 Exception Policies on an Ingress Leaf Device" on page 1144](#):

- `show igmp snooping evpn status detail`—Verify the EVPN OISM mode, the VLANs associated with the EVPN instance, and which VLAN is the OISM SBD.
- `show evpn igmp-snooping proxy internal l2-domain-id vni`—Verify the device creates snooping proxy entries for the groups in the exception policy (**1** in the **Enhanced OISM Forward on Source BD** output field).
- `show igmp snooping evpn proxy`—Verify that the snooping proxy state on the device for the multicast groups (or multicast groups and sources) in the policy also show **S** in the **Flags** field to indicate forwarding on the source VLAN is enabled.

Additionally use the CLI commands in this section to verify that an enhanced OISM device acting as a DCI gateway device forwards multicast traffic across the interconnection on the source VLAN instead of routing the traffic to the SBD.

For DCI support, when you enable a TTL=1 exception policy on a DCI gateway device, the device creates unilist entries for the revenue VLAN L2 multicast routes (for snooping) and L3 multicast routes to other DCI gateway devices. For multicast snooping, the gateway device creates unilist entries for both (*,G) and (S,G) multicast routes. An (S,G) entry for a revenue VLAN inherits the (*,G) unilist next hop. However, the device creates a normal multicast next hop instead of a unilist next hop with an (S,G)

entry for an L3 route output interface. The output from some of the show commands listed here indicate when a snooping next hop corresponds to a unilist entry; otherwise the next hop is a normal multicast next hop.



NOTE: DCI gateway devices create unilist entries for fast convergence when forwarding to and from other DCI gateway devices only.

The outputs here are based on the following exception policy configuration:

```
set policy-options policy-statement pol1 term 1 from route-filter 233.252.0.1/24 orlonger
set policy-options policy-statement pol1 term 1 then accept
set policy-options policy-statement pol1 term 2 then reject
set routing-instances VRF-1 protocols evpn oism enhanced forward-on-source-bridge-domain forward-policy pol1
```

- `show evpn igmp-snooping proxy extensive`—Verify the device has IGMP snooping proxy entries on all VLANs for the groups that match the enabled policy or policies. The core next hop IDs in the **Corenh** column of the output are 0 in the proxy entries for the SBD (VLAN_4 in this case) to indicate that the device doesn't forward matching traffic on the SBD. The non-zero **Corenh** values for the revenue VLANs are the remote VTEP next hop IDs for that VLAN. The device forwards the traffic on the source VLAN using those next hops based on received SMET EVPN Type 6 routes.

For example:

```
user@iGW-11> show evpn igmp-snooping proxy extensive
Instance: evpn-vxlan-A
  VN Identifier: 2
    Group      Source  Local  Remote  Remote DC  Corenh  Flood/Lclsrc/L3-MGM/Lcl-Refresh
    0.0.0.0    0.0.0.0   1      1       0          0       0/0/0/1
    233.252.0.1 0.0.0.0   1      3       2          524300  0/0/1/1
    233.252.0.2 0.0.0.0   1      3       2          524300  0/0/1/1
  VN Identifier: 4
    Group      Source  Local  Remote  Remote DC  Corenh  Flood/Lclsrc/L3-MGM/Lcl-Refresh
    0.0.0.0    0.0.0.0   1      1       0          0       0/0/0/1
    233.252.0.1 0.0.0.0   1      3       2          524409  0/0/1/1
    233.252.0.1 172.16.1.1 0      0       0          0       0/1/0/0
    233.252.0.2 0.0.0.0   1      3       2          524409  0/0/1/1
    233.252.0.2 172.16.1.1 0      0       0          0       0/1/0/0
  VN Identifier: 5
```

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-
Refresh						
0.0.0.0	0.0.0.0	1	1	0	0	0/0/0/1
233.252.0.1	0.0.0.0	1	3	2	524300	0/0/1/1
233.252.0.2	0.0.0.0	1	3	2	524300	0/0/1/1



NOTE: Enhanced OISM uses special handling when creating proxy entries learned from local sources, indicated by a '1' in the Lclsrc output field. For example, in the sample output here, the (S,G) entries for the SBD (VLAN_4 and VNI 4) weren't created from received EVPN Type 6 routes, but learned from local sources.

Also, the SBD core next hop might be non-zero on OISM PEG devices for proxy entries that match the exception policy, and DCI gateway devices often might act as the OISM PEG devices too. A PEG device might use this next-hop to route external source traffic on the SBD. In that case, though, the device wouldn't route and forward external source traffic with TTL=1 on any tenant revenue VLANs.

- `show multicast route extensive instance vrf-instance`—Verify that the device has multicast routes with unilist next hops for traffic arriving on revenue VLANs for the groups (or groups and sources) that match the exception policy. For example:

```
user@iGW-11> show multicast route extensive instance VRF-1
Instance: VRF-1 Family: INET
```

```
Group: 233.252.0.1
```

```
Source: 172.16.1.1/32
```

```
Upstream interface: irb.2 (L2 NH: 524324)
```

```
EVPN DCI NDF upstream interface: irb.2 (L2 NH: 524332)
```

```
Downstream interface list:
```

```
    irb.4 (L2 NH: 1816) irb.5 (L2 NH: 524323)
```

```
Number of outgoing interfaces: 2
```

```
EVPN DCI NDF downstream interface list:
```

```
    irb.4 (L2 NH: 1816) irb.5 (L2 NH: 524323)
```

```
Number of EVPN DCI NDF outgoing interfaces: 2
```

```
Session description: Unknown
```

```
Statistics: 371 kbps, 355 pps, 3497287903 packets
```

```
Next-hop ID: 524352 (Unilist NH)
```

```
Upstream protocol: Multicast
```

```
Route state: Active
```

```
Forwarding state: Forwarding
```

```
Cache lifetime/timeout: 360 seconds
```



```

Wrong incoming interface notifications: 0
Uptime: 00:28:19

Group: 233.252.0.2
Source: 172.16.1.1/32
Upstream interface: irb.2 (L2 NH: 524324)
EVPN DCI NDF upstream interface: irb.2 (L2 NH: 524332)
Downstream interface list:
    irb.4 (L2 NH: 1816) irb.5 (L2 NH: 524323)
Number of outgoing interfaces: 2
EVPN DCI NDF downstream interface list:
    irb.4 (L2 NH: 1816) irb.5 (L2 NH: 524323)
Number of EVPN DCI NDF outgoing interfaces: 2
Session description: Unknown
Statistics: 371 kBps, 355 pps, 3666423158 packets
Next-hop ID: 524352 (Unilist NH)
Upstream protocol: Multicast
Route state: Active
Forwarding state: Forwarding
Cache lifetime/timeout: 360 seconds
Wrong incoming interface notifications: 0
Uptime: 00:28:19

Instance: VRF-1 Family: INET6

user@iGW-11>

```

- `show multicast snooping route extensive instance evpn-instance vlan vlan-name`—Verify that the device has active snooping routes (in this case, for revenue VLANs VLAN_2 and VLAN-5) for:
 - Unilist next hops for (*,G) and (S,G) routes for input interface routes matched with snooping next hops. The (S,G) routes use the same next hops as the (*,G) routes for the same group.
 - Normal multicast nexthops for (S,G) routes for output interface routes matched to snooping next hops.

For example:

```

user@iGW-11> show multicast snooping route extensive instance evpn-vxlan-A vlan VLAN_2
Nexthop Bulking: OFF

Family: INET

```

Group: 224.0.0.0/4

Source: *

Vlan: VLAN_2

Mesh-group: __all_ces__

Downstream interface list:

evpn-core-nh -(524409)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 524413

Route state: Active

Forwarding state: Forwarding

Group: 233.252.0.1/32

Source: *

Vlan: VLAN_2

Mesh-group: __all_ces__

Downstream interface list:

evpn-dci-df-nh -(524324) evpn-dci-ndf-nh -(524332)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 0 (unilist)

Route state: Inactive

Forwarding state: Forwarding

Group: 233.252.0.1/32

Source: 172.16.1.1/32

Vlan: VLAN_2

Mesh-group: __all_ces__

Downstream interface list:

evpn-dci-df-nh -(524324) evpn-dci-ndf-nh -(524332)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 524333 (unilist)

Route state: Active

Forwarding state: Forwarding

Group: 233.252.0.2/32

Source: *

Vlan: VLAN_2

Mesh-group: __all_ces__

Downstream interface list:

evpn-dci-df-nh -(524324) evpn-dci-ndf-nh -(524332)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 0 (unilist)

Route state: Inactive

Forwarding state: Forwarding

Group: 233.252.0.2/32

Source: 172.16.1.1/32

Vlan: VLAN_2

Mesh-group: __all_ces__

Downstream interface list:

evpn-dci-df-nh -(524324) evpn-dci-ndf-nh -(524332)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 524333 (unilist)

Route state: Active

Forwarding state: Forwarding

user@iGW-11> **show multicast snooping route extensive instance evpn-vxlan-A vlan VLAN_5**

Nextthop Bulking: OFF

Family: INET

Group: 224.0.0.0/4

Source: *

Vlan: VLAN_5

Mesh-group: __all_ces__

Downstream interface list:

evpn-core-nh -(524409)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 524413

Route state: Active

Forwarding state: Forwarding

Group: 233.252.0.1/32

Source: *

Vlan: VLAN_5

Mesh-group: __all_ces__

Downstream interface list:

evpn-dci-df-nh -(0) evpn-dci-ndf-nh -(524323)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 0 (unilist)

Route state: Inactive

Forwarding state: Forwarding

Group: 233.252.0.1/32

Source: 172.16.1.1/32

Vlan: VLAN_5

```

Mesh-group: __all_ces__
  Downstream interface list:
    ae0.0 -(1708)
Statistics: 0 kbps, 0 pps, 0 packets
Next-hop ID: 524323
Route state: Active
Forwarding state: Forwarding

Group: 233.252.0.2/32
Source: *
Vlan: VLAN_5
Mesh-group: __all_ces__
  Downstream interface list:
    evpn-dci-df-nh -(0) evpn-dci-ndf-nh -(524323)
Statistics: 0 kbps, 0 pps, 0 packets
Next-hop ID: 0 (unilist)
Route state: Inactive
Forwarding state: Forwarding

Group: 233.252.0.2/32
Source: 172.16.1.1/32
Vlan: VLAN_5
Mesh-group: __all_ces__
  Downstream interface list:
    ae0.0 -(1708)
Statistics: 0 kbps, 0 pps, 0 packets
Next-hop ID: 524323
Route state: Active
Forwarding state: Forwarding

```

- show multicast route **extensive instance VRF-1 source-prefix *source-address*** and show multicast snooping route **extensive instance evpn-instance VLAN *vlan-name* source-prefix *source-address***—When you use IGMPv3 for SSM with a TTL=1 exception policy, verify the device creates the following entries for (S,G) elaboration routes at L2 (snooping) and L3:
 - Unilist next hops for input interface elaboration.
 - Regular multicast next hops for output interface elaboration.

For example, if the SSM source address is 172.16.1.1:

```

user@iGW-11> show multicast route extensive instance VRF-1 source-prefix 172.16.1.1
Instance: VRF-1 Family: INET

```

Group: 233.252.0.1

Source: 172.16.1.1/32

Upstream interface: irb.2 (L2 NH: 524310)

EVPN DCI NDF upstream interface: irb.2 (L2 NH: 524366)

Downstream interface list:

irb.4 (L2 NH: 1686) irb.5 (L2 NH: 524366)

Number of outgoing interfaces: 2

EVPN DCI NDF downstream interface list:

irb.4 (L2 NH: 1686) irb.5 (L2 NH: 524366)

Number of EVPN DCI NDF outgoing interfaces: 2

Session description: Source specific multicast

Statistics: 528 kBps, 504 pps, 3509909282 packets

Next-hop ID: 524317 (Unilist NH)

Upstream protocol: Multicast

Route state: Active

Forwarding state: Forwarding

Cache lifetime/timeout: forever

Wrong incoming interface notifications: 2

Uptime: 00:07:30

Group: 233.252.0.2

Source: 172.16.1.1/32

Upstream interface: irb.2 (L2 NH: 524310)

EVPN DCI NDF upstream interface: irb.2 (L2 NH: 524366)

Downstream interface list:

irb.4 (L2 NH: 1686) irb.5 (L2 NH: 524366)

Number of outgoing interfaces: 2

EVPN DCI NDF downstream interface list:

irb.4 (L2 NH: 1686) irb.5 (L2 NH: 524366)

Number of EVPN DCI NDF outgoing interfaces: 2

Session description: Source specific multicast

Statistics: 528 kBps, 504 pps, 3509908960 packets

Next-hop ID: 524317 (Unilist NH)

Upstream protocol: Multicast

Route state: Active

Forwarding state: Forwarding

Cache lifetime/timeout: forever

Wrong incoming interface notifications: 2

Uptime: 00:07:30

user@iGW-11> **show multicast snooping route extensive instance evpn-vxlan-A vlan VLAN_2 source-prefix 172.16.1.1**

Nextthop Bulking: OFF

Family: INET

Group: 233.252.0.1/32

Source: 172.16.1.1/32

Vlan: VLAN_2

Mesh-group: __all_ces__

Downstream interface list:

evpn-dci-df-nh -(524310) evpn-dci-ndf-nh -(524366)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 524312 (unilist)

Route state: Active

Forwarding state: Forwarding

Group: 233.252.0.2/32

Source: 172.16.1.1/32

Vlan: VLAN_2

Mesh-group: __all_ces__

Downstream interface list:

evpn-dci-df-nh -(524310) evpn-dci-ndf-nh -(524366)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 524312 (unilist)

Route state: Active

Forwarding state: Forwarding

user@iGW-11> **show multicast snooping route extensive instance evpn-vxlan-A vlan VLAN_5 source-prefix 172.16.1.1**

NextHop Bulking: OFF

Family: INET

Group: 233.252.0.1/32

Source: 172.16.1.1/32

Vlan: VLAN_5

Mesh-group: __all_ces__

Downstream interface list:

ae0.0 -(1711) xe-0/0/0:0.0 -(1847)

Statistics: 0 kbps, 0 pps, 0 packets

Next-hop ID: 524366

Route state: Active

Forwarding state: Forwarding

Group: 233.252.0.2/32

```

Source: 172.16.1.1/32
Vlan: VLAN_5
Mesh-group: __all_ces__
    Downstream interface list:
        ae0.0 -(1711) xe-0/0/0:0.0 -(1847)
Statistics: 0 kbps, 0 pps, 0 packets
Next-hop ID: 524366
Route state: Active
Forwarding state: Forwarding

```

Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices

Follow these steps to configure elements common to border leaf and server leaf devices in an EVPN-VXLAN fabric running OISM.



NOTE: You also configure these common elements on spine devices that act as standalone AR replicators in a fabric with OISM.

This configuration is based on an EVPN-VXLAN fabric configuration that supports OISM and has:

- An EBGp underlay with Bidirectional Forwarding Detection (BFD) and Ethernet operations, administration and management (OAM) link detection.
- An ERB overlay design.
- Lean spine devices that act only as IP transit nodes in the fabric.
- Server leaf devices configured as EVPN-VXLAN L2 gateways.
- Border leaf devices configured as EVPN-VXLAN L2 and L3 gateways.

With regular OISM (symmetric bridge domains model), you must configure all of the revenue bridge domains and the supplemental bridge domain (SBD) on all OISM devices in the fabric. With enhanced OISM (asymmetric bridge domains model), on each leaf device you can configure only the VLANs that the device hosts, except you must configure the same revenue VLANs symmetrically on leaf devices that are multihoming peers. See ["Configuration Elements for OISM Devices" on page 1072](#) for a summary of what elements you configure on border leaf and server leaf devices, and why you configure those elements.

The sample configuration blocks we provide for these configuration steps use an OISM environment with the following elements:

- An EVPN instance configured in either the default switch instance (no routing instance specified) OR a MAC-VRF EVPN instance. For example:

- Default switch EVPN instance:

```
set protocols evpn encapsulation vxlan
```

- MAC-VRF EVPN instances (for each MAC-VRF instance):

```
set routing-instances mac-vrf-instance-name protocols evpn encapsulation vxlan
```

With a MAC-VRF EVPN instance configuration, you configure some elements in the MAC-VRF instances. In OISM configurations, we support either the `vlan-aware` or `vlan-based` MAC-VRF instance service type.

To illustrate sample configuration steps here, we show a MAC-VRF instance named `MAC-VRF1` with the `vlan-aware` service type. The main differences between the two service types are:

- `vlan-aware`: You can define more than one VLAN, its corresponding IRB interface, and VXLAN network identifier (VNI) mapping in the instance. As a result, you specify VLAN-related OISM or multicast configuration statements in one `vlan-aware` MAC-VRF instance for all of the VLANs in that instance.
 - `vlan-based`: You configure separate MAC-VRF instances in which you define each VLAN and its IRB interface and VNI mapping. As a result, you include similar VLAN-related OISM or multicast configuration statements for each VLAN in the corresponding `vlan-based` MAC-VRF instance.
- SBD: VLAN-300
 SBD IRB interface: `irb.300`
 SBD IRB interface IP address: `10.0.30.1`
 - Revenue bridge domains: VLAN-100 and VLAN-200
 Revenue bridge domain IRB interfaces: `irb.100` and `irb.200`
 Revenue bridge domain IRB interface IP addresses: `10.0.10.1` and `10.0.20.1`
 - L3 VRF routing instance: `L3VRF-1`



NOTE: Here we describe elements you configure in the tenant L3 VRF instances in the EVPN network. If you use the default L3 routing instance with OISM instead of VRF instances, substitute the corresponding configuration statements at the global hierarchy

level instead of in a named routing instance. For example, substitute the configuration statements at this hierarchy level:

```
[edit routing-instances L3VRF-1 protocols evpn ...]
```

with the same statements at this global hierarchy level:

```
[edit protocols evpn ...]
```

See ["OISM Support with Tenant L3 VRF Instances or the Default L3 Routing Instance"](#) on page 1052 for more on OISM configurations with the default L3 routing instance.

- If you use the M-VLAN IRB method for external multicast connectivity:

M-VLAN: VLAN-900

M-VLAN IRB interface: irb.900

M-VLAN IRB interface IP address: 172.16.90.1/24

Interface name of port connecting to external PIM router: xe-0/0/9



NOTE: You use the same M-VLAN ID and assign IRB interface IP addresses in the same subnet across any border leaf devices to which the external PIM router is multihomed.

- If you use the classic L3 interface method for external multicast connectivity:

L3 interface name: xe-0/0/6

L3 interface IP address: 172.16.10.1/24



NOTE: You assign IP addresses in different subnets for the L3 interfaces on each border leaf device connected to the external PIM router.

- If you use the non-EVPN IRB method for external multicast connectivity:

Extra VLAN: VLAN-900

Non-EVPN IRB interface: irb.900

Non-EVN IRB interface IP address: 172.16.90.1/24

Interface name of port connecting to external PIM router: xe-0/0/9



NOTE: With the non-EVPN IRB method, you assign distinct extra VLAN IDs on each border leaf device. You also assign IP addresses in different subnets for the non-EVPN IRB interfaces on each border leaf device connected to the external PIM router.

Configure these OISM statements on both border leaf devices and server leaf devices:

1. Enable OISM globally on the device.

Configure regular OISM using the `evpn irb oism` option or the enhanced OISM using the `evpn irb enhanced-oism` option (if supported), at the `[edit forwarding-options multicast-replication]` hierarchy level. This option enables OISM optimizations and external multicast capabilities in an ERB overlay fabric. This option takes the place of the `evpn irb local-only` option you would otherwise use in ERB overlay fabrics for multicast without OISM.

To enable regular OISM mode (symmetric bridge domains model):

```
set forwarding-options multicast-replication evpn irb oism
```

To enable enhanced OISM mode (asymmetric bridge domains model):

```
set forwarding-options multicast-replication evpn irb enhanced-oism
```



NOTE:

- All OISM devices in the network must use the same OISM mode.
- When you enable any multicast protocols with EVPN (not only with OISM), we require you have the multicast mode set to `ingress-replication` at the `[edit protocols evpn multicast-mode]` hierarchy level. On most platforms, `ingress-replication` is the default replication mode, but in any case you can include the following statement in your configuration:

```
set protocols evpn multicast-mode ingress-replication
```

- Some platforms, such as PTX10001-36MR, PTX10002-36QDD, PTX10004, PTX10008, and PTX10016 routers, support only OISM for multicast traffic with EVPN. In that case, the device posts a warning in the configuration if

you set the `evpn irb local-only` option or the `evpn irb local-remote` option, and ignores those configuration items.

2. (ACX Series routers with enhanced OISM enabled) Configure the `vlan-extended` profile option at the `[edit system packet-forwarding-options system-profile]` hierarchy level.

We require this system profile on ACX Series routers with enhanced OISM. See the `vlan-extended` statement for details.

```
set system packet-forwarding-options system-profile vlan-extended
```



NOTE: When you change the system profile, the packet forwarding engine (PFE) reboots.

3. Configure the revenue VLANs with an IRB interface and a VNI mapping for each revenue bridge domain. If your configuration uses MAC-VRF EVPN instances, you configure these elements in the MAC-VRF EVPN routing instances. With regular OISM, configure all revenue VLANs in the network on all leaf devices. With enhanced OISM, on each leaf device, you can configure only the VLANs that the device hosts, except you must configure the same revenue VLANs symmetrically on leaf devices that are multihoming peers.

For example, if the revenue bridge domains are VLAN-100 and VLAN-200:

With the default switch instance:

```
set vlans VLAN-100 vlan-id 100
set vlans VLAN-100 l3-interface irb.100
set vlans VLAN-100 vxlan vni 100

set vlans VLAN-200 vlan-id 200
set vlans VLAN-200 l3-interface irb.200
set vlans VLAN-200 vxlan vni 200
```

With a `vlan-aware` MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 vlans VLAN-100 vlan-id 100
set routing-instances MAC-VRF1 vlans VLAN-100 l3-interface irb.100
set routing-instances MAC-VRF1 vlans VLAN-100 vxlan vni 100

set routing-instances MAC-VRF1 vlans VLAN-200 vlan-id 200
```

```
set routing-instances MAC-VRF1 vlans VLAN-200 l3-interface irb.200
set routing-instances MAC-VRF1 vlans VLAN-200 vxlan vni 200
```

4. (Enhanced OISM only—Recommended for multihoming peer OISM leaf devices) On sets of multihoming peer OISM leaf devices, statically identify each device's multihoming peers using the peer device's loopback addresses. This step avoids multicast traffic loss when peer devices go up and down. See ["Statically Identify Multihoming Peers With Enhanced OISM To Improve Convergence" on page 1139](#) for details.



NOTE: In Junos OS Release 24.2R1, use the `multihoming-peer-gateways` statement at the `[edit protocols evpn]` hierarchy level to perform this function. Starting in Junos OS and Junos OS Evolved Release 24.4R1, the Junos CLI provides the `static-multihoming-peer` statement at the `[edit protocols evpn]` hierarchy level instead, and the `multihoming-peer-gateways` statement isn't available anymore.

For example, if two SL devices SL-1 (lo0: 192.168.1.1) and SL-2 (lo0: 192.168.1.2) are multihoming peers for a multihomed host or CE device:

On SL-1:

```
set protocols evpn static-multihoming-peer 192.168.1.2
```

On SL-2:

```
set protocols evpn static-multihoming-peer 192.168.1.1
```

5. Configure a VLAN for the SBD with an IRB interface and VXLAN VNI mapping. You must configure the SBD on all OISM devices, whether you are running regular OISM or enhanced OISM.

For example, if the SBD is VLAN-300:

With the default switch instance:

```
set vlans VLAN-300 vlan-id 300
set vlans VLAN-300 l3-interface irb.300
set vlans VLAN-300 vxlan vni 300
```

With a vlan-aware MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 vlans VLAN-300 vlan-id 300
set routing-instances MAC-VRF1 vlans VLAN-300 l3-interface irb.300
set routing-instances MAC-VRF1 vlans VLAN-300 vxlan vni 300
```

6. Configure the revenue bridge domain and the SBD IRB interface IP addresses. For example:

```
set interfaces irb unit 100 family inet address 10.0.10.1/24
set interfaces irb unit 200 family inet address 10.0.20.1/24
set interfaces irb unit 300 family inet address 10.0.30.1/24
```

7. Extend the revenue bridge domains and the SBD VNIs in the EVPN-VXLAN overlay. If your configuration uses MAC-VRF EVPN instances, you do this in the MAC-VRF EVPN routing instances.

For example, if the revenue bridge domains are VLAN-100 and VLAN-200, and the SBD is VLAN-300:

With the default switch instance:

```
set protocols evpn extended-vni-list 100
set protocols evpn extended-vni-list 200
set protocols evpn extended-vni-list 300
```

With a vlan-aware MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 protocols evpn extended-vni-list 100
set routing-instances MAC-VRF1 protocols evpn extended-vni-list 200
set routing-instances MAC-VRF1 protocols evpn extended-vni-list 300
```

8. Enable IGMPv2 or IGMPv3 on the device for IPv4 multicast traffic, or enable MLDv1 or MLDv2 for IPv6 multicast traffic. Here for simplicity we show how to enable either IGMP version or either MLD version globally on the device.

You can alternatively enable IGMP or MLD on the specific IRB interfaces included in the tenant L3 VRF instances that handle the multicast traffic.

In general, on all OISM devices, enable IGMP on the SBD IRB interface and the revenue bridge domain IRB interfaces. On border leaf devices, also enable IGMP on the external multicast interfaces (depending on the external multicast method used).

See ["IGMPv2 and IGMPv3 \(or MLDv1 and MLDv2\) in the Same EVPN-VXLAN Fabric" on page 1117](#) for more on how to configure IGMP snooping with both IGMPv2 and IGMPv2 together (or MLD snooping with MLDv1 and MLDv2 together) on a device.

Also, to enable IGMP or MLD on individual IRB interfaces that handle multicast traffic, include multiple `igmp` or `mld` statements, one for each IRB interface. For example, set `protocols igmp interface irb-interface-name <version 3>`, or set `protocols mld interface irb-interface-name <version 2>`.

- a. For IGMPv2, which is the default IGMP version, enable IGMP globally. You can also optionally specify version 2 for clarity if you have both IGMP versions in your configuration. The configuration is the same if you use a default switch EVPN instance or MAC-VRF EVPN instances.

```
set protocols igmp interface all
```

- b. For IGMPv3, enable IGMP globally with the version 3 option. The configuration is the same if you use a default switch EVPN instance or MAC-VRF EVPN instances.

```
set protocols igmp interface all version 3
```

- c. For MLDv1, which is the default MLD version, enable MLD globally. You can also optionally specify version 1 for clarity if you have both MLD versions in your configuration.

```
set protocols mld interface all
```

- d. For MLDv2, enable MLD globally with the version 2 option.

```
set protocols mld interface all version 2
```

9. Enable IGMP snooping for IGMPv2 or IGMPv3, or MLD snooping for MLDv1 or MLDv2, on all the configured OISM VLANs. Here in the common configuration for all OISM leaf devices, we include configuration only for the revenue bridge domains and the SBD. In the configuration steps specific to border leaf devices, we include the IGMP snooping or the MLD snooping configuration specific to the external multicast method you use on those devices.

When you enable IGMP snooping or MLD snooping, you also automatically enable advertising SMET Type 6 routes in the EVPN core based on received IGMP or MLD reports. If you use MAC-VRF EVPN instances, you enable IGMP snooping or MLD snooping in the MAC-VRF instances.

- a. For IGMPv2, enable *igmp-snooping*.



NOTE: You can use individual `igmp-snooping` commands for each VLAN, or one command with the `vlan all` option.

With the default switch instance:

```
set protocols igmp-snooping vlan VLAN-100
set protocols igmp-snooping vlan VLAN-200
set protocols igmp-snooping vlan VLAN-300
```

With a `vlan-aware` MAC-VRF instance `MAC-VRF1`:

```
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-100 set routing-
instances MAC-VRF1 protocols igmp-snooping vlan VLAN-200
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-300
```

- b. For IGMPv3, enable *igmp-snooping* for IGMPv3 source-specific multicast mode (SSM) reports. Include the `evpn-ssm-reports-only` option for all the configured VLANs.

With the default switch instance:

```
set protocols igmp-snooping vlan VLAN-100 evpn-ssm-reports-only
set protocols igmp-snooping vlan VLAN-200 evpn-ssm-reports-only
set protocols igmp-snooping vlan VLAN-300 evpn-ssm-reports-only
```

With a `vlan-aware` MAC-VRF instance `MAC-VRF1`:

```
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-100 evpn-ssm-reports-
only
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-200 evpn-ssm-reports-
only
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-300 evpn-ssm-reports-
only
```

- c. For MLDv1, enable *mld-snooping*. You can use individual `mld-snooping` commands for each VLAN, or one command with the `vlan all` option. We support MLD snooping with OISM only in

configurations with MAC-VRF EVPN instances. Here we show configuration with VLAN-aware MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 protocols mld-snooping vlan VLAN-100
set routing-instances MAC-VRF1 protocols mld-snooping vlan VLAN-200
set routing-instances MAC-VRF1 protocols mld-snooping vlan VLAN-300
```

- d. For MLDv2, enable *mld-snooping* for MLDv2 source-specific multicast mode (SSM) reports. Include the *evpn-ssm-reports-only* option for all the configured VLANs. Like in the previous step for MLDv1, here for MLDv2 we show configuration with VLAN-aware MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 protocols mld-snooping vlan VLAN-100 evpn-ssm-reports-only
set routing-instances MAC-VRF1 protocols mld-snooping vlan VLAN-200 evpn-ssm-reports-only
set routing-instances MAC-VRF1 protocols mld-snooping vlan VLAN-300 evpn-ssm-reports-only
```

10. Configure an L3 VRF instance (*instance-type vrf*) that you associate with OISM routing functions. Include the revenue bridge domain IRB interfaces and the SBD IRB interface in the routing instance. For example:

```
set routing-instances L3VRF-1 interface irb.100
set routing-instances L3VRF-1 interface irb.200
set routing-instances L3VRF-1 interface irb.300
set routing-instances L3VRF-1 interface lo0.1
set routing-instances L3VRF-1 route-distinguisher 10.255.0.1:100
set routing-instances L3VRF-1 vrf-target target:100:100
set routing-instances L3VRF-1 instance-type vrf
```

11. In the L3 VRF instance, specify the IRB interface that you configured as the OISM SBD. For example:

```
set routing-instances L3VRF-1 protocols evpn oism supplemental-bridge-domain-irb irb.300
```

12. (Required on the QFX10000 line of switches and PTX10001-36MR, PTX10004, PTX10008, and PTX10016 routers in any OISM role in releases prior to Junos OS and Junos OS Evolved 23.4R1; required with regular OISM on the QFX10000 line of switches, QFX5130-32CD switches, and QFX5700 switches when you configure them in the AR replicator role; optional but not recommended in any use cases other than the required ones) To avoid traffic loss at the onset of multicast flows, enable the *install-star-g-routes* option at the [edit <routing-instances *instance-name*> multicast-snooping-options oism] hierarchy level on all OISM devices.

If you use the default switch EVPN instance, `install-star-g-routes` is a global option. If you use MAC-VRF EVPN instances, you set this option in each EVPN MAC-VRF routing instance. With this option, upon receiving an EVPN Type 6 route on the SBD, the device immediately installs corresponding (*,G) routes on the PFE for the revenue bridge domain VLANs in the L3 VRF instance.

For full details on OISM device requirements, recommendations, and behavior with or without this option, see ["Latency and Scaling Trade-Offs for Installing Multicast Routes with OISM \(install-star-g-routes Option\)"](#) on page 1122.

With the default switch instance:

```
set multicast-snooping-options oism install-star-g-routes
```

With a `vlan-aware` or `vlan-based` MAC-VRF instance called `MAC-VRF1`:

```
set routing-instances MAC-VRF1 multicast-snooping-options oism install-star-g-routes
```

You might choose to not enable this option if space on the PFE is at a premium and you don't have stringent latency requirements.

13. (Required on QFX5130-32CD and QFX5700 switches starting in Junos OS Evolved Releases 22.4R2 and 23.1R1, and ACX Series routers starting in Junos OS Evolved Release 23.4R1, when you configure those switches as OISM server leaf or border leaf devices) Set the `conserve-mcast-routes-in-pfe` option at the `[edit routing-instances name multicast-snooping-options oism]` hierarchy level in the MAC-VRF EVPN instances on the device.

For example, with a MAC-VRF instance called `MAC-VRF1`:

```
set routing-instances MAC-VRF1 multicast-snooping-options oism conserve-mcast-routes-in-pfe
```



NOTE: You don't need to set this option on the device if you configured it as a standalone AR replicator with OISM.

See *oism (Multicast Snooping Options)* and ["ACX Series Routers, QFX5130-32CD Switches, and QFX5700 Switches as Server Leaf and Border Leaf Devices with OISM"](#) on page 1125 for details on this option.

SEE ALSO

*igmp-snooping**mld-snooping**oism**oism (Multicast Snooping Options)***Configure Server Leaf Device OISM Elements**

First configure the OISM elements described in ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices "](#) on page 1158 in an EVPN-VXLAN fabric.

Then follow these steps to configure the additional required OISM elements on the server leaf devices. The same EVPN-VXLAN fabric base and sample OISM environment apply to the additional server leaf configuration steps here.



NOTE: Here we describe PIM and OSPF protocol elements you configure on server leaf devices for specific tenant L3 VRF instances in the EVPN network. You can configure these elements at the global level ([edit protocols ...]) instead of at the routing instance level ([edit routing-instances VRF-name protocol...]) for either of these reasons:

- You want the same settings to apply to all of the VRF instances on the device.
- You want the settings to apply to the default L3 routing instance if you're running OISM with the default routing instance (on supported devices).

See ["OISM Support with Tenant L3 VRF Instances or the Default L3 Routing Instance"](#) on page 1052 for more on OISM configurations with the default L3 routing instance.

You configure the elements specific to the server leaf functions (like PIM) in the tenant L3 VRF instances.

See ["Configuration Elements for OISM Devices "](#) on page 1072 for more information about why OISM server leaf devices require these settings.

1. Configure PIM passive mode on server leaf devices. For example:

```
set routing-instances L3VRF-1 protocols pim passive
set routing-instances L3VRF-1 protocols pim interface all
```

2. Enable the server leaf device to accept multicast traffic from the SBD IRB interface as the source interface using the *accept-remote-source* statement at the [edit routing-instances *name* protocols pim

interface *irb-interface-name*] hierarchy level. For example, for our sample SBD, VLAN-300, the IRB interface is `irb.300`:

```
set routing-instances L3VRF-1 protocols pim interface irb.300 accept-remote-source
```

3. Configure an OSPF area in the L3 VRF instance. The server leaf device creates the PIM (S,G) entries it needs to forward the traffic between the SBD and the revenue bridge domains.

With regular OISM, you configure all interfaces in the VRF instance in OSPF passive mode. In passive mode, the server leaf devices can advertise and receive routes but don't form OPSF adjacencies or process OSPF protocol messages. For example:

```
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface all passive
```

With enhanced OISM, include the SBD IRB interface in the OSPF area in OSPF active mode so the OISM leaf devices form adjacencies on the SBD to route east-west traffic internally on the SBD. However, we only want the border leaf devices to assume the DR role on the SBD, because those devices also handle shuttling multicast traffic on the SBD for external sources and receivers. So as a result, you set the OSPF priority for the SBD IRB interface to 0. With this setting, the server leaf devices aren't considered in the OSPF designated router or backup designated router election process for the SBD. Finally, with enhanced OISM, set all other interfaces in the L3 VRF instance to OSPF passive mode. For example:

```
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface irb.300 priority 0
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface all passive
```

Configure Border Leaf Device OISM Elements with M-VLAN IRB Method (Symmetric Bridge Domains Model Only)

This section covers how to configure border leaf devices that use the OISM M-VLAN IRB method to exchange multicast data with external sources and receivers. See ["Supported Methods for Multicast Data Transfer to or from an External PIM Domain" on page 1064](#) for more on the available external multicast methods.



NOTE: We support the M-VLAN IRB external multicast method only with regular OISM and only on some platforms. See [Table 58 on page 1064](#) for more on where we support this method.

First configure the OISM elements described in ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices" on page 1158](#) in an EVPN-VXLAN fabric.

Then follow these steps to configure the additional required OISM elements on the border leaf devices. The same EVPN-VXLAN fabric base and the sample OISM environment in that section apply to the additional border leaf configuration steps here.

You configure different elements that are specific to the border leaf functions either globally, in the EVPN instances, or in the tenant L3 VRF instances.

See "[Configuration Elements for OISM Devices](#) " on [page 1072](#) for more information about why OISM border leaf devices require these settings.



NOTE: If you use the default L3 routing instance with OISM instead of VRF instances, substitute the corresponding configuration statements at the global hierarchy level instead of in a named routing instance. For example, substitute the configuration statements at this hierarchy level:

```
[edit routing-instances L3VRF-1 protocols evpn ...]
```

with the same statements at this global hierarchy level:

```
[edit protocols evpn ...]
```

See "[OISM Support with Tenant L3 VRF Instances or the Default L3 Routing Instance](#)" on [page 1052](#) for more on OISM configurations with the default L3 routing instance.

1. Configure the M-VLAN similarly to how you configure the revenue bridge domains and the SBD.
 - a. Configure the M-VLAN with an IRB interface and a VXLAN network identifier (VNI) mapping. For example, if the M-VLAN is VLAN-900:

With the default switch instance:

```
set vlans VLAN-900 vlan-id 900
set vlans VLAN-900 l3-interface irb.900
set vlans VLAN-900 vxlan vni 900
```

With a vlan-aware MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 vlans VLAN-900 vlan-id 900
set routing-instances MAC-VRF1 vlans VLAN-900 l3-interface irb.900
set routing-instances MAC-VRF1 vlans VLAN-900 vxlan vni 900
```

- b. Configure the M-VLAN IRB interface IP address. For example:

```
set interfaces irb unit 900 family inet address 172.16.90.1/24
```

- c. Extend the M-VLAN in the EVPN-VXLAN overlay. For example:

With the default switch instance:

```
set protocols evpn extended-vni-list 900
```

With a vlan-aware MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 protocols evpn extended-vni-list 900
```

2. Enable IGMP snooping for the M-VLAN, and configure the *multicast-router-interface* option on the L2 port that connects to the external multicast PIM router. For example, if xe-0/0/9.0 is the L2 interface that connects the EVPN fabric to the external multicast router on the M-VLAN:

- a. With IGMPv2:

In the default switch instance:

```
set protocols igmp-snooping vlan VLAN-900 set protocols igmp-snooping vlan VLAN-900
interface xe-0/0/9.0 multicast-router-interface
```

In a vlan-aware MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-900
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-900 interface xe-0/0/9.0
multicast-router-interface
```

- b. With IGMPv3, include the *evpn-ssm-reports-only* option:

In the default switch instance:

```
set protocols igmp-snooping vlan VLAN-900 evpn-ssm-reports-only
set protocols igmp-snooping vlan VLAN-900 interface xe-0/0/9.0 multicast-router-interface
```

In a vlan-aware MAC-VRF instance MAC-VRF1:

```
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-900 evpn-ssm-reports-only
set routing-instances MAC-VRF1 protocols igmp-snooping vlan VLAN-900 interface xe-0/0/9.0
multicast-router-interface
```

3. In the common leaf device configuration steps, you configure an L3 VRF instance associated with OISM routing functions. On the border leaf devices, also include the M-VLAN IRB interface in that L3 VRF. The following configuration block shows the common L3 VRF instance configuration with the additional M-VLAN IRB interface configuration statement highlighted:

```
set routing-instances L3VRF-1 interface irb.100
set routing-instances L3VRF-1 interface irb.200
set routing-instances L3VRF-1 interface irb.300
set routing-instances L3VRF-1 interface irb.900
set routing-instances L3VRF-1 interface lo0.1
set routing-instances L3VRF-1 route-distinguisher 10.255.0.1:100
set routing-instances L3VRF-1 vrf-target target:100:100
set routing-instances L3VRF-1 instance-type vrf
```

4. In the L3 VRF instance, set the OISM PEG role on the M-VLAN IRB interface. For example:

```
set routing-instances L3VRF-1 protocols evpn oism pim-evpn-gateway external-irb irb.900
```

5. Configure PIM in the L3 VRF instance for a border leaf device.
 - a. In the L3 VRF instance, configure PIM in distributed DR mode on the revenue bridge domain IRB interfaces using the *distributed-dr* option at the [edit routing-instances *name* protocols pim interface *irb-interface-name*] hierarchy level.

Configure PIM in standard mode on the SBD IRB interface.

Configure PIM on the M-VLAN IRB interface in either standard mode or distributed DR mode. A border leaf device works well in standard PIM mode if the external PIM router is single-homed to one border leaf device. However, we strongly recommend to use distributed DR mode in any case, but especially if the external PIM router is multihomed to multiple border leaf devices. Distributed DR mode also helps the device to efficiently do local routing on the M-VLAN to local receivers on border leaf devices. As a result, in the sample configuration here, we show setting PIM with the *distributed-dr* option on the M-VLAN IRB.

You also configure a PIM static RP that corresponds to the external PIM RP router. In the sample use cases in this documentation, the external PIM router serves as the PIM RP.

For example, if the revenue bridge domains are VLAN-100 and VLAN-200, the SBD is VLAN-300, and the M-VLAN is VLAN-900:

```
set routing-instances L3VRF-1 protocols pim rp static address external-PIM-RP-IP-address
set routing-instances L3VRF-1 protocols pim interface irb.100 distributed-dr
set routing-instances L3VRF-1 protocols pim interface irb.200 distributed-dr
set routing-instances L3VRF-1 protocols pim interface irb.300
set routing-instances L3VRF-1 protocols pim interface irb.900 distributed-dr
set routing-instances L3VRF-1 protocols pim interface lo0.1
```

- b. In the L3 VRF instance, set the *accept-join-always-from* option at the [edit routing-instances *name* protocols pim interface *irb-interface-name*] hierarchy level on the M-VLAN IRB interface.

Configure a policy option along with this statement so that the device always installs PIM joins from the external PIM router. See ["Configuration Elements for OISM Devices" on page 1072](#) for more information about why OISM border leaf devices require this configuration.

This sample configuration block represents the external PIM router as an MX Series router. For the policy prefix list, include the IP address of the M-VLAN interface on the MX Series router that connects to the border leaf device. For example:

```
set routing-instances L3VRF-1 protocols pim interface irb.900 accept-join-always-from
ajaf-900
set policy-options prefix-list mx-900 192.168.1.9/32
set policy-options policy-statement ajaf-900 term term1 from prefix-list mx-900
set policy-options policy-statement ajaf-900 term term1 then accept
set policy-options policy-statement ajaf-900 then reject
```

6. Configure an OSPF area in the L3 VRF instance for external multicast peer interface connectivity.

The border leaf device uses OSPF to learn routes to multicast sources to forward traffic from external sources toward internal receivers, and from internal sources toward external receivers. The device needs these routes to create the PIM (S,G) entries to forward traffic on the revenue bridge domains, the SBD, and the external multicast interfaces.

On a border leaf device with the M-VLAN IRB method for external multicast, configure the OSPF area to include the device loopback interface, the SBD IRB interface, and the M-VLAN IRB interface. For example, with our sample SBD VLAN-300 and M-VLAN VLAN-900:

```
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface lo0.1
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface irb.300
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface irb.900
```

Configure Border Leaf Device OISM Elements with Classic L3 Interface Method

This section covers how to configure border leaf devices that use the OISM classic L3 interface method to exchange multicast data with external sources and receivers. See ["Supported Methods for Multicast Data Transfer to or from an External PIM Domain" on page 1064](#) for more on the available external multicast methods.

First configure the OISM elements described in ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices " on page 1158](#) in an EVPN-VXLAN fabric.

Then follow these steps to configure the additional required OISM elements on border leaf devices. The same EVPN-VXLAN fabric base and the sample OISM environment in that section apply to the additional border leaf configuration steps here.

You configure most of the elements that are specific to the border leaf functions at L3 in the tenant L3 VRF instances.

See ["Configuration Elements for OISM Devices " on page 1072](#) for more information about why OISM border leaf devices require these settings.



NOTE: If you use the default L3 routing instance with OISM instead of VRF instances, substitute the corresponding configuration statements at the global hierarchy level instead of in a named routing instance. For example, substitute the configuration statements at this hierarchy level:

```
[edit routing-instances L3VRF-1 protocols evpn ...]
```

with the same statements at this global hierarchy level:

```
[edit protocols evpn ...]
```

See ["OISM Support with Tenant L3 VRF Instances or the Default L3 Routing Instance" on page 1052](#) for more on OISM configurations with the default L3 routing instance.

1. Configure a physical L3 interface with an IP address for external multicast. For example, for an L3 interface xe-0/0/6 with IP address 172.16.10.1/24:

```
set interfaces xe-0/0/6 unit 0 family inet address 172.16.10.1/24
```

2. Include the L3 logical interface in the L3 VRF instance that you configured in the common leaf device configuration steps. For example:

```
set routing-instances L3VRF-1 interface xe-0/0/6.0
```


3. In the L3 VRF instance, set the OISM PEG role on the border leaf device. With the classic L3 interface method, you don't need to specify an external IRB interface for external multicast:

```
set routing-instances L3VRF-1 protocols evpn oism pim-evpn-gateway
```

4. Configure PIM in the L3 VRF instance for a border leaf device.

For the revenue bridge domain IRB interfaces, configure PIM in distributed DR mode using the *distributed-dr* option at the [edit routing-instances *name* protocols pim interface *irb-interface-name*] hierarchy level.

Configure PIM in standard mode on the SBD IRB interface and the external multicast L3 logical interface.

We also configure a PIM static RP that corresponds to the external PIM RP router. In the sample use cases in this documentation, the external PIM router serves as the PIM RP.

For example, if the revenue bridge domains are VLAN-100 and VLAN-200, the SBD is VLAN-300, and the L3 interface is xe-0/0/6:

```
set routing-instances L3VRF-1 protocols pim rp static address external-PIM-RP-IP-address
set routing-instances L3VRF-1 protocols pim interface irb.100 distributed-dr
set routing-instances L3VRF-1 protocols pim interface irb.200 distributed-dr
set routing-instances L3VRF-1 protocols pim interface irb.300
set routing-instances L3VRF-1 protocols pim interface xe-0/0/6.0
set routing-instances L3VRF-1 protocols pim interface lo0.1
```

5. Configure an OSPF area in the L3 VRF instance that includes the SBD IRB interface and the external L3 multicast interface. This step is the same with either regular OISM or enhanced OISM.

The border leaf device uses OSPF to learn routes to multicast sources to forward traffic from external sources toward internal receivers, and from internal sources toward external receivers. With enhanced OISM, the border leaf devices also use the SBD to route multicast traffic from internal sources to receivers on revenue VLANs inside the EVPN network. The device needs these routes to create the PIM (S,G) entries to forward traffic on the revenue bridge domains, the SBD, and the external multicast interfaces.

On a border leaf device with the classic L3 interface method for external multicast, configure the OSPF area to include the SBD IRB interface and the external multicast logical L3 interface, both in OSPF active mode. Configure all other interfaces in the L3 VRF instance in passive mode so the

devices can share internal routes for those interfaces without forming OSPF adjacencies. For example, with our sample SBD IRB interface `irb.300` and L3 interface `xe-0/0/6`:

```
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface irb.300
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface xe-0/0/6.0
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface all passive
```

Configure Border Leaf Device OISM Elements with Non-EVPN IRB Method

This section covers how to configure border leaf devices that use the OISM non-EVPN IRB method to exchange multicast data with external sources and receivers. See ["Supported Methods for Multicast Data Transfer to or from an External PIM Domain" on page 1064](#) for more on the available external multicast methods.

First configure the OISM elements described in ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices " on page 1158](#) in an EVPN-VXLAN fabric.

Then follow these steps to configure the additional required OISM elements on border leaf devices. The same EVPN-VXLAN fabric base and the sample OISM environment in that section apply to the additional border leaf configuration steps here.

You configure most of the elements that are specific to the border leaf functions (like PIM) in the tenant L3 VRF instances. With this method, you don't extend the extra VLAN in the EVPN instance, so you don't configure related elements in the EVPN instance. The external multicast configuration elements are the same whether you use the default switch instance or MAC-VRF EVPN instances.



NOTE: If you use the default L3 routing instance with OISM instead of VRF instances, substitute the corresponding configuration statements at the global hierarchy level instead of in a named routing instance. For example, instead of configuring the elements in this section in an L3 VRF instance at this hierarchy level:

```
[edit routing-instances L3VRF-1 protocols evpn ...]
```

use the same statements at the global level:

```
[edit protocols evpn ...]
```

See ["OISM Support with Tenant L3 VRF Instances or the Default L3 Routing Instance" on page 1052](#) for more on OISM configurations with the default L3 routing instance.

See ["Configuration Elements for OISM Devices " on page 1072](#) for more information about why OISM border leaf devices require these settings.

1. Configure the extra VLAN with an IRB interface for external multicast. For example:

```
set vlans VLAN-900 vlan-id 900
set vlans VLAN-900 l3-interface irb.900
set interfaces irb unit 900 family inet address 172.16.90.1/24
```

2. Enable IGMP snooping or MLD snooping for the extra VLAN, and configure the *multicast-router-interface* option on the port that connects to the external multicast PIM router. For example, if xe-0/0/9.0 is the interface on the border leaf device that connects to the external multicast router on the extra VLAN:

- a. With IGMPv2:

```
set protocols igmp-snooping vlan VLAN-900
set protocols igmp-snooping vlan VLAN-900 interface xe-0/0/9.0 multicast-router-interface
```

- b. With IGMPv3, you don't need the `evpn-ssm-reports only` option here because you don't extend the extra VLAN in the EVPN instance:

```
set protocols igmp-snooping vlan VLAN-900
set protocols igmp-snooping vlan VLAN-900 interface xe-0/0/9.0 multicast-router-interface
```

- c. With MLDv1:

```
set protocols mld-snooping vlan VLAN-900
set protocols mld-snooping vlan VLAN-900 interface xe-0/0/9.0 multicast-router-interface
```

- d. With MLDv2, you don't need the `evpn-ssm-reports only` option here because you don't extend the extra VLAN in the EVPN instance:

```
set protocols mld-snooping vlan VLAN-900
set protocols mld-snooping vlan VLAN-900 interface xe-0/0/9.0 multicast-router-interface
```

3. Include the extra VLAN IRB interface in the L3 VRF instance that you configured in the common leaf device configuration steps.

The following configuration block shows the common L3 VRF configuration and highlights the additional statement:

```
set routing-instances L3VRF-1 interface irb.100
set routing-instances L3VRF-1 interface irb.200
set routing-instances L3VRF-1 interface irb.300
set routing-instances L3VRF-1 interface irb.900
set routing-instances L3VRF-1 interface lo0.1
set routing-instances L3VRF-1 route-distinguisher 10.255.0.1:100
set routing-instances L3VRF-1 vrf-target target:100:100
set routing-instances L3VRF-1 instance-type vrf
```

4. In the L3 VRF instance, set the OISM PEG role on the border leaf device. For example:

```
set routing-instances L3VRF-1 protocols evpn oism pim-evpn-gateway
```

5. Configure PIM in the L3 VRF instance for a border leaf device.

For the revenue bridge domain IRB interfaces, configure PIM in distributed DR mode using the *distributed-dr* option at the [edit routing-instances *name* protocols pim interface *irb-interface-name*] hierarchy level.

Configure PIM in standard mode on the SBD IRB interface and the extra VLAN IRB interface.

We also configure a PIM static RP that corresponds to the external PIM RP router. In the sample use cases in this documentation, the external PIM router serves as the PIM RP.

For example, if the revenue bridge domains are VLAN-100 and VLAN-200, the SBD is VLAN-300, and the extra VLAN is VLAN-900:

```
set routing-instances L3VRF-1 protocols pim rp static address external-PIM-RP-IP-address
set routing-instances L3VRF-1 protocols pim interface irb.100 distributed-dr
set routing-instances L3VRF-1 protocols pim interface irb.200 distributed-dr
set routing-instances L3VRF-1 protocols pim interface irb.300
set routing-instances L3VRF-1 protocols pim interface irb.900
set routing-instances L3VRF-1 protocols pim interface lo0.1
```

6. Configure an OSPF area in the L3 VRF instance for external multicast peer interface connectivity. This step is the same with either regular OISM or enhanced OISM.

The border leaf device uses OSPF to learn routes to multicast sources to forward traffic from external sources toward internal receivers, and from internal sources toward external receivers. The device needs these routes to create the PIM (S,G) entries to forward traffic on the revenue bridge domains, the SBD, and the external multicast interfaces.

On a border leaf device with the non-EVPN IRB method for external multicast, configure the OSPF area to include the SBD IRB interface and the non-EVPN IRB interface, both in OSPF active mode. Configure all other interfaces in the L3 VRF instance in passive mode so the devices can share internal routes for those interfaces without forming OSPF adjacencies. For example, with our sample SBD IRB interface (irb.300) and extra non-EVPN IRB interface (irb.900);,

```
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface irb.300
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface irb.900
set routing-instances L3VRF-1 protocols ospf area 0.0.0.0 interface all passive
```

CLI Commands to Verify the OISM Configuration

To verify your OISM configuration:

1. Use the `show evpn oism` command to view the SBD IRB interface in each L3 VRF instance that you configured on the device for OISM. For example:

```
user@Leaf-1> show evpn oism
L3 context          SBD
L3VRF-1             irb.300
L3VRF-2             irb.400
```

To view information for a specific routing instance or to see more details about the configuration, use the `extensive` option with this `show` command. This command shows the OISM mode you have configured—Regular (the original symmetric bridge domains model) or Enhanced (the enhanced asymmetric bridge domains model). You can also display information only for a specified L3 VRF. For example:

```
user@Leaf-1> show evpn oism l3-context L3VRF-1 extensive
EVPN L3 context: L3VRF-1
  OISM SBD interface: irb.300
  OISM Mode: Enhanced OISM
```

2. Enter `show evpn oism spmsi-ad extensive` to see multicast (S,G) information corresponding to EVPN Type 10 S-PMSI A-D routes. The OISM leaf devices use the S-PMSI A-D routes to perform PIM source registration only for multicast sources inside the EVPN-VXLAN network. For example:

```
user@Leaf-1> show evpn oism spmsi-ad extensive
Instance evpn-vxlan-A
  VN Identifier: 903
```

Group	Source	Local	Remote
233.252.0.4	10.0.1.3	1	0

3. Use the `show route table bgp.evpn.0 ... extensive` command to see the OISM capabilities you have enabled on an OISM leaf device. These capabilities are in the EVPN multicast flags extended community in EVPN Type 3 IMET routes. This extended community is displayed in the `Communities: ... evpn-mcast-flags` output field as a hexadecimal flags value with the keyword for each enabled function. The OISM-related flags include:

- `igmp-snooping-enabled`—You enabled IGMP snooping. The `evpn-mcast-flags` bit for IGMP snooping without OISM or PEG configuration is 0x01.
- `mld-snooping-enabled`—You enabled MLD snooping. The `evpn-mcast-flags` bit for MLD snooping without OISM or PEG configuration is 0x02.
- `oism`—You globally enabled OISM. The `evpn-mcast-flags` bit for OISM is 0x08. You might see flags values such as the following (without PEG configuration on the device):
 - 0x9—OISM and IGMP snooping
 - 0xa—OISM and MLD snooping
 - 0xb—OISM with both IGMP snooping and MLD snooping
- `peg`—You configured PEG mode on the associated interface (for border leaf devices that connect to an external PIM domain). The `evpn-mcast-flags` bit for PEG mode is 0x10, so with PEG mode enabled, you might see flag values such as the following:
 - 0x19—PEG mode with OISM and IGMP snooping
 - 0x1a—PEG mode with OISM and MLD snooping
 - 0x1b—PEG mode with OISM, IGMP snooping, and MLD snooping
- `sbd`—The advertised EVPN Type 3 route is for an interface associated with the SBD. We set this bit for interoperability with other vendors and to comply with the IETF draft standard for OISM, [draft-ietf-bess-evpn-irb-mcast](#). The SBD `evpn-mcast-flags` bit is 0x100, so in EVPN routes for the SBD, you might see flag values such as the following:
 - 0x109—OISM with IGMP snooping for the SBD on a server leaf device
 - 0x119—PEG mode with OISM and IGMP snooping for the SBD on a border leaf device

Here are a few examples of the communities display.

For a route advertised for an M-VLAN IRB interface on a border leaf device that has the `peg` flag set:

```
user@BL-1> show route table bgp.evpn.0 match-prefix 3:<...> extensive
...
      Communities: target:1:1 encapsulation:vxlan(0x8) evpn-mcast-flags:0x19:igmp-
snooping-enabled:oism:peg
...
```

You don't see the `peg` flag set for a revenue bridge domain interface on a border leaf device or a server leaf device:

```
user@Leaf-1> show route table bgp.evpn.0 match-prefix 3:<...> extensive
...
      Communities: target:1:1 encapsulation:vxlan(0x8) evpn-mcast-flags:0x9:igmp-
snooping-enabled:oism
...
```

With IGMP snooping and MLD snooping enabled on an OISM server leaf device, you might see:

```
user@Leaf-1>show route table bgp.evpn.0 match-prefix 3:<...> extensive
...
      Communities: target:1:1 encapsulation:vxlan(0x8) evpn-mcast-
flags:0xb:igmp-snooping-enabled:ml-d-snooping-enabled:oism
...
```

The advertised route for the SBD IRB interface with IGMP snooping enabled on an OISM server leaf device will display:

```
user@Leaf-1>show route table bgp.evpn.0 match-prefix 3:<...> extensive
...
      Communities: target:1:1 encapsulation:vxlan(0x8) evpn-mcast-
flags:0x109:igmp-snooping-enabled:oism:sbd
...
```

4. Enter `show igmp snooping evpn status <vlan name> <detail>` to see the EVPN-VXLAN L2 multicast context for the OISM bridge domains (VLANs) on the device. The default output includes the VXLAN VNI mappings of the VLANs. For example:

```
user@Leaf-1> show igmp snooping evpn status
Instance: default-switch
  Bridge-Domain: VLAN-100, VN Identifier: 100
    OISM: Enabled
  Bridge-Domain: VLAN-200, VN Identifier: 200
    OISM: Enabled
  Bridge-Domain: VLAN-300, VN Identifier: 300
    OISM: Enabled
```

If you have MLD snooping enabled, use the `show mld snooping evpn status <vlan name><detail>` command, which shows the same output as the IGMP snooping version of the command.

The detail option also displays whether a VLAN is the OISM SBD (Supplementary BD) or the M-VLAN (External VLAN). Both of those output fields display No for revenue bridge domains (VLANs) or if you don't enable OISM.

Here is an example on a server leaf device in a fabric where the SBD is VLAN-300:

```
user@Leaf-1> show igmp snooping evpn status vlan VLAN-300 detail
Instance: default-switch
  Bridge-Domain: VLAN-300, VN Identifier: 300
    OISM           : Enabled
    Supplementary BD: Yes
    External VLAN   : No
```

Here is another example on a border leaf device in a fabric where the M-VLAN is VLAN-900:

```
user@Leaf-1> show igmp snooping evpn status vlan VLAN-900 detail
Instance: default-switch
  Bridge-Domain: VLAN-900, VN Identifier: 900
    OISM           : Enabled
    Supplementary BD: No
    External VLAN   : Yes
```

5. Enter `show evpn multicast-snooping status` to see if you enabled IGMP snooping or MLD snooping on VLANs you configured for OISM elements.

For example, in the following sample command, if the revenue bridge domains on a server leaf device are VLAN-100 and VLAN-200, the SBD is VLAN-300, and you enabled IGMP snooping (but not MLD snooping):

```
user@Leaf-1> show evpn multicast-snooping status
Instance: __default_evpn__

Instance: default-switch
  VLAN ID: 100
    Multicast Address Family: INET
      SG Sync: Enabled
    Multicast Address Family: INET6
      SG Sync: Disabled

  VLAN ID: 200
    Multicast Address Family: INET
      SG Sync: Enabled
    Multicast Address Family: INET6
      SG Sync: Disabled

  VLAN ID: 300
    Multicast Address Family: INET
      SG Sync: Enabled
    Multicast Address Family: INET6
      SG Sync: Disabled
```

With both IGMP snooping and MLD snooping enabled, you'll also see Multicast Address Family: INET6 with SG Sync: Enabled.

6. Use EVPN commands, multicast snooping commands, and PIM commands to see details for EVPN multihoming ESIs, learned multicast group routes, multicast traffic flow, and AR replication and forwarding. These commands are not specific to OISM operation, but are helpful to verify OISM operation and troubleshoot OISM issues.

For example:

- show igmp snooping membership vlan *vlan-name*<virtual-switch *EVPN-instance-name*>
- show mld snooping membership vlan *vlan-name*<virtual-switch *EVPN-instance-name*>
- show evpn instance <*EVPN-instance-name*> designated-forwarder <esi *esi-number*>
- show pim join summary <instance *VRF-instance-name*>
- show pim join detail <instance *VRF-instance-name*>

- `show evpn multicast-snooping assisted-replication next-hops <instance EVPN-instance-name>`
- `show evpn multicast-snooping assisted-replication replicators`

For a full OISM configuration example of a common data center fabric use case with regular OISM, which also shows how to use these commands to verify OISM operation, see [Optimized Intersubnet Multicast \(OISM\) with Assisted Replication \(AR\) for Edge-Routed Bridging Overlays](#).

SEE ALSO

Multicast Support in EVPN-VXLAN Overlay Networks 925
Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment 934
Overview of Selective Multicast Forwarding 1005
Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric With a Virtual Gateway 866

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
25.4R1	Starting in Junos OS Evolved Release 25.2R1, on some platforms you can configure OISM with the default L3 routing instance instead of with L3 VRF instances.
25.2R1	Starting in Junos OS Evolved Release 25.2R1, we support DCI with enhanced OISM to efficiently transfer multicast traffic between seamlessly stitched EVPN-VXLAN data centers.
25.2R1	Starting in Junos OS Release 25.2R1, we support DCI with enhanced OISM between seamlessly stitched EVPN-VXLAN data centers that use IPv6 underlay peering.
25.2R1	Starting in Junos OS and Junos OS Evolved Release 25.2R1, we support setting an exception policy on enhanced OISM devices to enable forwarding multicast flows on the source VLAN instead of the SBD, which fixes a limitation for flows where the packets have TTL=1.
24.4R1	Starting in Junos OS Evolved Release 24.4R1, we support regular OISM with IGMPv2, IGMPv3, and IGMP snooping on PTX10002-36QDD routers. We also support regular and enhanced OISM with IGMPv2, IGMPv3, and IGMP snooping on ACX7024X, ACX7332, ACX7348, and ACX7509 routers.

24.2R2-S2	Starting in Junos OS Release 24.2R2-S2, when you configure the GBP unified forwarding table (UFT) profile <code>vxlan-gbp-mc-profile</code> at the <code>[edit chassis forwarding-options]</code> hierarchy level, we seamlessly support VXLAN group-based policy (GBP) unicast traffic flows together with multicast traffic flows using enhanced OISM.
24.2R1	Starting in Junos OS Release 24.2R1, you can statically identify an OISM leaf devices's multihoming peer devices (devices that share an Ethernet segment for a multihomed host) to help avoid multicast traffic loss when peer devices go up and down. This feature is available only with enhanced OISM.
24.2R1	Starting in Junos OS Releases 24.2R1 and 23.4R2, on some platforms we support enhanced OISM for IPv4 and IPv6 multicast data traffic in an EVPN-VXLAN ERB overlay network that has an IPv6 underlay. With this release, you can configure any of the supported platforms as enhanced OISM server leaf devices, and only EX4650 and QFX5120 switches as enhanced OISM border leaf devices.
24.2R1	Starting in Junos OS Release 24.2R1, we support DCI with enhanced OISM to efficiently transfer multicast traffic between EVPN-VXLAN seamlessly stitched data centers. You can configure the DCI gateway devices in each data center as enhanced OISM server leaf devices or border leaf devices. The DCI gateway devices and OISM leaf devices use SMET (EVPN Type 6) routes to seamlessly send multicast traffic only to the remote receivers across the DCI who subscribed to a multicast flow. The data centers can use IPv4 underlay peering to support either IPv4 or IPv6 multicast data traffic.
23.4R1EVO	Starting in Junos OS Evolved Release 23.4R1, ACX7024, ACX7100-32C, and ACX7100-48L routers support OISM with IGMPv2, IGMPv3, and IGMP snooping. These devices support OISM using MAC-VRF EVPN instances with <code>vlan-aware</code> and <code>vlan-based</code> service types. You can configure these devices in OISM server leaf, border leaf, or lean spine roles. In the border leaf role, these devices support only the classic L3 interface method to connect to an external multicast PIM domain.
23.4R1	Starting in Junos OS Release 23.4R1 and Junos OS Evolved Release 23.4R1, you can customize the DF election method on multihoming peer OISM PEG devices to use mod-based or preference-based PEG DF election. When you configure this feature, the configured DF election method replaces the default PIM-based DF election method.
23.4R1	Starting in Junos OS and Junos OS Evolved Release 23.4R1, we no longer require you to set the <code>install-star-g-routes</code> option on the QFX10000 line of switches or the PTX10000 line of routers when you configure those devices as OISM server leaf or border leaf devices.

23.4R1	Starting in Junos OS Release 23.4R1, we introduce support for OISM in enhanced OISM mode, which uses an asymmetric bridge domains configuration model. With enhanced OISM, you don't need to configure all revenue VLANs in the network on all OISM devices. On each device, you can configure only the revenue VLANs that the device hosts. We support enhanced OISM with IGMPv2, IGMPv3, and IGMP snooping, and on some platforms, also with MLDv1, MLDv2, and MLD snooping. With enhanced OISM, you can configure EX4100 and EX4400 switches in the server leaf role only, and other supported devices in the border leaf role or the server leaf role.
23.1R1EVO	Starting in Junos OS Evolved Release 23.1R1, QFX5130-32CD and QFX5700 switches support OISM and AR with MLDv1, MLDv2, and MLD snooping. You can configure MLD and MLD snooping on these devices when they act as OISM server leaf devices, OISM border leaf devices, or standalone AR replicator devices with OISM.
22.3R1-EVO	Starting in Junos OS Evolved Release 22.3R1, PTX10001-36MR, PTX10004, PTX10008, and PTX10016 routers support OISM with IGMPv2 or IGMPv3, IGMP snooping, and SMET route optimization. These devices support OISM using MAC-VRF EVPN instances with vlan-aware and vlan-based service types. You can configure any of these devices in OISM server leaf, border leaf, or lean spine roles. In the border leaf role, these devices support any of the available OISM methods to connect to an external multicast PIM domain: M-VLAN IRB method, classic L3 interface method, or non-EVPN IRB method.
22.3R1	Starting in Junos OS Release 22.3R1, EX4300-48MP and EX4400 switches support forwarding and routing multicast traffic using OISM. These devices support OISM with IGMPv2 in the default switch instance with the VLAN-aware service model. You can configure these devices only in the OISM server leaf role, and not as an OISM border leaf device.
22.2R1-EVO	Starting in Junos OS Evolved Release 22.2R1, you can enable AR with OISM in MAC-VRF EVPN instance configurations on QFX5130-32CD and QFX5700 switches. You can configure the AR leaf role or AR replicator role on these devices. The AR replicator role operates only in standalone mode (the AR replicator role can't be collocated with the OISM border leaf role on the same device). We support AR and OISM with IGMPv2 or IGMPv3, and IGMP snooping.
22.2R1-EVO	Starting in Junos OS Evolved Release 22.2R1, QFX5130-32CD and QFX5700 switches configured as AR replicators with OISM install multicast (*,G) states with IGMPv2 or multicast (S,G) states with IGMPv3 for EVPN Type 6 routes only on the SBD VLAN. On these devices, you only see multicast group routes on the SBD in show multicast snooping route command output.

22.2R1	Starting in Junos OS Release 22.2R1, we support OISM with IGMPv2 or IGMPv3 with the default switch instance or MAC-VRF EVPN instances (vlan-aware and vlan-based service types) on EX4650, QFX5110, QFX5120, QFX10002 (except QFX10002-60C), QFX10008, and QFX10016 switches. You can configure any of these devices in the OISM server leaf role. All of these devices except EX4650 and QFX5110 switches can be OISM border leaf devices. On QFX10000 Series border leaf devices, you can use either the OISM M-VLAN IRB interface method or the classic L3 interface method to connect to an external multicast PIM domain. On EX4650 and QFX5120 border leaf devices, you can use only the classic L3 interface method.
22.2R1	Starting in Junos OS Release 22.2R1, you can enable AR with OISM in the default switch instance or MAC-VRF EVPN instances on EX4650, QFX5110, QFX5120, QFX10002 (except QFX10002-60C), QFX10008, and QFX10016 switches. The AR replicator role can be collocated with the OISM border leaf role on the same device, or you can configure the AR replicator role in standalone mode on a lean spine device in the fabric. (Only switches in the QFX10000 line can be AR replicators.) We support AR and OISM with IGMPv2 or IGMPv3, and IGMP snooping.
22.1R1EVO	Starting in Junos OS Evolved Release 22.1R1, QFX5130-32CD and QFX5700 switches support OISM with IGMPv2 or IGMPv3 in MAC-VRF EVPN instances (vlan-aware and vlan-based service types). These devices can be OISM server leaf or border leaf devices. The border leaf devices support the classic L3 interface model or the non-EVPN IRB model to connect to an external multicast PIM domain.
21.2R1	Starting in Junos OS Release 21.2R1, QFX5110, QFX5120, and QFX10002 (except QFX10002-60C) switches support OISM with IGMPv2 in the default switch instance with the VLAN-aware service model. You can configure any of these devices in the OISM server leaf role, but only QFX10002 switches can be OISM border leaf devices. The border leaf devices support either the OISM M-VLAN IRB model or the classic L3 interface model to connect to an external multicast PIM domain.

EVPN-VXLAN DCI Multicast with Enhanced OISM

SUMMARY

Transfer IPv4 or IPv6 multicast traffic efficiently and seamlessly across EVPN-VXLAN to EVPN-VXLAN interconnected data centers using enhanced OISM.

IN THIS SECTION

- [Benefits of DCI Multicast with Enhanced OISM | 1189](#)
- [EVPN-VXLAN to EVPN-VXLAN DCI Stitching Overview | 1189](#)

- [OISM Overview | 1190](#)
- [How Enhanced OISM Works over DCI Stitching | 1192](#)
- [More Enhanced OISM over DCI Use Cases | 1196](#)
- [Multicast Next Hops Convergence Optimization | 1203](#)
- [Configure EVPN-VXLAN DCI with Enhanced OISM | 1204](#)
- [Sample Configuration for DCI with Enhanced OISM | 1206](#)
- [Show Commands to Verify DCI with Enhanced OISM | 1232](#)

We support efficient transfer of IPv4 or IPv6 multicast traffic with Ethernet VPN–Virtual Extensible LAN (EVPN-VXLAN) to EVPN-VXLAN seamless data center interconnect (DCI) stitching using enhanced optimized intersubnet multicast (OISM). The EVPN-VXLAN fabrics on both sides of the interconnection can use either IPv4 underlay peering or IPv6 underlay peering.

Without this feature, the DCI gateway (iGW) devices in each data center flood multicast traffic across the WAN. Flooding consumes significant WAN bandwidth if your network has many multicast flows or high multicast traffic rates. This feature seamlessly replaces the multicast flooding behavior in DCI stitched networks to optimize multicast forwarding across the DCI.

Benefits of DCI Multicast with Enhanced OISM

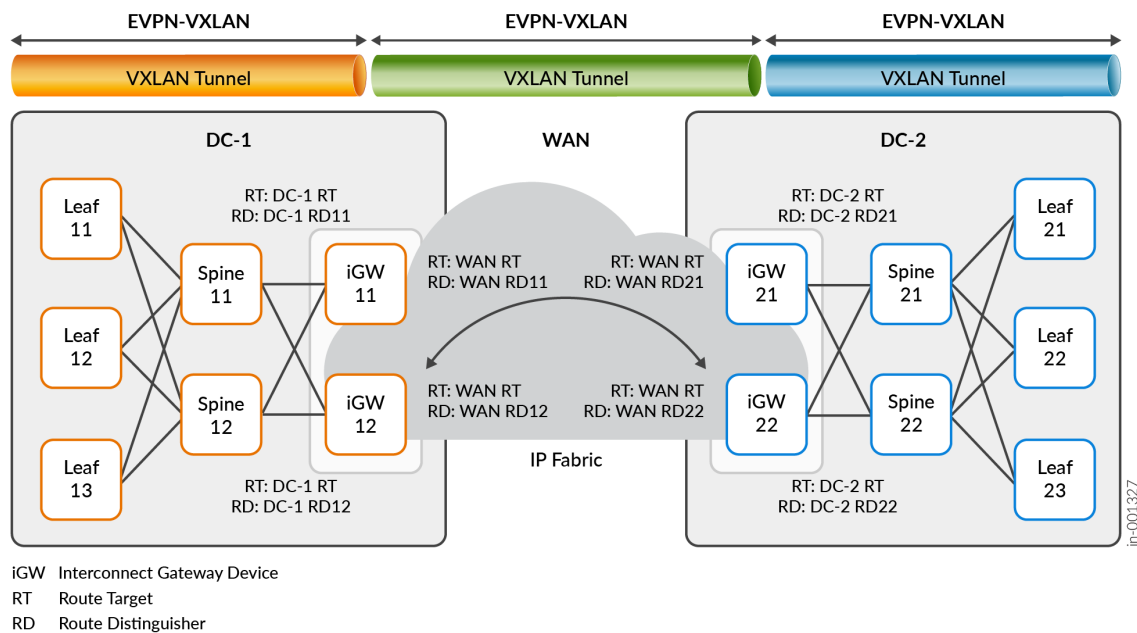
- Significantly reduces network bandwidth consumption when transferring multicast traffic across the WAN interconnecting data centers, especially if your network has a large number of multicast flows or high multicast traffic rates.
- Prevents traffic loss when the designated forwarder (DF) and non-DF (NDF) status changes among the multihomed iGW devices that share an interconnect Ethernet segment (I-ESI) in a data center.

EVPN-VXLAN to EVPN-VXLAN DCI Stitching Overview

DCI stitching seamlessly transfers control and data traffic in EVPN-VXLAN data centers across an interconnecting WAN. See [Figure 114 on page 1190](#). DCI gateway (iGW) devices in each data center use different route targets (RTs) for:

- Traffic inside the data center (DC-1 RT, DC-2 RT).
- Traffic exchanged across the WAN (WAN RT).

Figure 114: EVPN-VXLAN to EVPN-VXLAN DCI Stitching Design Overview



To configure iGW devices, you use the *interconnect* stanza at the [edit <routing-instances *routing-instance-name*> protocols evpn] hierarchy level. Also see the following for some DCI configuration examples:

- [Configure VXLAN Stitching for Layer 2 Data Center Interconnect](#)
- [EVPN Type 2 Symmetric Routing with DCI Stitching](#)

OISM Overview

IN THIS SECTION

- [OISM Configuration | 1191](#)
- [OISM Supplemental Bridge Domain \(SBD\) | 1192](#)

OISM handles multicast traffic efficiently in an EVPN-VXLAN data center for multicast sources and receivers inside and outside the data center.

OISM has two modes:

- **Regular OISM mode**—This mode uses a symmetric bridge domains OISM model that requires you to configure all revenue VLANs (the tenant VLANs) in the network on all OISM leaf devices. This is the original OISM implementation.
- **Enhanced OISM mode**—This mode doesn't require you to configure all revenue bridge domains (VLANs) in the network on all OISM devices. On each device, you can configure only the revenue VLANs hosted by that device (so we call this an asymmetric model). However, you must still configure the revenue VLANs symmetrically on OISM devices that are multihoming peers.



NOTE: We support this seamless optimized multicast over DCI feature with enhanced OISM only.

OISM operates on the server leaf and border leaf provider edge (PE) devices in EVPN-VXLAN edge-routed bridging (ERB) overlay data center architectures. If the topology includes a layer of transit spine devices, you don't configure OISM on those devices.

- **OISM server leaf devices** host the multicast sources and receivers within the data center. The sources and receivers can be either directly connected or connected through intermediary customer edge (CE) devices.
- **OISM border leaf devices** connect to external multicast domains to transfer multicast data into a data center from external sources and out of a data center to external receivers. Like server leaf devices, the border leaf devices might also host devices in the data center that are multicast sources or receivers.

See ["Optimized Intersubnet Multicast in EVPN Networks" on page 1049](#) for details on how OISM works in general. See these sections for more on the differences with enhanced OISM operation in particular:

- ["Overview of Enhanced OISM" on page 1057](#)
- ["How Enhanced OISM Works" on page 1112](#)

OISM Configuration

In regular or enhanced mode, OISM requires many configuration elements to operate. You configure some elements in common on server leaf devices and border leaf devices. Some elements are specific to OISM server leaf devices or border leaf devices. See the configuration sections in ["Optimized Intersubnet Multicast in EVPN Networks" on page 1049](#) for the steps to configure each type of OISM device. Many of the configuration steps are the same for either regular or enhanced OISM modes.

You can configure iGW devices to act as enhanced OISM server leaf or border leaf devices. If the iGW devices also provide the interfaces to external multicast sources or receivers, you configure them as OISM border leaf devices.

OISM Supplemental Bridge Domain (SBD)

One common configured element for OISM operation is the VLAN called the supplemental bridge domain (SBD). You configure a unique SBD VLAN ID and corresponding integrated routing and bridging (IRB) interface for each tenant virtual routing and forwarding (VRF) instance in a data center.

One major difference between regular OISM and enhanced OISM is that regular OISM ingress devices send multicast traffic from the source on the source VLAN to other OISM devices. In contrast, enhanced OISM ingress devices send source traffic on the source VLAN only to their multihoming peers, and use the SBD to send source traffic to all other OISM devices.

When you use enhanced OISM with DCI stitching, you assign the same VLAN as the SBD in both data centers for the VRF instances that span the interconnection. The iGW devices with enhanced OISM enabled transfer multicast control and data traffic across the interconnection only on the SBD.



NOTE: Because enhanced OISM uses the SBD toward remote devices that aren't its multihoming peers, the enhanced mode has an inherent limitation where packets with TTL=1 will not reach receivers on those remote devices.

If you use enhanced OISM with DCI stitching and need to avoid this limitation, we have a solution that enables enhanced OISM to behave like regular OISM to use the source VLAN for particular multicast data flows toward the remote OISM devices. You can enable this solution for particular multicast groups (or groups and sources) using routing policies and the *forward-on-source-bridge-domain* configuration option. See ["Enhanced OISM Exception Policy to Forward on Source VLAN Instead of SBD for Packets with TTL=1" on page 1141](#) for details.

How Enhanced OISM Works over DCI Stitching

IN THIS SECTION

- [SMET \(EVPN Type 6 Route\) Control Flow Across DCI | 1193](#)
- [Multicast Data Flow with SMET and Enhanced OISM Across DCI | 1195](#)

In a DCI stitching environment without enhanced OISM, the iGW devices default to flooding all multicast traffic across the interconnecting WAN. Flooding can consume significant WAN bandwidth when you have heavy multicast traffic loads.

This feature seamlessly replaces the multicast flooding behavior across the WAN with selective multicast Ethernet tag (SMET) forwarding between the data centers. SMET forwarding only sends

multicast traffic across the interconnection when the other data center has receivers that explicitly subscribed to that multicast flow. EVPN Type 6 routes implement SMET in EVPN data centers.

You enable enhanced OISM on the leaf devices in each data center as either OISM server leaf devices or OISM border leaf devices. The DCI iGW devices can be either type of OISM leaf device. Often, the iGW devices are the devices in the EVPN-VXLAN fabric that also act as the OISM border leaf devices to connect to external sources or receivers.

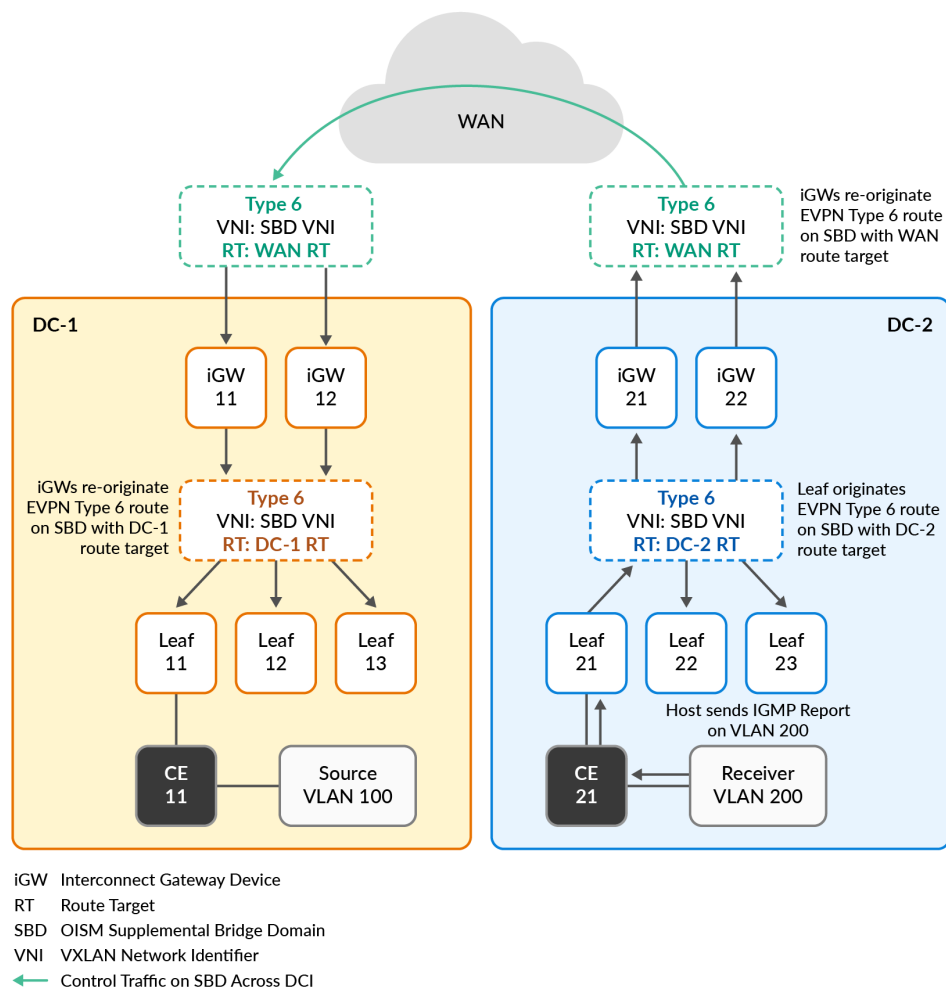


NOTE: In the figures in this section and the following sections showing DCI with enhanced OISM, the spine devices in the DCI architecture in [Figure 114 on page 1190](#) act only as L3 transit devices in each data center. You don't configure OISM on the spine devices. As a result, to more clearly show how the multicast control and data traffic flows among the OISM leaf devices, we don't include the spine devices in these figures.

SMET (EVPN Type 6 Route) Control Flow Across DCI

[Figure 115 on page 1194](#) shows the control flow for EVPN Type 6 routes in each data center and across the interconnection.

Figure 115: SMET EVPN Type 6 Routes Sent Across DCI for Subscribed Receivers



The SMET control flow operates as follows:

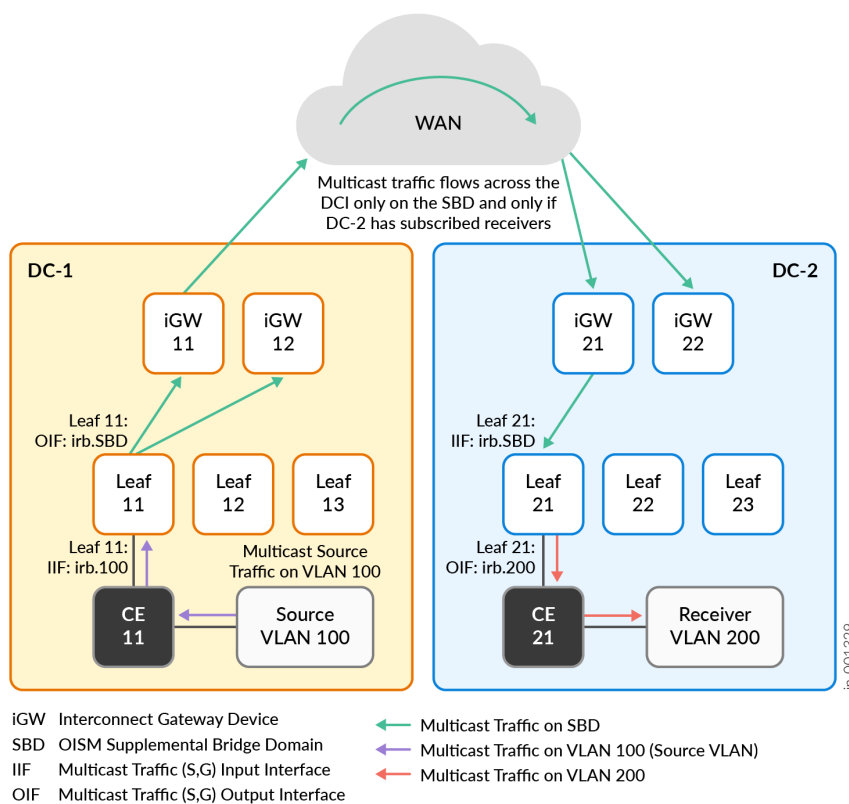
1. To subscribe to a multicast data flow, a receiver in DC-2, for example, sends an Internet Group Management Protocol (IGMP) report for IPv4 multicast flows, or sends a Multicast Listener Discovery (MLD) report for IPv6 multicast flows.
2. The OISM leaf device that receives the IGMP or MLD report in DC-2, Leaf 21 in this case, creates a SMET route (EVPN Type 6 route) for that subscription. The SMET route uses the RT for DC-2.
3. Leaf 21 advertises the route on the OISM SBD to all other OISM leaf devices in DC-2 (Leaf 22, Leaf 23, iGW 21, and iGW 22).
4. The iGW devices configured as OISM leaf devices in DC-2 (iGW 21, and iGW 22) receive the SMET route on the SBD, seamlessly re-originate the SMET route using the WAN RT, and forward the SMET route on the OISM SBD across the DCI.

5. The iGW devices on the other side of the DCI in DC-1 (iGW 11 and iGW 12) receive the SMET route with the WAN RT on the SBD. iGW 11 and iGW 12 seamlessly re-originate the SMET route using the RT for DC-1, and forward the SMET route to all other OISM leaf devices in DC-1.
6. The OISM leaf devices in both data centers store the multicast subscriber information from the SMET routes in their multicast routing and forwarding tables.

Multicast Data Flow with SMET and Enhanced OISM Across DCI

Figure 116 on page 1195 shows that based on received SMET routes, the OISM leaf devices in each data center send multicast traffic from the source only to the remote receivers across the DCI who subscribed to that multicast flow.

Figure 116: Multicast Data Flow with Enhanced OISM and DCI



In this case:

1. The multicast source is in DC-1 on VLAN 100.
2. A receiver in DC-2 on VLAN 200 subscribes to the multicast data from the source in DC-1.

3. As we show in "[SMET \(EVPN Type 6 Route\) Control Flow Across DCI](#)" on page 1193, the OISM leaf devices in both data centers, including iGW 11, iGW 12, and Leaf 11 in DC-1, receive SMET routes for that subscription.
4. In DC-1, Leaf 11 receives multicast traffic from the source on VLAN 100. Leaf 11 knows from receiving the SMET routes that there is a receiver in DC-2, so Leaf 11 routes the multicast traffic to iGW 11 and iGW 12 on the SBD.



NOTE: Although [Figure 116 on page 1195](#) doesn't show local receivers on the other leaf devices in DC-1, Leaf 11 would also route the multicast traffic on the SBD to those devices if (and only if) those devices also have subscribed receivers.

5. iGW 11 and iGW 12 also know from receiving the SMET routes that there is a receiver in DC-2. In this case, iGW 11 is the DF for the pair of iGW devices in DC-1. So iGW 11 forwards the traffic over the interconnection to DC-2 on the SBD.
6. The iGW devices in DC-2 receive the multicast traffic. In this case, iGW 21 is the DF for the pair of iGW devices in DC-2, so iGW 21 forwards the traffic on the SBD to Leaf 21 toward the subscribed receiver.
7. In DC-2, Leaf 21 routes the multicast traffic toward the receiver on the receiver VLAN, VLAN 200.

See "[Multicast Next Hops Convergence Optimization](#)" on page 1203 for more on how we improve EVPN multicast next hop convergence over the DCI when iGW device DF and NDF status changes.

More Enhanced OISM over DCI Use Cases

IN THIS SECTION

- [Multicast Traffic Flow across DCI from Multihomed Source on DCI Gateway Devices](#) | 1197
- [Multicast Traffic Flow from External Source to Internal Receivers](#) | 1199
- [Multicast Control and Data Flow for Internal Source to External Receiver](#) | 1201

We support a variety of configurations for DCI multicast with enhanced OISM.

For example, the **multicast source** can be:

- Behind an OISM server leaf device in one of the data centers.

"[Multicast Data Flow with SMET and Enhanced OISM Across DCI](#)" on page 1195 describes a use case with an internal source behind an OISM server leaf device in one data center to subscribed receivers behind OISM server leaf devices in another data center.

"[Multicast Traffic Flow across DCI from Multihomed Source on DCI Gateway Devices](#)" on page 1197 describes a use case with an internal source that is multihomed to the iGW devices acting as OISM server leaf devices in one data center. The receivers are behind OISM leaf devices in both data centers.

- In an external PIM domain connected to the iGW devices in one of the data centers.
 "[Multicast Traffic Flow from External Source to Internal Receivers](#)" on page 1199 describes a use case where the iGW devices also act as the OISM PIM EVPN gateway (PEG) devices. The receivers are inside both of the interconnected data centers.
- In an external PIM domain connected to an OISM leaf device or multihomed to more than one OISM leaf device in one of the data centers.

In this use case, you configure those leaf devices as the OISM border leaf PEG devices. We don't include a figure or describe this use case in detail here.

With a multicast source in one of the data centers, the **multicast receivers** can be:

- In the same data center as the multicast source, or in another data center across the DCI.
 "[Multicast Traffic Flow across DCI from Multihomed Source on DCI Gateway Devices](#)" on page 1197 describes a use case where the source is multihomed to the iGW devices in one data center and the receivers are behind OISM leaf devices in both data centers.
- In an external PIM domain where the PIM router is connected to OISM PEG devices in either data center.
 "[Multicast Control and Data Flow for Internal Source to External Receiver](#)" on page 1201 describes a use case where the source is behind an OISM server leaf devices in one data center. The external PIM domain connected to that data center has an external receiver, and both interconnected data centers also have internal receivers.

We show some of these additional use cases in the next sections.

Multicast Traffic Flow across DCI from Multihomed Source on DCI Gateway Devices

[Figure 117 on page 1198](#) shows a use case where the multicast source on VLAN 100 is multihomed to iGW 11 and iGW 12 in DC-1. You configure all of the leaf devices in this use case as OISM server leaf devices because all sources and receivers are inside the interconnected data centers.

Both of the interconnected data centers have internal subscribed receivers behind the following OISM server leaf devices:

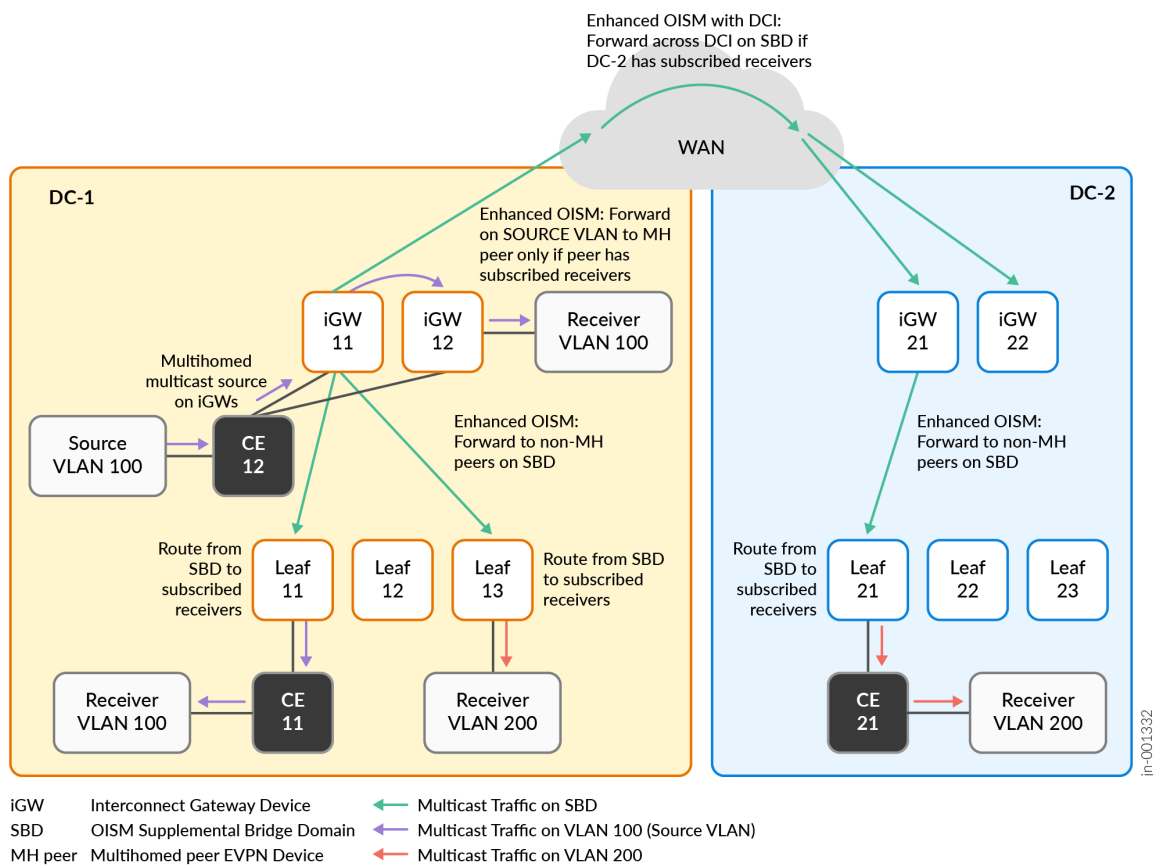
- In DC-1: Leaf 11, Leaf 13, and iGW 12
- In DC-2: Leaf 21

The multihoming peer iGW devices in each data center elect a DF; the other iGW devices become NDFs (also called backup DFs [BDFs]). The DF forwards or routes multicast source traffic to the other OISM leaf devices. The NDF devices forward traffic only to the receivers that are local to their own device.

With enhanced OISM operation, if the multicast source is multihomed to a set of OISM leaf devices, the ingress PE device that receives the traffic from the source will:

- Forward the traffic to its multihoming peers using the source VLAN.
- Route the traffic on the SBD to all the other OISM leaf devices that are not its multihoming peers.
- Use SMET to forward or route the traffic only to other OISM leaf devices with subscribed receivers.

Figure 117: Multihomed Multicast Source on DCI Gateways with Receiver across DCI



In this use case:

1. The multicast source is in DC-1 on VLAN 100, behind CE 12. CE 12 is multihomed to the two iGW devices in DC-1, iGW 11 and iGW 12.
2. Receivers in both DC-1 and DC-2 on VLAN 200 subscribe to the multicast data from the source in DC-1. Receivers in DC-1 on VLAN 100 behind iGW 12 and behind Leaf 11 also subscribe to the multicast data.
3. In DC-1, both iGW devices receive the multicast traffic from the source on VLAN 100. Both of the iGW devices in DC-1 know from receiving the SMET routes that there are receivers in DC-1 and remote receivers in DC-2. In this case, iGW 11 is the DF for the iGW devices in DC-1, so iGW 11:
 - a. Routes the traffic on the SBD across the DCI toward DC-2.
 - b. Forwards the traffic locally on the source VLAN to its multihoming peer, iGW 12.
 - c. Routes the traffic locally on the SBD to the other (non-multihoming peer) OISM leaf devices in DC-1.
4. In DC-1, the multihoming peer iGW device, iGW 12, forwards the traffic on the source VLAN to its local receiver. The other OISM leaf devices in DC-1 route the traffic from the SBD toward the receivers on the receiver VLANs (VLAN 100 and VLAN 200).
5. In DC-2, the iGW devices receive the multicast traffic on the SBD. In this case, iGW 21 is the DF for the iGW devices in DC-2, so iGW 21 forwards the traffic on the SBD to Leaf 21 toward the subscribed receiver.
6. In DC-2, Leaf 21 routes the multicast traffic toward the receiver on the receiver VLAN, VLAN 200.

See ["Multicast Next Hops Convergence Optimization" on page 1203](#) for more on how we improve EVPN multicast next hop convergence over the DCI when DF and NDF status changes for the peer iGW devices in a data center.

Multicast Traffic Flow from External Source to Internal Receivers

[Figure 118 on page 1200](#) shows a use case where the iGW devices in DC-1 also act as the OISM PEG devices to connect to a source in an external PIM domain.

The receivers in this use case are behind customer edge (CE) devices connected to the leaf devices inside each of the interconnected data centers.



NOTE: For simplicity, we omit showing the receivers behind the CE devices here. The CE devices don't serve in an EVPN OISM leaf device role and you don't configure OISM on these devices.

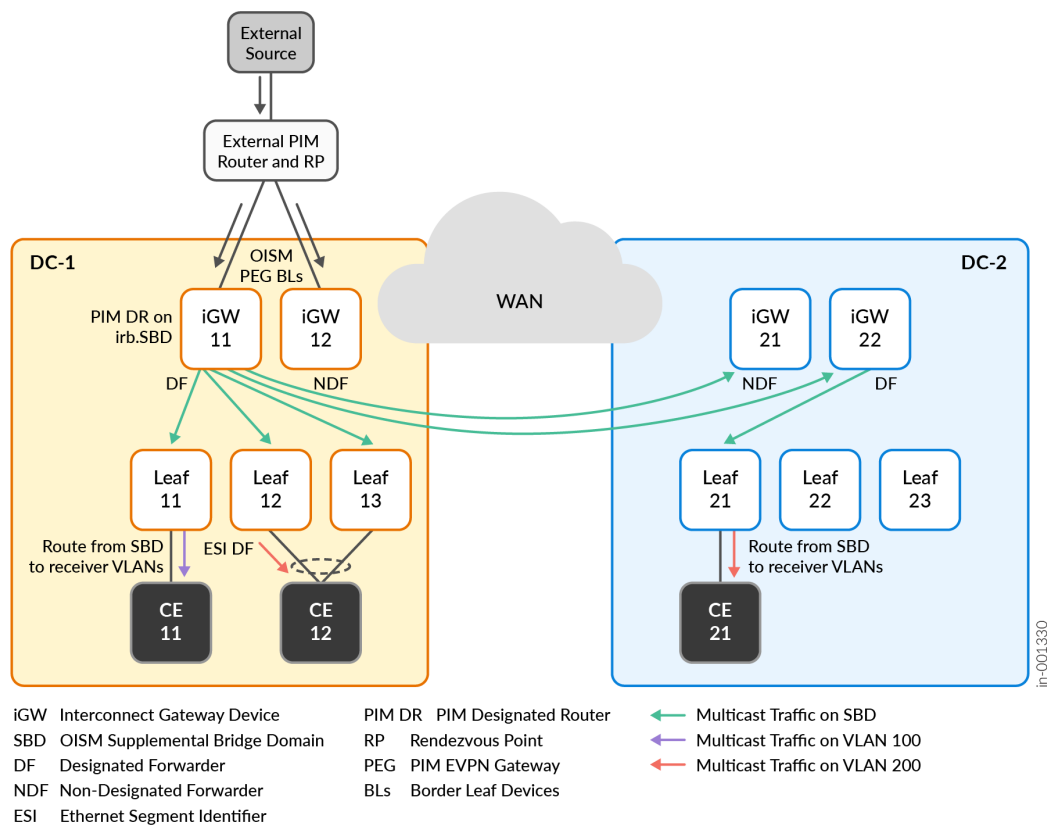
For details on how OISM works with an external source and internal receivers, see ["Multicast Traffic from an External Source to Receivers Inside the EVPN Data Center—L3 Interface Method or Non-EVPN IRB Method"](#) on page 1102.

You configure:

- The iGW devices in DC-1 as OISM border leaf PEG devices.
- The iGW devices in DC-2 as OISM server leaf devices.
- The remaining leaf devices in both data centers as OISM server leaf devices.

We include a topology variation here with a receiver behind CE 12 in DC-1, where CE 12 is multihomed to OISM server leaf devices Leaf 12 and Leaf 13 for redundancy. These two leaf devices share an Ethernet segment identifier (ESI) for the multihomed receiver, and elect a DF for the ESI. In this case, Leaf 12 is the DF that forwards the multicast traffic destined for the receiver behind CE 12.

Figure 118: Multicast Traffic from External Source to Internal Receivers through OISM Border Leaf PEG Devices



In this use case:

1. Receivers in DC-1 and DC-2 on VLAN 100 and VLAN 200 subscribe to the multicast data from the external source.
2. In DC-1, iGW 11 and iGW 12 receive the multicast traffic from the external source. Both of the iGWs in DC-1 know from receiving the SMET routes that there are receivers in DC-1 and remote receivers in DC-2. In this case, iGW 11 is the DF for the iGW devices in DC-1, so iGW 11:
 - a. Routes the traffic on the SBD across the DCI to the iGW devices in DC-2.
 - b. Routes the traffic locally on the SBD to the OISM leaf devices in DC-1 that are not its multihoming peers and have subscribed receivers—in this case, Leaf 11, Leaf 12 and Leaf 13.
3. In DC-1, the OISM leaf devices route the traffic from the SBD toward the receivers on the receiver VLANs—VLAN 100 and VLAN 200. In this case, a receiver is behind CE 12, which is multihomed to Leaf 12 and Leaf 13. Leaf 12 is the DF and forwards the multicast traffic toward that receiver.
4. In DC-2, the iGW devices receive the multicast traffic on the SBD. In this case, iGW 22 is the DF for the iGW devices in DC-2, so iGW 22 forwards the traffic on the SBD to Leaf 21 toward the subscribed receiver.
5. In DC-2, Leaf 21 routes the multicast traffic from the SBD toward the receiver on the receiver VLAN, VLAN 200.

Multicast Control and Data Flow for Internal Source to External Receiver

[Figure 119 on page 1202](#) shows a use case where the source is in DC-1. We have a receiver across the DCI in DC-2 and an external receiver. The iGW devices in DC-1 are the OISM PEG devices connecting to the receiver in an external PIM domain.

For details on how OISM works with an internal source and external receiver, see ["Multicast Traffic from an Internal Source to Receivers Outside the EVPN Data Center—L3 Interface Method or Non-EVPN IRB Method" on page 1094](#).

For details on how enhanced OISM ensures that the OISM PEG devices correctly perform PIM source registration for multicast sources inside an EVPN data center to an external PIM domain, see ["PIM Registration with Enhanced OISM for Internal Sources Based on EVPN Type 10 S-PMSI A-D Routes" on page 1115](#).

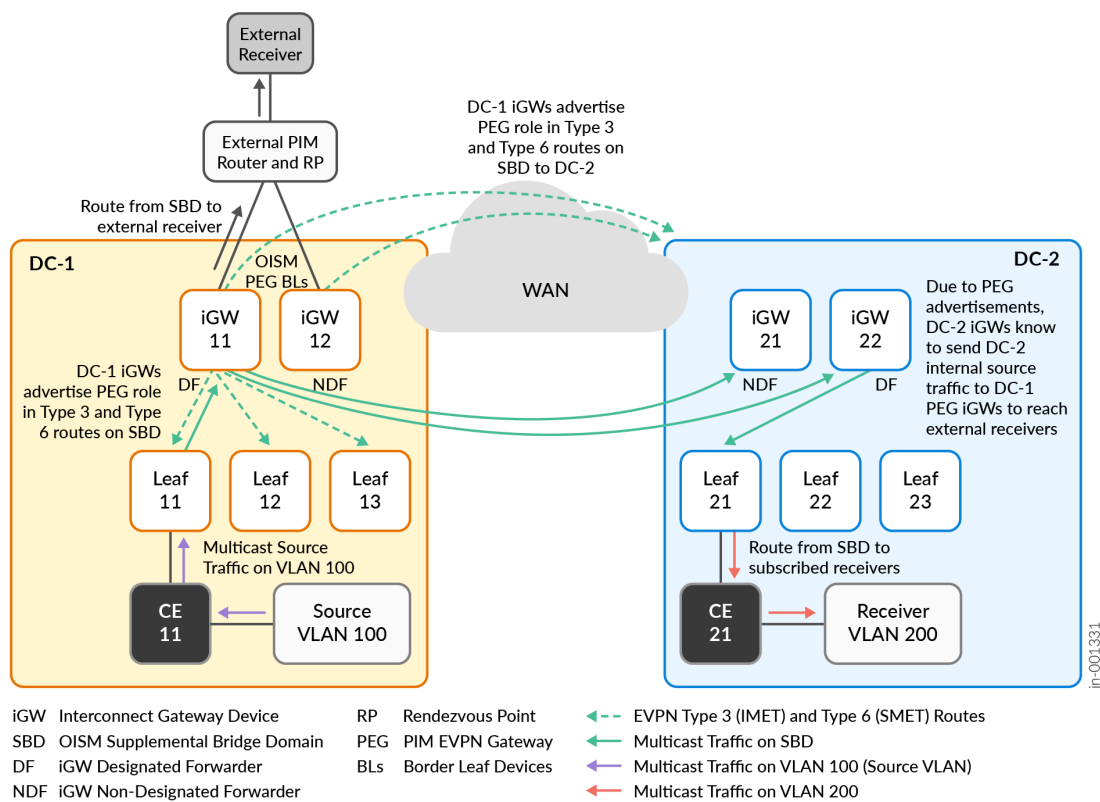
The multicast source and iGW devices in the OISM border leaf PEG role are in the same data center in this use case. However, with an external receiver, the multicast source might not be in the same data center as the OISM PEG devices that connect to the external receivers. The seamless enhanced OISM multicast over DCI implementation ensures that multicast source traffic from any of the interconnected data centers can reach external receivers wherever the OISM border leaf PEG devices are. As a result, [Figure 119 on page 1202](#) also shows the following control flow:

- The OISM PEG devices advertise their PEG role in the local data center and across the DCI to the other iGW devices.
- The OISM PEG devices send these advertisements on the OISM SBD in the SMET EVPN Type 6 routes, and in Inclusive Multicast Ethernet Tag (IMET) routes, which are EVPN Type 3 routes.

In this use case, you configure:

- The iGW devices in DC-1 as OISM border leaf PEG devices.
- The iGW devices in DC-2 as OISM server leaf devices.
- The remaining EVPN PE leaf devices in both data centers as OISM server leaf devices.

Figure 119: Multicast Traffic from Internal Source to External Receiver through OISM Border Leaf PEG Devices



In this use case:

1. The iGW devices in DC-1 connected to the external PIM domain advertise that they are OISM PEG devices using EVPN Type 3 and Type 6 routes. The PEG devices send these advertisements to all local OISM PE devices in DC-1 and across the DCI to the iGW devices in DC-2.

2. A receiver in DC-2 on VLAN 200 and an external receiver subscribe to the multicast data from the source on VLAN 100 in DC-1.
3. In DC-1, Leaf 11 receives multicast traffic from the source on VLAN 100. Leaf 11 knows from receiving EVPN Type 6 SMET routes that DC-2 has a receiver and there is an external receiver connected by way of the iGW devices in OISM PEG role. As a result, Leaf 11 routes the multicast traffic to iGW 11 and iGW 12 on the SBD.
4. In DC-1, iGW 11 and iGW 12 receive the multicast traffic from Leaf 11 on the SBD. The iGW devices know from receiving EVPN Type 6 SMET routes that there are remote receivers in DC-2. The iGW devices also act as OISM border leaf PEG devices to the external receiver. In this case, [Figure 119 on page 1202](#) shows that:
 - a. iGW 11 is the DF for the iGW devices in DC-1, so iGW 11 forwards the multicast traffic on the SBD across the DCI to the iGW devices in DC-2.
 - b. iGW 11 routes the multicast traffic toward the external receiver.
5. In DC-2, the iGW devices receive the multicast traffic on the SBD. In this case, iGW 22 is the DF for the iGW devices in DC-2, so iGW 22 forwards the traffic on the SBD to Leaf 21 toward the subscribed receiver.
6. In DC-2, Leaf 21 routes the multicast traffic from the SBD toward the receiver on the receiver VLAN, VLAN 200.

If DC-2 included a multicast data source, an external receiver connected to DC-1 might subscribe to that source data. In that case, the iGW devices in DC-2 receive the PEG role advertisements from the DC-1 iGW devices. The iGW devices in DC-2 would send multicast source traffic to an external receiver seamlessly using their interconnection to the DC-1 iGW devices acting as the OISM border leaf PEG devices.

Multicast Next Hops Convergence Optimization

When you interconnect EVPN data centers, we identify the peer multihomed iGW devices in each data center with a unique interconnect Ethernet segment ID (I-ESI). The peer iGW devices in each data center have a designated forwarder (DF) that forwards traffic across the interconnection and to the local receivers in the data center. The other iGW devices that are the NDF devices (sometimes called backup DF [BDF] devices) forward traffic only to receivers that are local to their own device.

The control plane on the iGW devices determines the multicast next hops for the DF and NDF iGW devices, and installs the next hops on the Packet Forwarding Engine (PFE) to efficiently forward traffic toward the destinations.

To improve multicast next hop information convergence in the PFE for EVPN multicast traffic over DCI, the iGW devices use a session model to track I-ESI DF and NDF status and multicast next hop information.

The control plane on the iGW devices:

- Assigns a session ID to identify the DF and NDF device sessions.
- Sends session status update messages to the PFE when the DF and NDF status for the I-ESI changes.

Upon receiving DF and NDF session status updates, the PFE on the iGW devices can quickly update the forwarding paths accordingly.

You can use the *show multicast next-hops session* command to see the DF and NDF session information for the I-ESI on iGW devices that you configure to support enhanced OISM traffic across the DCI.

Configure EVPN-VXLAN DCI with Enhanced OISM

IN THIS SECTION

- [Configure EVPN-VXLAN in Each Data Center | 1204](#)
- [Configure DCI Gateway Devices across Data Centers | 1205](#)
- [Configure Enhanced OISM in Each Data Center | 1205](#)

To enable seamless multicast optimization with DCI and enhanced OISM, you configure many statements to set up EVPN-VXLAN, DCI, and enhanced OISM in each data center.

Keep in mind the following when planning an optimized DCI multicast environment:

- We support enhanced OISM only in EVPN-VXLAN ERB overlay architectures.
- If your environment includes external multicast sources or receivers, you configure EVPN devices in one of the data centers to act as the OISM border leaf PEG devices to connect to the external PIM domain. You can configure the iGW devices to also be the OISM PEG devices, or other OISM leaf devices in one of the data centers can be the OISM PEG devices.

This section provides a high-level overview of the different components you configure for multicast with enhanced OISM over DCI. We also include a simple sample configuration in ["Sample Configuration for DCI with Enhanced OISM" on page 1206](#), and corresponding verification show command outputs in ["Show Commands to Verify DCI with Enhanced OISM" on page 1232](#).

Configure EVPN-VXLAN in Each Data Center

Configure the EVPN-VXLAN networks in the data centers on both sides of the interconnection. See ["Configure DCI Gateway Devices across Data Centers" on page 1205](#) for more on how to set up the data centers for DCI.

We support DCI and enhanced OISM with interconnected EVPN-VXLAN data centers that use IPv4 underlay peering or IPv6 underlay peering. You can transfer IPv4 multicast data or IPv6 multicast data across DCI with either type of underlay.

See the following references for how to configure an IPv6 underlay with EVPN-VXLAN:

- ["EVPN-VXLAN with an IPv6 Underlay" on page 897](#)
- ["Example: Configure an IPv6 Underlay for Layer 2 VXLAN Gateway Leaf Devices" on page 905](#)
- [IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGp](#)

Configure DCI Gateway Devices across Data Centers

You configure the iGW devices to interconnect the data centers the same way you would without configuring enhanced OISM.

See the *interconnect* configuration hierarchy, and the following reference for a DCI stitching configuration example:

- [Configure VXLAN Stitching for Layer 2 Data Center Interconnect](#)

Configure Enhanced OISM in Each Data Center

Configure enhanced OISM in the EVPN-VXLAN data center on each side of the DCI in one of the many different use cases on this page with multicast source and multicast receiver locations. ["Optimized Intersubnet Multicast in EVPN Networks" on page 1049](#) has details on other supported OISM use cases in a data center.

Note the following about configuring enhanced OISM:

- You enable enhanced OISM mode on all OISM devices using the `enhanced-oism` option at the `[edit forwarding-options multicast-replication evpn irb]` hierarchy level. You use the `enhanced-oism` option instead of the regular OISM mode `oism` option at the same hierarchy level. The `enhanced-oism` and `oism` options are mutually exclusive. All EVPN PE devices in each data center must use enhanced OISM mode.
- With enhanced OISM, you can configure each OISM device with only the VLANs that device hosts. You don't need to configure all VLANs in the EVPN network on all OISM devices like you do with regular OISM. The exception is multihoming peer OISM devices—on those devices, you must configure the OISM revenue VLANs symmetrically on all peers. This requirement applies to the peer DCI gateways in each data center when you configure them as OISM server leaf devices or OISM border leaf devices.
- You must configure the OISM SBD with the same VLAN ID in the matching tenant virtual routing and forwarding (VRF) instances in the interconnected data centers.

OISM requires some common configuration statements on both OISM server leaf devices and OISM border leaf devices. You configure some statements only on the server leaf devices, and configure some statements only on the border leaf devices. See ["Optimized Intersubnet Multicast in EVPN Networks" on page 1049](#) for details on understanding and configuring server leaf and border leaf OISM device roles and all elements for OISM to work (some options are platform-specific).

See the following sections in ["Optimized Intersubnet Multicast in EVPN Networks" on page 1049](#) for configuration considerations, steps to configure each OISM device role, and CLI commands to verify OISM operation:

- ["Considerations for OISM Configurations" on page 1117](#)
- ["Configure Common OISM Elements on Border Leaf Devices and Server Leaf Devices " on page 1158](#)
- ["Configure Server Leaf Device OISM Elements" on page 1169](#)
- ["Configure Border Leaf Device OISM Elements with Classic L3 Interface Method" on page 1175](#)
- ["CLI Commands to Verify the OISM Configuration" on page 1180](#)

Sample Configuration for DCI with Enhanced OISM

IN THIS SECTION

- [Related Platform-Specific Configuration Elements for EVPN-VXLAN Configurations | 1209](#)
- [WAN Router EVPN Overlay Sample Configuration | 1210](#)
- [DC-1: iGW-11 Sample Configuration | 1211](#)
- [DC-1: iGW-12 Sample Configuration | 1216](#)
- [DC-1: Leaf-11 Sample Configuration | 1220](#)
- [DC-2: iGW-21 Sample Configuration | 1224](#)
- [DC-2: Leaf-21 Sample Configuration | 1228](#)

The following sections show a simple sample configuration for DCI with enhanced OISM with two stitched data centers, DC-1 and DC-2, similar to the topologies in the use cases in [Figure 116 on page 1195](#) through [Figure 119 on page 1202](#).

The setup uses the following parameters:

- Interconnect WAN
 - OSPF underlay peering

- IBGP overlay peering for EVPN
- DCI EVPN instance: evpn-vxlan-A
 - EBGp underlay and overlay peering
 - DC-1 EVPN instance RT—target:9:9
 - DC-2 EVPN instance RT—target:21:21
 - WAN (interconnect) RT—target:11:11
- L3 VRF instance: VRF-1 (RT target:1:2)
- OISM SBD VLAN: VLAN_4
- OISM tenant VLANs: VLAN_2, VLAN_3, VLAN_5



NOTE: Remember that with enhanced OISM, you must configure the same tenant VLANs on OISM multihoming peer PE devices. On all of the other OISM devices, you only need to configure the VLANs hosted by that device.

- Multicast groups (IPv4): 233.252.0.1, 233.252.0.2
- Multicast source IP address: 172.20.1.15
- External multicast PIM RP static address: (IPv4) 172.30.7.7, (IPv6) 2001:0db8::172:30:7:7

Table 64: Sample Configuration Parameters

Device	Device Address	VRF-1 Router ID	AS #	EVPN Instance RD	VRF-1 RD	Interconnect RD	Tenant VLANs
WAN Router	192.168.50.50	n/a	65000	n/a	n/a	n/a	n/a

Table 64: Sample Configuration Parameters (Continued)

Device	Device Address	VRF-1 Router ID	AS #	EVPN Instance RD	VRF-1 RD	Interconnect RD	Tenant VLANs
<p>DC-1—AS 65111</p> <p>ERB overlay with transit lean spine devices connecting the multihoming peer interconnect gateway devices iGW-11 and iGW-12 devices to each other and to the other leaf devices:</p> <ul style="list-style-type: none"> • Spine-1: 192.168.7.1, AS# 65107 • Spine-2: 192.168.8.1, AS# 65108 <p>Interconnect ESI for peer iGW devices iGW-11 and iGW-12—00:0a:0b:0c:0d:0a:0b:0c:0d:0a</p>							
iGW-11	192.168.1.1	192.168.1.2	65101	192.168.1.1:9	192.168.1.1:8	111:100	VLAN_2, VLAN_5
iGW-12	192.168.2.1	192.168.2.2	65102	192.168.1.1:9	192.168.2.1:8	112:100	VLAN_2, VLAN_5
Leaf-11	192.168.3.1	192.168.3.2	65103	192.168.3.1:9	192.168.3.1:8	n/a	VLAN_2, VLAN_5
Leaf-12	192.168.4.1	192.168.4.2	65104	192.168.4.1:9	192.168.4.1:8	n/a	VLAN_2, VLAN_5
Leaf-13	192.168.5.1	192.168.5.2	65105	192.168.5.1:9	192.168.5.1:8	n/a	VLAN_3, VLAN_5
<p>DC-2—AS 65222</p> <p>In this ERB overlay data center, Leaf-21, Leaf-22, Leaf-23 are directly connected to the multihoming peer interconnect gateway devices iGW-21 and iGW-22, without a transit lean spine layer.</p> <p>Interconnect ESI for peer iGW devices iGW-21 and iGW-22—00:aa:bb:cc:dd:aa:bb:cc:dd:aa</p>							
iGW-21	192.168.21.1	192.168.21.2	65201	192.168.21.1:9	192.168.21.1:8	221:100	VLAN_3, VLAN_5

Table 64: Sample Configuration Parameters (Continued)

Device	Device Address	VRF-1 Router ID	AS #	EVPN Instance RD	VRF-1 RD	Interconnect RD	Tenant VLANs
iGW-22	192.168.22.1	192.168.22.2	65202	192.168.22.1:9	192.168.22.1:8	222:100	VLAN_3, VLAN_5
Leaf-21	192.168.23.1	192.168.23.2	65203	192.168.23.1:9	192.168.23.1:8	n/a	VLAN_2, VLAN_5
Leaf-22	192.168.24.1	192.168.24.2	65204	192.168.24.1:9	192.168.24.1:8	n/a	VLAN_2, VLAN_5
Leaf-23	192.168.25.1	192.168.25.2	65205	192.168.25.1:9	192.168.25.1:8	n/a	VLAN_3, VLAN_5

We provide sample configurations for DCI seamless stitching and enhanced OISM on the following devices in the topology:

- WAN router: Simple EVPN overlay for seamless stitching across DC-1 and DC-2
- DC-1: Multihoming interconnect peers iGW-11 and iGW-12, and Leaf-11
- DC-2: iGW-21 and Leaf-21

The configuration on the multihoming interconnect peer iGW-22 in DC-2 would be similar to the configuration on iGW-21. The same applies to the configurations for Leaf-12, Leaf-13, Leaf-22, and Leaf-23.

Related Platform-Specific Configuration Elements for EVPN-VXLAN Configurations

We require you to include the following statements in EVPN-VXLAN configurations with OISM or DCI on some platforms. Remember to add these statements to the indicated type of OISM device or DCI gateway device configuration on the stated platforms:

- `conserve-mcast-routes-in-pfe` at the [edit routing-instances *evpn-instance-name* multicast-snooping-options oism] hierarchy level:

Required on devices in the OISM server leaf or border leaf role on Junos OS Evolved QFX Series switches and ACX Series routers. For example, in our configuration:

```
set routing-instances evpn-vxlan-A multicast-snooping-options oism conserve-mcast-routes-in-pfe
```

For details, see *oism (Multicast Snooping Options)* and ["ACX Series Routers, QFX5130-32CD Switches, and QFX5700 Switches as Server Leaf and Border Leaf Devices with OISM"](#) on page 1125.

- vxlan-dci-enable at the [edit <routing-instances *instance-name*> forwarding-options] hierarchy level:

Required only on EX4400 switches you configure as DCI gateway devices for DCI stitching to work (with or without enhanced OISM). For example:

```
set forwarding-options vxlan-dci-enable
```

For details, see [evpn-vxlan](#).

- shared-tunnels at the [edit forwarding-options evpn-vxlan] hierarchy level:

Required for any EVPN devices that are Junos OS EX Series or QFX Series switches. For example:

```
set forwarding-options evpn-vxlan shared-tunnels
```

For details, see *evpn-vxlan* and ["Shared VTEP Tunnels in EVPN-VXLAN Fabrics with Multiple MAC-VRF Routing Instances"](#) on page 45.

WAN Router EVPN Overlay Sample Configuration

```
set chassis aggregated-devices ethernet device-count 10
set interfaces mge-0/0/12 description "CONNECTED TO iGW-11"
set interfaces mge-0/0/12 unit 0 family inet address 10.1.51.2/24
set interfaces mge-0/0/9 description "CONNECTED TO iGW-12"
set interfaces mge-0/0/9 unit 0 family inet address 10.2.52.2/24
set interfaces mge-0/0/10 description "CONNECTED TO iGW-21"
set interfaces mge-0/0/10 unit 0 family inet address 10.21.53.2/24
set interfaces mge-0/0/11 description "CONNECTED TO iGW-22"
set interfaces mge-0/0/11 unit 0 family inet address 10.22.54.2/24
set interfaces lo0 unit 0 family inet address 192.168.50.50/32

set routing-options router-id 192.168.50.50
```

```

set routing-options autonomous-system 65000

set protocols bgp group OVERLAY-WAN type internal
set protocols bgp group OVERLAY-WAN family evpn signaling
set protocols bgp group OVERLAY-WAN cluster 192.168.50.50
set protocols bgp group OVERLAY-WAN local-as 65000
set protocols bgp group OVERLAY-WAN neighbor 192.168.1.1
set protocols bgp group OVERLAY-WAN neighbor 192.168.2.1
set protocols bgp group OVERLAY-WAN neighbor 192.168.21.1
set protocols bgp group OVERLAY-WAN neighbor 192.168.22.1
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface mge-0/0/9.0
set protocols ospf area 0.0.0.0 interface mge-0/0/10.0
set protocols ospf area 0.0.0.0 interface mge-0/0/11.0
set protocols ospf area 0.0.0.0 interface mge-0/0/12.0
set protocols ospf area 0.0.0.0 interface lo0.0

set protocols lldp port-id-subtype interface-name
set protocols lldp tlv-select system-name
set protocols lldp interface all

```

DC-1: iGW-11 Sample Configuration

```

set system host-name iGW-11
set routing-options router-id 192.168.1.1
set routing-options autonomous-system 65111

#Interfaces configuration

set chassis aggregated-devices ethernet device-count 10
set chassis network-services enhanced-ip
set protocols lldp port-id-subtype interface-name
set protocols lldp tlv-select system-name
set protocols lldp tlv-select system-capabilities
set protocols lldp interface all

set interfaces et-0/0/0 number-of-sub-ports 4
set interfaces et-0/0/0 speed 10g

set interfaces et-0/0/0:0 description "CONNECTED TO CE-Host"
set interfaces et-0/0/0:0 unit 0 family ethernet-switching interface-mode trunk

```

```
set interfaces et-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_2
set interfaces et-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_5
```

```
set interfaces et-0/0/0:1 description "CONNECTED TO CE-Host"
set interfaces et-0/0/0:1 ether-options 802.3ad ae0
set interfaces ae0 description "CONNECTED TO CE-Host"
set interfaces ae0 esi 00:44:44:44:44:44:44:44:44
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 00:44:44:44:44:44
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_2
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_5
```

```
set interfaces et-0/0/0:2 description "CONNECTED TO PIM RP"
set interfaces et-0/0/0:2 unit 0 family inet address 172.30.80.1/24
set interfaces et-0/0/0:2 unit 0 family inet6 address 2001:0db8:80::1/64
```

```
set interfaces et-0/0/1 number-of-sub-ports 4
set interfaces et-0/0/1 speed 10g
set interfaces et-0/0/1:0 description "CONNECTED TO SPINE-1"
set interfaces et-0/0/1:0 ether-options 802.3ad ae1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 family inet address 10.1.7.2/24
set interfaces et-0/0/1:1 description "CONNECTED TO SPINE-2"
set interfaces et-0/0/1:1 unit 0 family inet address 10.1.8.2/24
set interfaces et-0/0/1:2 description "CONNECTED TO WAN Router"
set interfaces et-0/0/1:2 unit 0 family inet address 10.1.51.1/24
```

EBGP underlay and overlay peering for EVPN

```
set policy-options policy-statement EXPORT-LO0 term LOOPBACK from protocol direct
set policy-options policy-statement EXPORT-LO0 term LOOPBACK from interface lo0.0
set policy-options policy-statement EXPORT-LO0 term LOOPBACK then accept
set policy-options policy-statement EXPORT-LO0 term REJECT then reject
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set routing-options forwarding-table export load-balancing-policy
```

```
set protocols bgp group IP-UNDERLAY type external
set protocols bgp group IP-UNDERLAY export EXPORT-LO0
```

```

set protocols bgp group IP-UNDERLAY local-as 10001
set protocols bgp group IP-UNDERLAY multipath multiple-as
set protocols bgp group IP-UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group IP-UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group IP-UNDERLAY bfd-liveness-detection session-mode automatic
set protocols bgp group IP-UNDERLAY neighbor 10.1.7.1 peer-as 65107
set protocols bgp group IP-UNDERLAY neighbor 10.1.8.1 peer-as 65108
set protocols bgp group IP-OVERLAY type external
set protocols bgp group IP-OVERLAY multihop ttl 2
set protocols bgp group IP-OVERLAY multihop no-nexthop-change
set protocols bgp group IP-OVERLAY local-address 192.168.1.1
set protocols bgp group IP-OVERLAY family inet-vpn unicast
set protocols bgp group IP-OVERLAY family evpn signaling
set protocols bgp group IP-OVERLAY local-as 65101
set protocols bgp group IP-OVERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group IP-OVERLAY bfd-liveness-detection multiplier 3
set protocols bgp group IP-OVERLAY bfd-liveness-detection session-mode automatic
set protocols bgp group IP-OVERLAY neighbor 192.168.4.1 peer-as 65104
set protocols bgp group IP-OVERLAY neighbor 192.168.5.1 peer-as 65105
set protocols bgp group IP-OVERLAY neighbor 192.168.2.1 peer-as 65102
set protocols bgp group IP-OVERLAY neighbor 192.168.3.1 peer-as 65103

```

WAN overlay

```

set protocols bgp group OVERLAY-WAN type internal
set protocols bgp group OVERLAY-WAN local-address 192.168.1.1
set protocols bgp group OVERLAY-WAN family evpn signaling
set protocols bgp group OVERLAY-WAN local-as 65000
set protocols bgp group OVERLAY-WAN neighbor 192.168.50.50

```

EVPN instance with enhanced OISM - enable snooping and SMET, define tenant VLANs and OISM SBD

```

set interfaces irb mtu 1514
set interfaces irb unit 2 virtual-gateway-accept-data
set interfaces irb unit 2 family inet address 10.102.1.1/24 virtual-gateway-address 10.102.1.100
set interfaces irb unit 2 family inet6 address 2001:0db8:10:102::1/64 virtual-gateway-address 2001:0db8:10:102::100
set interfaces irb unit 2 mac 00:01:00:01:00:01
set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.104.1.1/24 virtual-gateway-address 10.104.1.100
set interfaces irb unit 4 family inet6 address 2001:0db8:10:104::1/64 virtual-gateway-address 2001:0db8:10:104::100
set interfaces irb unit 4 mac 00:01:00:03:00:01

```

```

set interfaces irb unit 5 virtual-gateway-accept-data
set interfaces irb unit 5 family inet address 10.105.1.1/24 virtual-gateway-address 10.105.1.100
set interfaces irb unit 5 family inet6 address 2001:0db8:10:105::1/64 virtual-gateway-address
2001:0db8:10:105::100
set interfaces irb unit 5 mac 00:01:00:04:00:01
set interfaces lo0 unit 0 family inet address 192.168.1.1/32 primary
set interfaces lo0 unit 1 family inet address 192.168.1.2/32
set interfaces lo0 unit 1 family inet6 address 2001:0db8::192:168:1:2/128

```

```

set forwarding-options multicast-replication evpn irb enhanced-oism

```

```

set routing-instances evpn-vxlan-A instance-type mac-vrf
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_2 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_4 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_5 proxy
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_2
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_4
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_5
set routing-instances evpn-vxlan-A protocols evpn encapsulation vxlan
set routing-instances evpn-vxlan-A protocols evpn default-gateway no-gateway-community
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 2
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 4
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 5
set routing-instances evpn-vxlan-A vtep-source-interface lo0.0
set routing-instances evpn-vxlan-A service-type vlan-aware
set routing-instances evpn-vxlan-A interface et-0/0/0:0.0
set routing-instances evpn-vxlan-A interface ae0.0
set routing-instances evpn-vxlan-A route-distinguisher 192.168.1.1:9
set routing-instances evpn-vxlan-A vrf-target target:9:9

```

```

set routing-instances evpn-vxlan-A vlans VLAN_2 vlan-id 2
set routing-instances evpn-vxlan-A vlans VLAN_2 l3-interface irb.2
set routing-instances evpn-vxlan-A vlans VLAN_2 vxlan vni 2
set routing-instances evpn-vxlan-A vlans VLAN_4 vlan-id 4
set routing-instances evpn-vxlan-A vlans VLAN_4 l3-interface irb.4
set routing-instances evpn-vxlan-A vlans VLAN_4 vxlan vni 4
set routing-instances evpn-vxlan-A vlans VLAN_5 vlan-id 5
set routing-instances evpn-vxlan-A vlans VLAN_5 l3-interface irb.5
set routing-instances evpn-vxlan-A vlans VLAN_5 vxlan vni 5

```

DCI parameters and multihoming peer designation

```

set routing-instances evpn-vxlan-A protocols evpn interconnect vrf-target target:11:11

```

```

set routing-instances evpn-vxlan-A protocols evpn interconnect route-distinguisher 111:100
set routing-instances evpn-vxlan-A protocols evpn interconnect esi 00:0a:0b:0c:0d:0a:0b:0c:0d:0a
set routing-instances evpn-vxlan-A protocols evpn interconnect esi all-active
set routing-instances evpn-vxlan-A protocols evpn interconnect esi df-election-type preference
value 300
set routing-instances evpn-vxlan-A protocols evpn interconnect interconnected-vni-list all

```

```

set protocols evpn interconnect-multihoming-peer-gateways 192.168.2.1

```

```

set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface et-0/0/1:2.0

```

Tenant VRF VRF-1 configuration, including OISM PEG setting and external multicast routing parameters

```

set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 routing-options router-id 192.168.1.2
set routing-instances VRF-1 protocols evpn oism supplemental-bridge-domain-irb irb.4
set routing-instances VRF-1 protocols evpn oism pim-evpn-gateway
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface lo0.1
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.2 passive
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface et-0/0/0:2.0
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface lo0.1
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.2 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface et-0/0/0:2.0
set routing-instances VRF-1 protocols pim rp static address 172.30.7.7
set routing-instances VRF-1 protocols pim rp static address 2001:0db8::172:30:7:7
set routing-instances VRF-1 protocols pim interface et-0/0/0:2.0
set routing-instances VRF-1 protocols pim interface irb.2 distributed-dr
set routing-instances VRF-1 protocols pim interface irb.4 accept-remote-source
set routing-instances VRF-1 protocols pim interface irb.5 distributed-dr
set routing-instances VRF-1 protocols pim interface lo0.1
set routing-instances VRF-1 protocols pim disable-packet-register
set routing-instances VRF-1 interface et-0/0/0:2.0
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface irb.5
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 route-distinguisher 192.168.1.1:8

```



```
set routing-instances VRF-1 vrf-target target:1:2
set routing-instances VRF-1 vrf-table-label
```

DC-1: iGW-12 Sample Configuration

```
set system host-name iGW-12
set routing-options router-id 192.168.2.1
set routing-options autonomous-system 65111

# Interfaces configuration

set chassis aggregated-devices ethernet device-count 10
set chassis network-services enhanced-ip
set protocols lldp port-id-subtype interface-name
set protocols lldp tlv-select system-name
set protocols lldp tlv-select system-capabilities
set protocols lldp interface all

set interfaces et-0/0/0 number-of-sub-ports 4
set interfaces et-0/0/0 speed 10g

set interfaces et-0/0/0:0 description "CONNECTED TO CE-Host"
set interfaces et-0/0/0:0 unit 0 family ethernet-switching interface-mode trunk
set interfaces et-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_2
set interfaces et-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_5

set interfaces et-0/0/0:1 description "CONNECTED TO CE-Host"
set interfaces et-0/0/0:1 ether-options 802.3ad ae0
set interfaces ae0 description "CONNECTED TO CE-Host"
set interfaces ae0 esi 00:44:44:44:44:44:44:44:44
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 00:44:44:44:44:44
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_2
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_5

set interfaces et-0/0/0:2 description "CONNECTED TO PIM RP"
set interfaces et-0/0/0:2 unit 0 family inet address 172.30.90.1/24
```

```
set interfaces et-0/0/0:2 unit 0 family inet6 address 2001:0db8:90::1/64
```

```
set interfaces et-0/0/1 number-of-sub-ports 4
```

```
set interfaces et-0/0/1 speed 10g
```

```
set interfaces et-0/0/1:0 description "CONNECTED TO SPINE-1"
```

```
set interfaces et-0/0/1:0 unit 0 family inet address 10.2.7.2/24
```

```
set interfaces et-0/0/1:1 description "CONNECTED TO SPINE-2"
```

```
set interfaces et-0/0/1:1 unit 0 family inet address 10.2.8.2/24
```

```
set interfaces et-0/0/1:2 description "CONNECTED TO WAN Router"
```

```
set interfaces et-0/0/1:2 unit 0 family inet address 10.2.52.1/24
```

EBGp underlay and overlay peering for EVpn

```
set policy-options policy-statement EXPORT-LO0 term LOOPBACK from protocol direct
```

```
set policy-options policy-statement EXPORT-LO0 term LOOPBACK from interface lo0.0
```

```
set policy-options policy-statement EXPORT-LO0 term LOOPBACK then accept
```

```
set policy-options policy-statement EXPORT-LO0 term REJECT then reject
```

```
set policy-options policy-statement load-balancing-policy then load-balance per-packet
```

```
set routing-options forwarding-table export load-balancing-policy
```

```
set protocols bgp group IP-UNDERLAY type external
```

```
set protocols bgp group IP-UNDERLAY export EXPORT-LO0
```

```
set protocols bgp group IP-UNDERLAY local-as 65102
```

```
set protocols bgp group IP-UNDERLAY multipath multiple-as
```

```
set protocols bgp group IP-UNDERLAY bfd-liveness-detection minimum-interval 350
```

```
set protocols bgp group IP-UNDERLAY bfd-liveness-detection multiplier 3
```

```
set protocols bgp group IP-UNDERLAY bfd-liveness-detection session-mode automatic
```

```
set protocols bgp group IP-UNDERLAY neighbor 10.2.7.1 peer-as 65107
```

```
set protocols bgp group IP-UNDERLAY neighbor 10.2.8.1 peer-as 65108
```

```
set protocols bgp group IP-OVERLAY type external
```

```
set protocols bgp group IP-OVERLAY multihop ttl 2
```

```
set protocols bgp group IP-OVERLAY multihop no-nexthop-change
```

```
set protocols bgp group IP-OVERLAY local-address 192.168.2.1
```

```
set protocols bgp group IP-OVERLAY family inet-vpn unicast
```

```
set protocols bgp group IP-OVERLAY family evpn signaling
```

```
set protocols bgp group IP-OVERLAY local-as 65102
```

```
set protocols bgp group IP-OVERLAY bfd-liveness-detection minimum-interval 350
```

```
set protocols bgp group IP-OVERLAY bfd-liveness-detection multiplier 3
```

```
set protocols bgp group IP-OVERLAY bfd-liveness-detection session-mode automatic
```

```
set protocols bgp group IP-OVERLAY neighbor 192.168.3.1 peer-as 65103
```

```
set protocols bgp group IP-OVERLAY neighbor 192.168.5.1 peer-as 65105
```

```
set protocols bgp group IP-OVERLAY neighbor 192.168.1.1 peer-as 65101
```

```
set protocols bgp group IP-OVERLAY neighbor 192.168.4.1 peer-as 65104
```

WAN overlay

```

set protocols bgp group OVERLAY-WAN type internal
set protocols bgp group OVERLAY-WAN local-address 192.168.2.1
set protocols bgp group OVERLAY-WAN family evpn signaling
set protocols bgp group OVERLAY-WAN local-as 65000
set protocols bgp group OVERLAY-WAN neighbor 192.168.50.50

```

EVPN instance with enhanced OISM - enable snooping and SMET, define tenant VLANs and OISM SBD, and their IRB interfaces

```

set interfaces irb mtu 1514
set interfaces irb unit 2 virtual-gateway-accept-data
set interfaces irb unit 2 family inet address 10.102.1.2/24 virtual-gateway-address 10.102.1.100
set interfaces irb unit 2 family inet6 address 2001:0db8:10:102::2/64 virtual-gateway-address 2001:0db8:10:102::100
set interfaces irb unit 2 mac 00:01:00:01:00:02
set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.104.1.2/24 virtual-gateway-address 10.104.1.100
set interfaces irb unit 4 family inet6 address 2001:0db8:10:104::2/64 virtual-gateway-address 2001:0db8:10:104::100
set interfaces irb unit 4 mac 00:01:00:03:00:02
set interfaces irb unit 5 virtual-gateway-accept-data
set interfaces irb unit 5 family inet address 10.105.1.2/24 virtual-gateway-address 10.105.1.100
set interfaces irb unit 5 family inet6 address 2001:0db8:10:105::2/64 virtual-gateway-address 2001:0db8:10:105::100
set interfaces irb unit 5 mac 00:01:00:04:00:02
set interfaces lo0 unit 0 family inet address 192.168.2.1/32 primary
set interfaces lo0 unit 1 family inet address 192.168.2.2/32
set interfaces lo0 unit 1 family inet6 address 2001:0db8::192:168:2:2/128

set forwarding-options multicast-replication evpn irb enhanced-oism

set routing-instances evpn-vxlan-A instance-type mac-vrf
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_2 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_4 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_5 proxy
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_2
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_4
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_5
set routing-instances evpn-vxlan-A protocols evpn encapsulation vxlan
set routing-instances evpn-vxlan-A protocols evpn default-gateway no-gateway-community

```

```

set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 2
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 4
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 5
set routing-instances evpn-vxlan-A vtep-source-interface lo0.0
set routing-instances evpn-vxlan-A service-type vlan-aware
set routing-instances evpn-vxlan-A interface et-0/0/0:0.0
set routing-instances evpn-vxlan-A interface ae0.0
set routing-instances evpn-vxlan-A route-distinguisher 192.168.2.1:9
set routing-instances evpn-vxlan-A vrf-target target:9:9

```

```

set routing-instances evpn-vxlan-A vlans VLAN_2 vlan-id 2
set routing-instances evpn-vxlan-A vlans VLAN_2 l3-interface irb.2
set routing-instances evpn-vxlan-A vlans VLAN_2 vxlan vni 2
set routing-instances evpn-vxlan-A vlans VLAN_4 vlan-id 4
set routing-instances evpn-vxlan-A vlans VLAN_4 l3-interface irb.4
set routing-instances evpn-vxlan-A vlans VLAN_4 vxlan vni 4
set routing-instances evpn-vxlan-A vlans VLAN_5 vlan-id 5
set routing-instances evpn-vxlan-A vlans VLAN_5 l3-interface irb.5
set routing-instances evpn-vxlan-A vlans VLAN_5 vxlan vni 5

```

DCI parameters and multihoming peer designation

```

set routing-instances evpn-vxlan-A protocols evpn interconnect vrf-target target:11:11
set routing-instances evpn-vxlan-A protocols evpn interconnect route-distinguisher 112:100
set routing-instances evpn-vxlan-A protocols evpn interconnect esi 00:0a:0b:0c:0d:0a:0b:0c:0d:0a
set routing-instances evpn-vxlan-A protocols evpn interconnect esi all-active
set routing-instances evpn-vxlan-A protocols evpn interconnect esi df-election-type preference
value 100
set routing-instances evpn-vxlan-A protocols evpn interconnect interconnected-vni-list all

set protocols evpn interconnect-multihoming-peer-gateways 192.168.1.1

set protocols ospf area 0.0.0.0 interface et-0/0/1:2.0
set protocols ospf area 0.0.0.0 interface lo0.0

```

Tenant VRF VRF-1 configuration, including OISM PEG setting and external multicast routing parameters

```

set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 routing-options router-id 192.168.2.2
set routing-instances VRF-1 protocols evpn oism supplemental-bridge-domain-irb irb.4
set routing-instances VRF-1 protocols evpn oism pim-evpn-gateway
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface lo0.1

```

```

set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.2 passive
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface et-0/0/0:2.0
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface lo0.1
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.2 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface et-0/0/0:2.0
set routing-instances VRF-1 protocols pim rp static address 172.30.7.7
set routing-instances VRF-1 protocols pim rp static address 2001:0db8::172:30:7:7
set routing-instances VRF-1 protocols pim interface irb.2 distributed-dr
set routing-instances VRF-1 protocols pim interface irb.4 accept-remote-source
set routing-instances VRF-1 protocols pim interface irb.5 distributed-dr
set routing-instances VRF-1 protocols pim interface lo0.1
set routing-instances VRF-1 protocols pim interface et-0/0/0:2.0
set routing-instances VRF-1 protocols pim disable-packet-register
set routing-instances VRF-1 interface et-0/0/0:2.0
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface irb.5
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 route-distinguisher 192.168.2.1:8
set routing-instances VRF-1 vrf-target target:1:2
set routing-instances VRF-1 vrf-table-label

```

DC-1: Leaf-11 Sample Configuration

```

set system host-name Leaf-11
set routing-options router-id 192.168.3.1
set routing-options autonomous-system 65111

# Interfaces configuration

set chassis aggregated-devices ethernet device-count 10
set chassis fpc 0 pic 0 port 0 channel-speed 10g
set chassis fpc 0 pic 0 port 1 channel-speed 10g
set chassis network-services enhanced-ip
set protocols lldp port-id-subtype interface-name
set protocols lldp tlv-select system-name
set protocols lldp tlv-select system-capabilities

```

```

set protocols lldp interface all

set interfaces xe-0/0/0:0 description "CONNECTED TO CE-Host"
set interfaces xe-0/0/0:0 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_2
set interfaces xe-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_5

set interfaces xe-0/0/0:1 description "CONNECTED TO CE-Host"
set interfaces xe-0/0/0:1 ether-options 802.3ad ae0
set interfaces ae0 description "CONNECTED TO CE-Host"
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 esi df-election-type preference value 100
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 00:11:11:11:11:11
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_2
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_5

set interfaces xe-0/0/1:0 description "CONNECTED TO SPINE-1"
set interfaces xe-0/0/1:0 unit 0 family inet address 10.3.7.2/24
set interfaces xe-0/0/1:1 description "CONNECTED TO SPINE-2"
set interfaces xe-0/0/1:1 unit 0 family inet address 10.3.8.2/24

# EBGp underlay and overlay peering for EVpn

set policy-options policy-statement EXPORT-L00 term LOOPBACK from protocol direct
set policy-options policy-statement EXPORT-L00 term LOOPBACK from interface lo0.0
set policy-options policy-statement EXPORT-L00 term LOOPBACK then accept
set policy-options policy-statement EXPORT-L00 term REJECT then reject
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set routing-options forwarding-table export load-balancing-policy

set protocols bgp group IP-UNDERLAY type external
set protocols bgp group IP-UNDERLAY export EXPORT-L00
set protocols bgp group IP-UNDERLAY local-as 65103
set protocols bgp group IP-UNDERLAY multipath multiple-as
set protocols bgp group IP-UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group IP-UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group IP-UNDERLAY bfd-liveness-detection session-mode automatic
set protocols bgp group IP-UNDERLAY neighbor 10.3.7.1 peer-as 65107

```

```

set protocols bgp group IP-UNDERLAY neighbor 10.3.8.1 peer-as 65108
set protocols bgp group IP-OVERLAY type external
set protocols bgp group IP-OVERLAY multihop ttl 2
set protocols bgp group IP-OVERLAY multihop no-nexthop-change
set protocols bgp group IP-OVERLAY local-address 192.168.3.1
set protocols bgp group IP-OVERLAY family inet-vpn unicast
set protocols bgp group IP-OVERLAY family evpn signaling
set protocols bgp group IP-OVERLAY local-as 65103
set protocols bgp group IP-OVERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group IP-OVERLAY bfd-liveness-detection multiplier 3
set protocols bgp group IP-OVERLAY bfd-liveness-detection session-mode automatic
set protocols bgp group IP-OVERLAY neighbor 192.168.1.1 peer-as 65101
set protocols bgp group IP-OVERLAY neighbor 192.168.4.1 peer-as 65104
set protocols bgp group IP-OVERLAY neighbor 192.168.5.1 peer-as 65105
set protocols bgp group IP-OVERLAY neighbor 192.168.2.1 peer-as 65102

# EVPN instance with enhanced OISM - enable snooping and SMET, define tenant VLANs and OISM SBD, and their IRB interfaces

set interfaces irb mtu 1500
set interfaces irb unit 2 virtual-gateway-accept-data
set interfaces irb unit 2 family inet address 10.102.1.3/24 virtual-gateway-address 10.102.1.100
set interfaces irb unit 2 family inet6 address 2001:0db8:10:102::3/64 virtual-gateway-address 2001:0db8:10:102::100
set interfaces irb unit 2 mac 00:01:00:01:00:04
set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.104.1.3/24 virtual-gateway-address 10.104.1.100
set interfaces irb unit 4 family inet6 address 2001:0db8:10:104::3/64 virtual-gateway-address 2001:0db8:10:104::100
set interfaces irb unit 4 mac 00:01:00:03:00:04
set interfaces irb unit 5 virtual-gateway-accept-data
set interfaces irb unit 5 family inet address 10.105.1.3/24 virtual-gateway-address 10.105.1.100
set interfaces irb unit 5 family inet6 address 2001:0db8:10:105::3/64 virtual-gateway-address 2001:0db8:10:105::100
set interfaces irb unit 5 mac 00:01:00:04:00:04
set interfaces lo0 unit 0 family inet address 192.168.3.1/32 primary
set interfaces lo0 unit 1 family inet address 192.168.3.2/32
set interfaces lo0 unit 1 family inet6 address 2001:0db8::192:168:3:2/128

set forwarding-options multicast-replication evpn irb enhanced-oism

set routing-instances evpn-vxlan-A instance-type mac-vrf
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_2 proxy

```

```

set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_4 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_5 proxy
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_2
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_4
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_5
set routing-instances evpn-vxlan-A protocols evpn encapsulation vxlan
set routing-instances evpn-vxlan-A protocols evpn default-gateway no-gateway-community
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 2
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 4
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 5
set routing-instances evpn-vxlan-A vtep-source-interface lo0.0
set routing-instances evpn-vxlan-A service-type vlan-aware
set routing-instances evpn-vxlan-A interface xe-0/0/0:0.0
set routing-instances evpn-vxlan-A interface ae0.0
set routing-instances evpn-vxlan-A route-distinguisher 192.168.3.1:9
set routing-instances evpn-vxlan-A vrf-target target:9:9
set routing-instances evpn-vxlan-A vlans VLAN_2 vlan-id 2
set routing-instances evpn-vxlan-A vlans VLAN_2 l3-interface irb.2
set routing-instances evpn-vxlan-A vlans VLAN_2 vxlan vni 2
set routing-instances evpn-vxlan-A vlans VLAN_4 vlan-id 4
set routing-instances evpn-vxlan-A vlans VLAN_4 l3-interface irb.4
set routing-instances evpn-vxlan-A vlans VLAN_4 vxlan vni 4
set routing-instances evpn-vxlan-A vlans VLAN_5 vlan-id 5
set routing-instances evpn-vxlan-A vlans VLAN_5 l3-interface irb.5
set routing-instances evpn-vxlan-A vlans VLAN_5 vxlan vni 5

```

Tenant VRF VRF-1 configuration with hosted tenant VLANs

```

set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 routing-options router-id 192.168.3.2
set routing-instances VRF-1 protocols evpn oism supplemental-bridge-domain-irb irb.4
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface lo0.1
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.2 passive
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface lo0.1
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.2 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols pim passive
set routing-instances VRF-1 protocols pim interface all
set routing-instances VRF-1 protocols pim interface irb.4 accept-remote-source
set routing-instances VRF-1 interface irb.2

```



```

set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface irb.5
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 route-distinguisher 192.168.3.1:8
set routing-instances VRF-1 vrf-target target:1:2
set routing-instances VRF-1 vrf-table-label

```

DC-2: iGW-21 Sample Configuration

```

set system host-name iGW-21
set routing-options router-id 192.168.21.1
set routing-options autonomous-system 65222

# Interfaces configuration

set chassis aggregated-devices ethernet device-count 10
set chassis network-services enhanced-ip
set protocols lldp port-id-subtype interface-name
set protocols lldp tlv-select system-name
set protocols lldp tlv-select system-capabilities
set protocols lldp interface all

set interfaces et-0/0/0 number-of-sub-ports 4
set interfaces et-0/0/0 speed 10g

set interfaces et-0/0/0:0 description "CONNECTED TO CE-Host"
set interfaces et-0/0/0:0 unit 0 family ethernet-switching interface-mode trunk
set interfaces et-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_5
set interfaces et-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_3

set interfaces et-0/0/0:1 description "CONNECTED to CE-Host"
set interfaces et-0/0/0:1 ether-options 802.3ad ae0
set interfaces ae0 description "CONNECTED TO CE-Host"
set interfaces ae0 esi 00:55:55:55:55:55:55:55:55:55
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 00:55:55:55:55:55
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_5

```

```
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_3
```

```
set interfaces et-0/0/1 number-of-sub-ports 4
```

```
set interfaces et-0/0/1 speed 10g
```

```
set interfaces et-0/0/1:0 description "iGW-21 to Leaf-22"
```

```
set interfaces et-0/0/1:0 unit 0 family inet address 10.24.21.1/24
```

```
set interfaces et-0/0/1:1 description "iGW-21 to WAN Router"
```

```
set interfaces et-0/0/1:1 unit 0 family inet address 10.21.53.1/24
```

```
set interfaces et-0/0/1:3 description "iGW-21 to Leaf-21"
```

```
set interfaces et-0/0/1:3 unit 0 family inet address 10.23.21.1/24
```

```
set interfaces et-0/0/2 number-of-sub-ports 4
```

```
set interfaces et-0/0/2 speed 10g
```

```
set interfaces et-0/0/2:0 description "iGW-21 to Leaf-23"
```

```
set interfaces et-0/0/2:0 unit 0 family inet address 10.25.21.1/24
```

EBGp underlay and overlay peering for EVPN

```
set policy-options policy-statement EXPORT-L00 term LOOPBACK from protocol direct
```

```
set policy-options policy-statement EXPORT-L00 term LOOPBACK from interface lo0.0
```

```
set policy-options policy-statement EXPORT-L00 term LOOPBACK then accept
```

```
set policy-options policy-statement EXPORT-L00 term SL from route-filter 192.168.23.1/32 exact
```

```
set policy-options policy-statement EXPORT-L00 term SL from route-filter 192.168.24.1/32 exact
```

```
set policy-options policy-statement EXPORT-L00 term SL from route-filter 192.168.25.1/32 exact
```

```
set policy-options policy-statement EXPORT-L00 term SL then accept
```

```
set policy-options policy-statement EXPORT-L00 term REJECT then reject
```

```
set policy-options policy-statement load-balancing-policy then load-balance per-packet
```

```
set routing-options forwarding-table export load-balancing-policy
```

```
set protocols bgp group IP-UNDERLAY type external
```

```
set protocols bgp group IP-UNDERLAY export EXPORT-L00
```

```
set protocols bgp group IP-UNDERLAY local-as 65201
```

```
set protocols bgp group IP-UNDERLAY multipath multiple-as
```

```
set protocols bgp group IP-UNDERLAY bfd-liveness-detection minimum-interval 350
```

```
set protocols bgp group IP-UNDERLAY bfd-liveness-detection multiplier 3
```

```
set protocols bgp group IP-UNDERLAY bfd-liveness-detection session-mode automatic
```

```
set protocols bgp group IP-UNDERLAY neighbor 10.23.21.2 peer-as 65203
```

```
set protocols bgp group IP-UNDERLAY neighbor 10.24.21.2 peer-as 65204
```

```
set protocols bgp group IP-UNDERLAY neighbor 10.25.21.2 peer-as 65205
```

```
set protocols bgp group IP-OVERLAY type external
```

```
set protocols bgp group IP-OVERLAY multihop ttl 2
```

```
set protocols bgp group IP-OVERLAY multihop no-next-hop-change
```

```
set protocols bgp group IP-OVERLAY local-address 192.168.21.1
```

```

set protocols bgp group IP-OVERLAY family inet-vpn unicast
set protocols bgp group IP-OVERLAY family evpn signaling
set protocols bgp group IP-OVERLAY local-as 65201
set protocols bgp group IP-OVERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group IP-OVERLAY bfd-liveness-detection multiplier 3
set protocols bgp group IP-OVERLAY bfd-liveness-detection session-mode automatic
set protocols bgp group IP-OVERLAY neighbor 192.168.22.1 peer-as 65202
set protocols bgp group IP-OVERLAY neighbor 192.168.23.1 peer-as 65203
set protocols bgp group IP-OVERLAY neighbor 192.168.24.1 peer-as 65204
set protocols bgp group IP-OVERLAY neighbor 192.168.25.1 peer-as 65205

```

WAN overlay

```

set protocols bgp group OVERLAY-WAN type internal
set protocols bgp group OVERLAY-WAN local-address 192.168.21.1
set protocols bgp group OVERLAY-WAN family evpn signaling
set protocols bgp group OVERLAY-WAN local-as 65000
set protocols bgp group OVERLAY-WAN neighbor 192.168.50.50

```

EVPN instance with enhanced OISM - enable snooping and SMET, define tenant VLANs and OISM SBD

```

set interfaces irb mtu 1514
set interfaces irb unit 3 virtual-gateway-accept-data
set interfaces irb unit 3 family inet address 10.103.1.21/24 virtual-gateway-address 10.103.1.100
set interfaces irb unit 3 family inet6 address 2001:0db8:10:103::21/64 virtual-gateway-address 2001:0db8:10:103::100
set interfaces irb unit 3 mac 00:02:00:02:00:01
set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.104.1.21/24 virtual-gateway-address 10.104.1.100
set interfaces irb unit 4 family inet6 address 2001:0db8:10:104::21/64 virtual-gateway-address 2001:0db8:10:104::100
set interfaces irb unit 4 mac 00:02:00:03:00:01
set interfaces irb unit 5 virtual-gateway-accept-data
set interfaces irb unit 5 family inet address 10.105.1.21/24 virtual-gateway-address 10.105.1.100
set interfaces irb unit 5 family inet6 address 2001:0db8:10:105::21/64 virtual-gateway-address 2001:0db8:10:105::100
set interfaces irb unit 5 mac 00:02:00:04:00:01
set interfaces lo0 unit 0 family inet address 192.168.21.1/32 primary
set interfaces lo0 unit 1 family inet address 192.168.21.2/32
set interfaces lo0 unit 1 family inet6 address 2001:0db8::192:168:21:2/128 primary
set interfaces lo0 unit 1 family inet6 address 2001:0db8::192:168:21:2/128 preferred

set forwarding-options multicast-replication evpn irb enhanced-oism

```

```

set routing-instances evpn-vxlan-A instance-type mac-vrf
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_4 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_5 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_3 proxy
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_4
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_5
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_3
set routing-instances evpn-vxlan-A protocols evpn encapsulation vxlan
set routing-instances evpn-vxlan-A protocols evpn default-gateway no-gateway-community
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 3
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 4
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 5
set routing-instances evpn-vxlan-A vtep-source-interface lo0.0
set routing-instances evpn-vxlan-A service-type vlan-aware
set routing-instances evpn-vxlan-A interface et-0/0/0:0.0
set routing-instances evpn-vxlan-A interface ae0.0
set routing-instances evpn-vxlan-A route-distinguisher 192.168.21.1:9
set routing-instances evpn-vxlan-A vrf-target target:21:21

set routing-instances evpn-vxlan-A vlans VLAN_3 vlan-id 3
set routing-instances evpn-vxlan-A vlans VLAN_3 l3-interface irb.3
set routing-instances evpn-vxlan-A vlans VLAN_3 vxlan vni 3
set routing-instances evpn-vxlan-A vlans VLAN_4 vlan-id 4
set routing-instances evpn-vxlan-A vlans VLAN_4 l3-interface irb.4
set routing-instances evpn-vxlan-A vlans VLAN_4 vxlan vni 4
set routing-instances evpn-vxlan-A vlans VLAN_5 vlan-id 5
set routing-instances evpn-vxlan-A vlans VLAN_5 l3-interface irb.5
set routing-instances evpn-vxlan-A vlans VLAN_5 vxlan vni 5

# DCI parameters and multihoming peer designation

set routing-instances evpn-vxlan-A protocols evpn interconnect vrf-target target:11:11
set routing-instances evpn-vxlan-A protocols evpn interconnect route-distinguisher 221:100
set routing-instances evpn-vxlan-A protocols evpn interconnect esi 00:aa:bb:cc:dd:aa:bb:cc:dd:aa
set routing-instances evpn-vxlan-A protocols evpn interconnect esi all-active
set routing-instances evpn-vxlan-A protocols evpn interconnect esi df-election-type preference
value 300
set routing-instances evpn-vxlan-A protocols evpn interconnect interconnected-vni-list all

set protocols evpn interconnect-multihoming-peer-gateways 192.168.22.1

set protocols ospf area 0.0.0.0 interface lo0.0

```

```

set protocols ospf area 0.0.0.0 interface et-0/0/1:1.0

# Tenant VRF VRF-1 configuration, including OISM PEG setting and external multicast routing
parameters

set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface lo0.1
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.3 passive
set routing-instances VRF-1 protocols pim passive
set routing-instances VRF-1 protocols pim interface all
set routing-instances VRF-1 protocols pim interface irb.4 accept-remote-source
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface irb.5
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 route-distinguisher 192.168.21.1:8
set routing-instances VRF-1 vrf-target target:1:2
set routing-instances VRF-1 vrf-table-label

```

DC-2: Leaf-21 Sample Configuration

```

set system host-name Leaf-21
set routing-options router-id 192.168.23.1
set routing-options autonomous-system 65222

# Interfaces configuration

set chassis aggregated-devices ethernet device-count 10
set chassis fpc 0 pic 0 port 0 channel-speed 10g
set chassis fpc 0 pic 0 port 1 channel-speed 10g
set chassis network-services enhanced-ip
set protocols lldp port-id-subtype interface-name
set protocols lldp tlv-select system-name
set protocols lldp tlv-select system-capabilities
set protocols lldp interface all

sset interfaces xe-0/0/0:0 description "CONNECTED TO CE-Host"
set interfaces xe-0/0/0:0 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_5
set interfaces xe-0/0/0:0 unit 0 family ethernet-switching vlan members VLAN_2

```

```

set interfaces xe-0/0/0:1 description "CONNECTED TO CE-Host"
set interfaces xe-0/0/0:1 ether-options 802.3ad ae0
set interfaces ae0 description "CONNECTED TO CE-Host"
set interfaces ae0 esi 00:66:66:66:66:66:66:66:66
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 00:66:66:66:66:66
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_5
set interfaces ae0 unit 0 family ethernet-switching vlan members VLAN_2

```

```

set interfaces xe-0/0/0:3 description "CONNECTED to iGW-21"
set interfaces xe-0/0/0:3 unit 0 family inet address 10.23.21.2/24
set interfaces xe-0/0/1:0 description "CONNECTED to iGW-22"
set interfaces xe-0/0/1:0 unit 0 family inet address 10.23.22.2/24

```

EBGp underlay and overlay peering for EVpn

```

set policy-options policy-statement EXPORT-L00 term LOOPBACK from protocol direct
set policy-options policy-statement EXPORT-L00 term LOOPBACK from interface lo0.0
set policy-options policy-statement EXPORT-L00 term LOOPBACK then accept
set policy-options policy-statement EXPORT-L00 term GW from route-filter 192.168.21.1/32 exact
set policy-options policy-statement EXPORT-L00 term GW from route-filter 192.168.22.1/32 exact
set policy-options policy-statement EXPORT-L00 term GW then accept
set policy-options policy-statement EXPORT-L00 term REJECT then reject
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set routing-options forwarding-table export load-balancing-policy

```

```

set protocols bgp group IP-UNDERLAY type external
set protocols bgp group IP-UNDERLAY export EXPORT-L00
set protocols bgp group IP-UNDERLAY local-as 65203
set protocols bgp group IP-UNDERLAY multipath multiple-as
set protocols bgp group IP-UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group IP-UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group IP-UNDERLAY bfd-liveness-detection session-mode automatic
set protocols bgp group IP-UNDERLAY neighbor 10.23.21.1 peer-as 65201
set protocols bgp group IP-UNDERLAY neighbor 10.23.22.1 peer-as 65202
set protocols bgp group IP-OVERLAY type external
set protocols bgp group IP-OVERLAY multihop ttl 2
set protocols bgp group IP-OVERLAY multihop no-nexthop-change

```

```

set protocols bgp group IP-OVERLAY local-address 192.168.23.1
set protocols bgp group IP-OVERLAY family inet-vpn unicast
set protocols bgp group IP-OVERLAY family evpn signaling
set protocols bgp group IP-OVERLAY local-as 65203
set protocols bgp group IP-OVERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group IP-OVERLAY bfd-liveness-detection multiplier 3
set protocols bgp group IP-OVERLAY bfd-liveness-detection session-mode automatic
set protocols bgp group IP-OVERLAY neighbor 192.168.21.1 peer-as 65201
set protocols bgp group IP-OVERLAY neighbor 192.168.22.1 peer-as 65202
set protocols bgp group IP-OVERLAY neighbor 192.168.24.1 peer-as 65204
set protocols bgp group IP-OVERLAY neighbor 192.168.25.1 peer-as 65205

# EVPN instance with enhanced OISM - enable snooping and SMET, define tenant VLANs and OISM SBD,
and their IRB interfaces

set interfaces irb mtu 1500
set interfaces irb unit 2 virtual-gateway-accept-data
set interfaces irb unit 2 family inet address 10.102.1.23/24 virtual-gateway-address 10.102.1.100
set interfaces irb unit 2 family inet6 address 2001:0db8:10:102::23/64 virtual-gateway-address
2001:0db8:10:102::100
set interfaces irb unit 2 mac 00:02:00:01:00:04
set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.104.1.23/24 virtual-gateway-address 10.104.1.100
set interfaces irb unit 4 family inet6 address 2001:0db8:10:104::23/64 virtual-gateway-address
2001:0db8:10:104::100
set interfaces irb unit 4 mac 00:02:00:03:00:04
set interfaces irb unit 5 virtual-gateway-accept-data
set interfaces irb unit 5 family inet address 10.105.1.23/24 virtual-gateway-address 10.105.1.100
set interfaces irb unit 5 family inet6 address 2001:0db8:10:105::23/64 virtual-gateway-address
2001:0db8:10:105::100
set interfaces irb unit 5 mac 00:02:00:04:00:04
set interfaces lo0 unit 0 family inet address 192.168.23.1/32 primary
set interfaces lo0 unit 1 family inet address 192.168.23.2/32
set interfaces lo0 unit 1 family inet6 address 2001:0db8::192:168:23:2/128 primary
set interfaces lo0 unit 1 family inet6 address 2001:0db8::192:168:23:2/128 preferred

set forwarding-options multicast-replication evpn irb enhanced-oism

set routing-instances evpn-vxlan-A instance-type mac-vrf
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_4 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_5 proxy
set routing-instances evpn-vxlan-A protocols igmp-snooping vlan VLAN_2 proxy
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_4

```

```

set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_5
set routing-instances evpn-vxlan-A protocols mld-snooping vlan VLAN_2
set routing-instances evpn-vxlan-A protocols evpn encapsulation vxlan
set routing-instances evpn-vxlan-A protocols evpn default-gateway no-gateway-community
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 2
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 4
set routing-instances evpn-vxlan-A protocols evpn extended-vni-list 5
set routing-instances evpn-vxlan-A vtep-source-interface lo0.0
set routing-instances evpn-vxlan-A service-type vlan-aware
set routing-instances evpn-vxlan-A interface xe-0/0/0:0.0
set routing-instances evpn-vxlan-A interface ae0.0
set routing-instances evpn-vxlan-A route-distinguisher 192.168.23.1:9
set routing-instances evpn-vxlan-A vrf-target target:21:21
set routing-instances evpn-vxlan-A vlans VLAN_2 vlan-id 2
set routing-instances evpn-vxlan-A vlans VLAN_2 l3-interface irb.2
set routing-instances evpn-vxlan-A vlans VLAN_2 vxlan vni 2
set routing-instances evpn-vxlan-A vlans VLAN_4 vlan-id 4
set routing-instances evpn-vxlan-A vlans VLAN_4 l3-interface irb.4
set routing-instances evpn-vxlan-A vlans VLAN_4 vxlan vni 4
set routing-instances evpn-vxlan-A vlans VLAN_5 vlan-id 5
set routing-instances evpn-vxlan-A vlans VLAN_5 l3-interface irb.5
set routing-instances evpn-vxlan-A vlans VLAN_5 vxlan vni 5

```

Tenant VRF VRF-1 configuration with hosted tenant VLANs

```

set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 routing-options router-id 192.168.23.2
set routing-instances VRF-1 protocols evpn oism supplemental-bridge-domain-irb irb.4
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface lo0.1
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols ospf area 0.0.0.0 interface irb.2 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface lo0.1
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.4
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.5 passive
set routing-instances VRF-1 protocols ospf3 area 0.0.0.0 interface irb.2 passive
set routing-instances VRF-1 protocols pim passive
set routing-instances VRF-1 protocols pim interface all
set routing-instances VRF-1 protocols pim interface irb.4 accept-remote-source
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface irb.5
set routing-instances VRF-1 interface lo0.1

```



```
set routing-instances VRF-1 route-distinguisher 23.23.23.1:8
set routing-instances VRF-1 vrf-target target:1:2
set routing-instances VRF-1 vrf-table-label
```

Show Commands to Verify DCI with Enhanced OISM

We provide show commands in the CLI to verify DCI with enhanced OISM is enabled, and the iGW devices are transferring multicast traffic across the interconnection. The following sample show command outputs display results from the configuration described in ["Sample Configuration for DCI with Enhanced OISM" on page 1206](#).

1. *show multicast next-hops session <identifier session-id>*—Check the EVPN multicast next hop session information for DF status on the multihomed EVPN iGW devices that share an I-ESI. See ["Multicast Next Hops Convergence Optimization" on page 1203](#) for more on how DCI with enhanced OISM uses a session model to optimize next hop convergence in the Packet Forwarding Engine when peer iGW device DF and NDF statuses change.

For example, in our configuration, currently among the peer iGW devices in DC-1, iGW-11 is the DF and iGW-12 is the NDF. In DC-2, iGW-21 is the DF (so you can assume its peer iGW-22 is the NDF).

iGW-11:

```
user@iGW-11> show multicast next-hops session
MCNH session ID: 1
  Role: DF
  L3 VRF: VRF-1
  State: UP
  References: 2
MCNH session ID: 2
  Role: NDF
  L3 VRF: VRF-1
  State: DOWN
  References: 2
```

iGW-12:

```
user@iGW-12> show multicast next-hops session
MCNH session ID: 1
  Role: DF
  L3 VRF: VRF-1
```

```

State: DOWN
References: 2
MCNH session ID: 2
Role: NDF
L3 VRF: VRF-1
State: UP
References: 2

```

iGW-21:

```

user@iGW-21> show multicast next-hops session
MCNH session ID: 1
  Role: DF
  L3 VRF: VRF-1
  State: UP
  References: 2
MCNH session ID: 2
  Role: NDF
  L3 VRF: VRF-1
  State: DOWN
  References: 2

```

2. *show multicast next-hops list* **<identifier next-hop-id>** *show multicast next-hops list* **<identifier next-hop-id>**—Check the multicast next hops unilist entries.

For example, in our configuration:

iGW-11:

```

user@GW11> show multicast next-hops list
ID          Refcount    Kernel Table Id  Member Next-hop
9093         4             0                4087
              4085
              Flags: 0x20000819

```

iGW-12:

```

user@iGW-12> show multicast next-hops list
ID          Refcount    Kernel Table Id  Member Next-hop
9093         4             0                4097

```

```
4095
Flags: 0x20000819
```

iGW-21

```
user@iGW-21> show multicast next-hops list
ID          Refcount    Kernel Table Id  Member Next-hop
9096        4           0                4083
                                     4063
Flags: 0x20000819
```

3. *show multicast snooping next-hops list* <identifier *next-hop-id*> *show multicast snooping next-hops list* <identifier *next-hop-id*>—Check the multicast snooping (L2 multicast forwarding) next hops unilist entries.

For example, in our configuration:

iGW-11:

```
user@iGW11> show multicast snooping next-hops list
ID          Refcount    Kernel Table Id  Member Next-hop
9093        3           0                4087
                                     4085
                                     Flags: 0x28000808
7040        4           54                4073
                                     mdiscard-(7018)
                                     Flags: 0x20000819
7039        2           54                mdiscard-(7027)
                                     mdiscard-(7027)
                                     Flags: 0x20000819
7038        2           54                mdiscard-(7018)
                                     mdiscard-(7018)
                                     Flags: 0x20000819
```

iGW-12:

```
user@iGW-12> show multicast snooping next-hops list
ID          Refcount    Kernel Table Id  Member Next-hop
9093        3           0                4097
                                     4095
                                     Flags: 0x28000808
```

7039	2	54	mdiscard-(7027) mdiscard-(7027) Flags: 0x20000819	
7038	2	54	mdiscard-(7018) mdiscard-(7018) Flags: 0x200008197040	4
54	4083		mdiscard-(7018) Flags: 0x20000819	

iGW-21

```
user@iGW-21> show multicast snooping next-hops list
```

ID	Refcount	Kernel Table Id	Member Next-hop
9096	3	0	4083 4063 Flags: 0x28000808
7043	6	54	4081 mdiscard-(7018) Flags: 0x20000819
7036	2	54	4035 mdiscard-(7018) Flags: 0x20000819
7037	2	54	4037 mdiscard-(7027) Flags: 0x20000819

4. *show evpn oism dci* <l3-context l3-vrf-name> <extensive> *show evpn oism dci* <l3-context l3-vrf-name> <extensive>—Verify DCI is enabled with enhanced OISM, and check the multihoming peer I-ESI value and I-ESI DF or NDF status for DCI iGW peer devices.

For example, in our configuration:

iGW-11:

```
user@iGW-11> show evpn oism dci
```

L3 context	OISM-Mode	DCI status
VRF-1	Enhanced OISM	Enabled

```
user@iGW-11> show evpn oism dci extensive
```

EVPN L3 context: VRF-1
OISM Mode: Enhanced OISM

```

DCI status: Enabled
I-ESI: 00:0a:0b:0c:0d:0a:0b:0c:0d:0a
I-ESI status: DF
Designated forwarder: 192.168.1.1
    I-ESI Session ID: 1
    I-ESI Session state: UP
Backup forwarder: 192.168.2.1
    I-ESI Session ID: 2
    I-ESI Session state: DOWN

```

iGW-12:

```

user@iGW-12> show evpn oism dci
L3 context          OISM-Mode          DCI status
VRF-1               Enhanced OISM      Enabled

user@iGW-12> show evpn oism dci extensive
EVPN L3 context: VRF-1
    OISM Mode: Enhanced OISM
    DCI status: Enabled
    I-ESI: 00:0a:0b:0c:0d:0a:0b:0c:0d:0a
    I-ESI status: Non-DF
    Designated forwarder: 192.168.1.1
        I-ESI Session ID: 1
        I-ESI Session state: DOWN
    Backup forwarder: 192.168.2.1
        I-ESI Session ID: 2
        I-ESI Session state: UP

```

Leaf-11:

```

user@Leaf-11> show evpn oism dci
L3 context          OISM-Mode          DCI status
VRF-1               Enhanced OISM      Disabled

user@Leaf-11> show evpn oism dci extensive
EVPN L3 context: VRF-1
    OISM Mode: Enhanced OISM
    DCI status: Disabled

```

iGW-21:

```

user@iGW-21> show evpn oism dci
L3 context          OISM-Mode          DCI status
VRF-1               Enhanced OISM      Enabled

user@iGW-21> show evpn oism dci extensive
EVPN L3 context: VRF-1
  OISM Mode: Enhanced OISM
  DCI status: Enabled
  I-ESI: 00:aa:bb:cc:dd:aa:bb:cc:dd:aa
  I-ESI status: DF
  Designated forwarder: 192.168.21.1
    I-ESI Session ID: 1
    I-ESI Session state: UP
  Backup forwarder: 192.168.22.1
    I-ESI Session ID: 2
    I-ESI Session state: DOWN

```

Leaf-21:

```

user@Leaf-21> show evpn oism dci
L3 context          OISM-Mode          DCI status
VRF-1               Enhanced OISM      Disabled

user@Leaf-21> show evpn oism dci extensive
EVPN L3 context: VRF-1
  OISM Mode: Enhanced OISM
  DCI status: Disabled

```

5. `show evpn oism <l3-context l3-vrf-name> <extensive>` `show evpn oism <l3-context l3-vrf-name> <extensive>`—Verify device OISM status, including that DC-1 and DC-2 have the same OISM SBD configured and both are in enhanced OISM mode. When the device has DCI status Enabled, that means the device is also an iGW device with enhanced OISM. Add the `l3-context` option to display information only for a specific L3 VRF. Otherwise, the command displays information for all OISM-enabled VRFs instances.

For example, in our configuration:

iGW-11:

```

user@iGW-11> show evpn oism
L3 context          SBD      PEG-DF-ELECTION
VRF-1               irb.4    Disabled

user@iGW-11> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Disabled
  OISM Mode: Enhanced OISM
  DCI status: Enabled

```

iGW-12:

```

user@iGW-12> show evpn oism
L3 context          SBD      PEG-DF-ELECTION
VRF-1               irb.4    Disabled

user@iGW-12> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Disabled
  OISM Mode: Enhanced OISM
  DCI status: Enabled

```

Leaf-11:

```

user@Leaf-11> show evpn oism
L3 context          SBD      PEG-DF-ELECTION
VRF-1               irb.4    Disabled

user@Leaf-11> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Disabled
  OISM Mode: Enhanced OISM
  DCI status: Disabled

```

iGW-21:

```
user@iGW-21> show evpn oism
L3 context          SBD      PEG-DF-ELECTION
VRF-1               irb.4    Disabled

user@iGW-21> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Disabled
  OISM Mode: Enhanced OISM
  DCI status: Enabled
```

Leaf-21:

```
user@Leaf-21> show evpn oism
L3 context          SBD      PEG-DF-ELECTION
VRF-1               irb.4    Disabled

{master:0}
user@Leaf-21> show evpn oism extensive
EVPN L3 context: VRF-1
  OISM SBD interface: irb.4
  PEG DF Election: Disabled
  OISM Mode: Enhanced OISM
  DCI status: Disabled
```

6. ***show evpn l3-context l3-vrf-name extensive*** `show evpn l3-context l3-vrf-name extensive`—Verify information about an L3 VRF corresponding to an EVPN instance, including the configured multicast mode and RD associated with the L3 VRF. EVPN Multicast Routing mode output field value is Enhanced OISM for enhanced OISM mode. All PE devices in the EVPN instance in each data center must use enhanced OISM for this mode to work.

For example, in our configuration:

iGW-11:

```
user@iGW-11> show evpn l3-context VRF-1 extensive
L3 context: VRF-1
  Type: Configured
  Route Distinguisher: 192.168.1.1:8
```



```
Reference count: 4  
EVPN Multicast Routing mode: Enhanced OISM
```

iGW-12:

```
user@iGW-12> show evpn l3-context VRF-1 extensive  
L3 context: VRF-1  
Type: Configured  
Route Distinguisher: 192.168.2.1:8  
Reference count: 4  
EVPN Multicast Routing mode: Enhanced OISM
```

Leaf-11:

```
user@Leaf-11> show evpn l3-context VRF-1 extensive  
L3 context: VRF-1  
Type: Configured  
Route Distinguisher: 192.168.3.1:8  
Reference count: 4  
EVPN Multicast Routing mode: Enhanced OISM
```

iGW-21:

```
user@iGW-21> show evpn l3-context VRF-1 extensive  
L3 context: VRF-1  
Type: Configured  
Route Distinguisher: 192.168.21.1:8  
Reference count: 4  
EVPN Multicast Routing mode: Enhanced OISM
```

Leaf-21:

```
user@Leaf-21> show evpn l3-context VRF-1 extensive  
L3 context: VRF-1  
Type: Configured  
Route Distinguisher: 192.168.23.1:8  
Reference count: 4  
EVPN Multicast Routing mode: Enhanced OISM
```

7. *show multicast route* **<extensive>**—Display includes details about DCI DF and NDF multicast next-hop unicast entries.

show multicast route **<instance** *l3-vrf-name* **<extensive>**—Verify the multicast routes for the applicable multicast groups and sources, including the DCI DF upstream and downstream interfaces. The traffic flows from the source toward the network on the upstream interfaces, and from the network toward the receivers on the downstream interfaces. On iGW devices, the output also shows the DCI NDF upstream and downstream interfaces.



NOTE: You can configure enhanced OISM to forward multicast traffic for particular flows on the source VLAN instead of the SBD to avoid time-to-live (TTL) expiration issues for TTL=1 packets. When you enable that behavior, this command displays the following in the Next-hop ID output field for a multicast route:

- A multicast unicast next hop when the interface is a revenue VLAN IRB interface on an iGW device.
- A normal multicast next hop otherwise.

You configure that behavior using the *forward-on-source-bridge-domain* option. In the configuration we show here, we don't have *forward-on-source-bridge-domain* enabled.

For example, in our configuration:

iGW-11 (peer iGW-12 output is similar):

```
user@GW11> show multicast route instance VRF-1
Instance: VRF-1 Family: INET

Group: 233.252.0.1
  Source: 172.20.1.15/32
  Upstream interface: irb.4 (L2 NH: 4073)
  EVPN DCI NDF upstream interface: irb.4 (L2 NH: 7018)
  Downstream interface list:
    irb.2 (L2 NH: 4083) et-0/0/0:2.0
  EVPN DCI NDF downstream interface list:
    irb.2 (L2 NH: 4083) et-0/0/0:2.0

Group: 233.252.0.2
  Source: 172.20.1.15/32
  Upstream interface: irb.4 (L2 NH: 4073)
  EVPN DCI NDF upstream interface: irb.4 (L2 NH: 7018)
```

Downstream interface list:

irb.2 (L2 NH: 4083) et-0/0/0:2.0

EVPN DCI NDF downstream interface list:

irb.2 (L2 NH: 4083) et-0/0/0:2.0

Instance: VRF-1 Family: INET6

user@igw-11> **show multicast route instance VRF-1 extensive**

Instance: VRF-1 Family: INET

Group: 233.252.0.1

Source: 172.20.1.15/32

Upstream interface: irb.4 (L2 NH: 4073)

EVPN DCI NDF upstream interface: irb.4 (L2 NH: 7018)

Downstream interface list:

irb.2 (L2 NH: 4083) et-0/0/0:2.0

Number of outgoing interfaces: 2

EVPN DCI NDF downstream interface list:

irb.2 (L2 NH: 4083) et-0/0/0:2.0

Number of EVPN DCI NDF outgoing interfaces: 2

Session description: Unknown

Statistics: 523 kBps, 500 pps, 131589943 packets

Next-hop ID: 9093 (Unilist NH)

Upstream protocol: Multicast

Route state: Active

Forwarding state: Forwarding

Cache lifetime/timeout: 360 seconds

Wrong incoming interface notifications: 1

Uptime: 3d 01:06:21

Sensor ID: 0xf0000005

Group: 233.252.0.2

Source: 172.20.1.15/32

Upstream interface: irb.4 (L2 NH: 4073)

EVPN DCI NDF upstream interface: irb.4 (L2 NH: 7018)

Downstream interface list:

irb.2 (L2 NH: 4083) et-0/0/0:2.0

Number of outgoing interfaces: 2

EVPN DCI NDF downstream interface list:

irb.2 (L2 NH: 4083) et-0/0/0:2.0

Number of EVPN DCI NDF outgoing interfaces: 2

Session description: Unknown

Statistics: 523 kBps, 500 pps, 131589942 packets

```

Next-hop ID: 9093 (Unilist NH)
Upstream protocol: Multicast
Route state: Active
Forwarding state: Forwarding
Cache lifetime/timeout: 360 seconds
Wrong incoming interface notifications: 1
Uptime: 3d 01:06:21
Sensor ID: 0xf0000004

```

```
Instance: VRF-1 Family: INET6
```

Leaf-11:

```
user@Leaf-11> show multicast route instance VRF-1
```

```
Instance: VRF-1 Family: INET
```

```
Group: 233.252.0.1
```

```
Source: 172.20.1.15/32
```

```
Upstream interface: irb.2
```

```
Downstream interface list:
```

```
    irb.4
```

```
Group: 233.252.0.2
```

```
Source: 172.20.1.15/32
```

```
Upstream interface: irb.2
```

```
Downstream interface list:
```

```
    irb.4
```

```
Instance: VRF-1 Family: INET6
```

```
user@Leaf-11> show multicast route instance VRF-1 extensive
```

```
Instance: VRF-1 Family: INET
```

```
Group: 233.252.0.1
```

```
Source: 172.20.1.15/32
```

```
Upstream interface: irb.2
```

```
Downstream interface list:
```

```
    irb.4
```

```
Number of outgoing interfaces: 1
```

```
Session description: Unknown
```

```
Statistics: 511 kbps, 488 pps, 3734273212 packets
```

```
Next-hop ID: 524290
```

```

Upstream protocol: PIM
Route state: Active
Forwarding state: Forwarding
Cache lifetime/timeout: 360 seconds
Wrong incoming interface notifications: 1
Uptime: 5d 04:20:39

```

```

Group: 233.252.0.2
Source: 172.20.1.15/32
Upstream interface: irb.2
Downstream interface list:
    irb.4
Number of outgoing interfaces: 1
Session description: Unknown
Statistics: 488 kbps, 488 pps, 3734272613 packets
Next-hop ID: 524290
Upstream protocol: PIM
Route state: Active
Forwarding state: Forwarding
Cache lifetime/timeout: 360 seconds
Wrong incoming interface notifications: 0
Uptime: 5d 04:20:39

```

```

Instance: VRF-1 Family: INET6

```

iGW-21:

```

user@iGW-21> show multicast route instance VRF-1
Instance: VRF-1 Family: INET

Group: 233.252.0.1
Source: 172.20.1.15/32
Upstream interface: irb.4 (L2 NH: 4081)
EVPN DCI NDF upstream interface: irb.4 (L2 NH: 7018)
Downstream interface list:
    irb.3 (L2 NH: 4059)
EVPN DCI NDF downstream interface list:
    irb.3 (L2 NH: 4059)

Group: 233.252.0.2
Source: 172.20.1.15/32
Upstream interface: irb.4 (L2 NH: 4081)

```

EVPN DCI NDF upstream interface: irb.4 (L2 NH: 7018)

Downstream interface list:

irb.3 (L2 NH: 4059)

EVPN DCI NDF downstream interface list:

irb.3 (L2 NH: 4059)

Instance: VRF-1 Family: INET6

user@iGW-21> **show multicast route instance VRF-1 extensive**

Instance: VRF-1 Family: INET

Group: 233.252.0.1

Source: 172.20.1.15/32

Upstream interface: irb.4 (L2 NH: 4081)

EVPN DCI NDF upstream interface: irb.4 (L2 NH: 7018)

Downstream interface list:

irb.3 (L2 NH: 4059)

Number of outgoing interfaces: 1

EVPN DCI NDF downstream interface list:

irb.3 (L2 NH: 4059)

Number of EVPN DCI NDF outgoing interfaces: 1

Session description: Unknown

Statistics: 523 kBps, 500 pps, 213974110 packets

Next-hop ID: 9096 (Unilist NH)

Upstream protocol: Multicast

Route state: Active

Forwarding state: Forwarding

Cache lifetime/timeout: 360 seconds

Wrong incoming interface notifications: 1

Uptime: 4d 22:54:58

Sensor ID: 0xf0000005

Group: 233.252.0.2

Source: 172.20.1.15/32

Upstream interface: irb.4 (L2 NH: 4081)

EVPN DCI NDF upstream interface: irb.4 (L2 NH: 7018)

Downstream interface list:

irb.3 (L2 NH: 4059)

Number of outgoing interfaces: 1

EVPN DCI NDF downstream interface list:

irb.3 (L2 NH: 4059)

Number of EVPN DCI NDF outgoing interfaces: 1

Session description: Unknown

```

Statistics: 523 kbps, 500 pps, 213974109 packets
Next-hop ID: 9096 (Unicast NH)
Upstream protocol: Multicast
Route state: Active
Forwarding state: Forwarding
Cache lifetime/timeout: 360 seconds
Wrong incoming interface notifications: 1
Uptime: 4d 22:54:58
Sensor ID: 0xf0000004

```

Instance: VRF-1 Family: INET6

Leaf-21:

```
user@Leaf-21> show multicast route instance VRF-1
```

Instance: VRF-1 Family: INET

Group: 233.252.0.1

Source: 172.20.1.15/32

Upstream interface: irb.4

Downstream interface list:
irb.2

Group: 233.252.0.2

Source: 172.20.1.15/32

Upstream interface: irb.4

Downstream interface list:
irb.2

Instance: VRF-1 Family: INET6

```
user@Leaf-21> show multicast route instance VRF-1 extensive
```

Instance: VRF-1 Family: INET

Group: 233.252.0.1

Source: 172.20.1.15/32

Upstream interface: irb.4

Downstream interface list:
irb.2

Number of outgoing interfaces: 1

Session description: Unknown

Statistics: 530 kbps, 507 pps, 3891371111 packets

```

Next-hop ID: 524306
Upstream protocol: PIM
Route state: Active
Forwarding state: Forwarding
Cache lifetime/timeout: 360 seconds
Wrong incoming interface notifications: 0
Uptime: 5d 04:20:13

Group: 233.252.0.2
Source: 172.20.1.15/32
Upstream interface: irb.4
Downstream interface list:
    irb.2
Number of outgoing interfaces: 1
Session description: Unknown
Statistics: 530 kbps, 507 pps, 3891338808 packets
Next-hop ID: 524306
Upstream protocol: PIM
Route state: Active
Forwarding state: Forwarding
Cache lifetime/timeout: 360 seconds
Wrong incoming interface notifications: 0
Uptime: 5d 04:20:13

Instance: VRF-1 Family: INET6

```

8. `show route table evpn-instance-table-name match-prefix 6:*...`—Verify EVPN type 6 route control flow across the interconnect when a remote OISM leaf device receives an IGMP subscription to a multicast group (as [Figure 115 on page 1194](#) illustrates). We show propagation of the EVPN Type 6 route from Leaf-21 through iGW-21 (and iGW-22) in DC-2, across the WAN to iGW-11 (and iGW-12) and reaching Leaf-11 in DC-1.
 - a. Verify Leaf-21 in DC-2 (192.168.23.1) advertises an EVPN Type 6 route upon receiving an IGMP report from its host for multicast group 233.252.0.1 to the OISM PE devices in DC-2 (iGW-21, iGW-22, Leaf-22, and Leaf-23):

```

user@Leaf-21> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*233.252.0.1*

evpn-vxlan-A.evpn.0: 257 destinations, 935 routes (257 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.21.1:9::4::233.252.0.1::192.168.21.1/520

```



```

*[BGP/170] 3d 07:08:13, localpref 100, from 192.168.21.1
  AS path: 65201 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
[BGP/170] 3d 07:08:10, localpref 100, from 192.168.22.1
  AS path: 65202 65000 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
[BGP/170] 3d 07:08:06, localpref 100, from 192.168.24.1
  AS path: 65204 65201 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
[BGP/170] 3d 07:07:54, localpref 100, from 192.168.25.1
  AS path: 65205 65201 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
6:192.168.22.1:9::4::233.252.0.1::192.168.22.1/520
*[BGP/170] 3d 07:08:10, localpref 100, from 192.168.22.1
  AS path: 65202 I, validation-state: unverified
> to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:08:13, localpref 100, from 192.168.21.1
  AS path: 65201 65000 I, validation-state: unverified
> to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:08:06, localpref 100, from 192.168.24.1
  AS path: 65204 65202 I, validation-state: unverified
> to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:07:54, localpref 100, from 192.168.25.1
  AS path: 65205 65202 I, validation-state: unverified
> to 10.23.22.1 via xe-0/0/1:0.0
6:192.168.23.1:9::4::233.252.0.1::192.168.23.1/520
*[EVPN/170] 3d 07:08:01
  Indirect
6:192.168.24.1:9::4::233.252.0.1::192.168.24.1/520
*[BGP/170] 3d 07:08:06, localpref 100, from 192.168.24.1
  AS path: 65204 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
  to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:08:13, localpref 100, from 192.168.21.1
  AS path: 65201 65204 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
  to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:08:10, localpref 100, from 192.168.22.1
  AS path: 65202 65204 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
  to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:07:54, localpref 100, from 192.168.25.1
  AS path: 65205 65204 I, validation-state: unverified

```

```

> to 10.23.21.1 via xe-0/0/0:3.0
  to 10.23.22.1 via xe-0/0/1:0.0
6:192.168.25.1:9::4::233.252.0.1::192.168.25.1/520
*[BGP/170] 3d 07:07:54, localpref 100, from 192.168.25.1
  AS path: 65205 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
  to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:08:13, localpref 100, from 192.168.21.1
  AS path: 65201 65205 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
  to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:08:10, localpref 100, from 192.168.22.1
  AS path: 65202 65205 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
  to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 07:08:06, localpref 100, from 192.168.24.1
  AS path: 65204 65205 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
  to 10.23.22.1 via xe-0/0/1:0.0

```

- b. Verify iGW-21 in DC-2 (192.168.21.1) receives the EVPN Type 6 routes from Leaf-21, and re-originates the advertisement with its interconnect RD (221:100) across the WAN (WAN router 192.168.50.50) toward DC-1:

```

user@iGW-21> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*233.252.0.1*

evpn-vxlan-A.evpn.0: 571 destinations, 1796 routes (566 active, 0 holddown, 317 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.21.1:9::4::233.252.0.1::192.168.21.1/520
*[EVPN/170] 3d 07:19:03
  Indirect
[EVPN/170] 3d 07:18:32
  Indirect
6:192.168.22.1:9::4::233.252.0.1::192.168.22.1/520
*[BGP/170] 3d 07:19:02, localpref 100, from 192.168.50.50
  AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 07:19:02, localpref 100, from 192.168.22.1
  AS path: 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 07:15:34, localpref 100, from 192.168.23.1

```

```

        AS path: 65203 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 07:19:02, localpref 100, from 192.168.24.1
        AS path: 65204 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 07:19:02, localpref 100, from 192.168.25.1
        AS path: 65205 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:192.168.23.1:9::4::233.252.0.1::192.168.23.1/520
*[BGP/170] 3d 07:15:25, localpref 100, from 192.168.23.1
        AS path: 65203 I, validation-state: unverified
> to 10.23.21.2 via et-0/0/1:3.0
[BGP/170] 3d 07:15:25, localpref 100, from 192.168.22.1
        AS path: 65202 65203 I, validation-state: unverified
> to 10.23.21.2 via et-0/0/1:3.0
[BGP/170] 3d 07:15:25, localpref 100, from 192.168.24.1
        AS path: 65204 65203 I, validation-state: unverified
> to 10.23.21.2 via et-0/0/1:3.0
[BGP/170] 3d 07:15:18, localpref 100, from 192.168.25.1
        AS path: 65205 65203 I, validation-state: unverified
> to 10.23.21.2 via et-0/0/1:3.0
6:192.168.24.1:9::4::233.252.0.1::192.168.24.1/520
*[BGP/170] 3d 07:18:58, localpref 100, from 192.168.24.1
        AS path: 65204 I, validation-state: unverified
> to 10.24.21.2 via et-0/0/1:0.0
[BGP/170] 3d 07:18:58, localpref 100, from 192.168.22.1
        AS path: 65202 65204 I, validation-state: unverified
> to 10.24.21.2 via et-0/0/1:0.0
[BGP/170] 3d 07:15:30, localpref 100, from 192.168.23.1
        AS path: 65203 65204 I, validation-state: unverified
> to 10.24.21.2 via et-0/0/1:0.0
[BGP/170] 3d 07:18:58, localpref 100, from 192.168.25.1
        AS path: 65205 65204 I, validation-state: unverified
> to 10.24.21.2 via et-0/0/1:0.0
6:192.168.25.1:9::4::233.252.0.1::192.168.25.1/520
*[BGP/170] 3d 07:18:17, localpref 100, from 192.168.25.1
        AS path: 65205 I, validation-state: unverified
> to 10.25.21.2 via et-0/0/2:0.0
[BGP/170] 3d 07:18:16, localpref 100, from 192.168.22.1
        AS path: 65202 65205 I, validation-state: unverified
> to 10.25.21.2 via et-0/0/2:0.0
[BGP/170] 3d 07:15:18, localpref 100, from 192.168.23.1
        AS path: 65203 65205 I, validation-state: unverified

```

```

> to 10.25.21.2 via et-0/0/2:0.0
[BGP/170] 3d 07:18:16, localpref 100, from 192.168.24.1
AS path: 65204 65205 I, validation-state: unverified
> to 10.25.21.2 via et-0/0/2:0.0
6:111:100::4::233.252.0.1::192.168.1.1/520
*[BGP/170] 3d 07:19:03, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:112:100::4::233.252.0.1::192.168.2.1/520
*[BGP/170] 3d 07:19:03, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:221:100::4::233.252.0.1::192.168.21.1/520
*[EVPN/170] 3d 07:18:58
Indirect
6:222:100::4::233.252.0.1::192.168.22.1/520
*[BGP/170] 3d 07:18:58, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 07:18:58, localpref 100, from 192.168.22.1
AS path: 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 07:15:34, localpref 100, from 192.168.23.1
AS path: 65203 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 07:18:57, localpref 100, from 192.168.24.1
AS path: 65204 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 07:18:57, localpref 100, from 192.168.25.1
AS path: 65205 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0

```

- c. Verify iGW-11 in DC-1 (192.168.1.1) receives the EVPN type 6 route advertisement from the WAN (WAN router 192.168.50.50), and re-originates the advertisement with the DC-1 RD (192.168.1.1:9) toward the OISM PE devices in DC-1:

```

user@iGW-11> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*233.252.0.1*

evpn-vxlan-A.evpn.0: 586 destinations, 1883 routes (581 active, 0 holddown, 312 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.1.1:9::4::233.252.0.1::192.168.1.1/520

```

```

*[EVPN/170] 3d 07:24:12
    Indirect
[EVPN/170] 3d 07:24:12
    Indirect
6:192.168.2.1:9::4::233.252.0.1::192.168.2.1/520
*[BGP/170] 3d 07:24:12, localpref 100, from 192.168.50.50
    AS path: I, validation-state: unverified
> to 10.1.51.2 via et-0/0/1:2.0
[BGP/170] 3d 07:24:12, localpref 100, from 192.168.2.1
    AS path: 65102 I, validation-state: unverified
> to 10.1.51.2 via et-0/0/1:2.0
[BGP/170] 3d 07:24:12, localpref 100, from 192.168.3.1
    AS path: 65103 65102 I, validation-state: unverified
> to 10.1.51.2 via et-0/0/1:2.0
[BGP/170] 3d 07:24:12, localpref 100, from 192.168.4.1
    AS path: 65104 65102 I, validation-state: unverified
> to 10.1.51.2 via et-0/0/1:2.0
[BGP/170] 3d 07:24:12, localpref 100, from 192.168.5.1
    AS path: 65105 65102 I, validation-state: unverified
> to 10.1.51.2 via et-0/0/1:2.0
6:192.168.3.1:9::4::233.252.0.1::192.168.3.1/520
*[BGP/170] 3d 07:24:18, localpref 100, from 192.168.3.1
    AS path: 65103 I, validation-state: unverified
> to 10.1.7.1 via ae1.0
    to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.2.1
    AS path: 65102 65103 I, validation-state: unverified
> to 10.1.7.1 via ae1.0
    to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.4.1
    AS path: 65104 65103 I, validation-state: unverified
> to 10.1.7.1 via ae1.0
    to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.5.1
    AS path: 65105 65103 I, validation-state: unverified
> to 10.1.7.1 via ae1.0
    to 10.1.8.1 via et-0/0/1:1.0
6:192.168.4.1:9::4::233.252.0.1::192.168.4.1/520
*[BGP/170] 3d 07:24:17, localpref 100, from 192.168.4.1
    AS path: 65104 I, validation-state: unverified
> to 10.1.7.1 via ae1.0
    to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.2.1

```

```

        AS path: 65102 65104 I, validation-state: unverified
    > to 10.1.7.1 via ae1.0
        to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.3.1
        AS path: 65103 65104 I, validation-state: unverified
    > to 10.1.7.1 via ae1.0
        to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.5.1
        AS path: 65105 65104 I, validation-state: unverified
    > to 10.1.7.1 via ae1.0
        to 10.1.8.1 via et-0/0/1:1.0
6:192.168.5.1:9::4::233.252.0.1::192.168.5.1/520
*[BGP/170] 3d 07:23:38, localpref 100, from 192.168.5.1
        AS path: 65105 I, validation-state: unverified
    > to 10.1.7.1 via ae1.0
        to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:23:38, localpref 100, from 192.168.2.1
        AS path: 65102 65105 I, validation-state: unverified
    > to 10.1.7.1 via ae1.0
        to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:23:38, localpref 100, from 192.168.3.1
        AS path: 65103 65105 I, validation-state: unverified
    > to 10.1.7.1 via ae1.0
        to 10.1.8.1 via et-0/0/1:1.0
[BGP/170] 3d 07:23:38, localpref 100, from 192.168.4.1
        AS path: 65104 65105 I, validation-state: unverified
    > to 10.1.7.1 via ae1.0
        to 10.1.8.1 via et-0/0/1:1.0
6:111:100::4::233.252.0.1::192.168.1.1/520
*[EVPN/170] 3d 07:24:18
    Indirect
6:112:100::4::233.252.0.1::192.168.2.1/520
*[BGP/170] 3d 07:24:17, localpref 100, from 192.168.50.50
        AS path: I, validation-state: unverified
    > to 10.1.51.2 via et-0/0/1:2.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.2.1
        AS path: 65102 I, validation-state: unverified
    > to 10.1.51.2 via et-0/0/1:2.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.3.1
        AS path: 65103 65102 I, validation-state: unverified
    > to 10.1.51.2 via et-0/0/1:2.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.4.1
        AS path: 65104 65102 I, validation-state: unverified

```

```

> to 10.1.51.2 via et-0/0/1:2.0
[BGP/170] 3d 07:24:17, localpref 100, from 192.168.5.1
AS path: 65105 65102 I, validation-state: unverified
> to 10.1.51.2 via et-0/0/1:2.0
6:221:100::4::233.252.0.1::192.168.21.1/520
*[BGP/170] 3d 07:24:12, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.1.51.2 via et-0/0/1:2.0
6:222:100::4::233.252.0.1::192.168.22.1/520
*[BGP/170] 3d 07:24:12, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.1.51.2 via et-0/0/1:2.0

```

- d. Verify Leaf-11 in DC-1 (192.168.3.1) receives the EVPN Type 6 route(s) from the iGW devices in DC-1 (192.168.1.1, 192.168.2.1):

```

user@Leaf-11> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*233.252.0.1*

evpn-vxlan-A.evpn.0: 272 destinations, 986 routes (272 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.1.1:9::4::233.252.0.1::192.168.1.1/520
*[BGP/170] 3d 07:38:14, localpref 100, from 192.168.1.1
AS path: 65101 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:13, localpref 100, from 192.168.2.1
AS path: 65102 65000 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:13, localpref 100, from 192.168.4.1
AS path: 65104 65101 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:13, localpref 100, from 192.168.5.1
AS path: 65105 65101 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
to 10.3.8.1 via xe-0/0/1:1.0
6:192.168.2.1:9::4::233.252.0.1::192.168.2.1/520
*[BGP/170] 3d 07:38:14, localpref 100, from 192.168.2.1
AS path: 65102 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0

```

```

        to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:13, localpref 100, from 192.168.1.1
    AS path: 65101 65000 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:14, localpref 100, from 192.168.4.1
    AS path: 65104 65102 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:14, localpref 100, from 192.168.5.1
    AS path: 65105 65102 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
6:192.168.3.1:9::4::233.252.0.1::192.168.3.1/520
*[EVPN/170] 3d 07:38:19
    Indirect
6:192.168.4.1:9::4::233.252.0.1::192.168.4.1/520
*[BGP/170] 3d 07:38:19, localpref 100, from 192.168.4.1
    AS path: 65104 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:19, localpref 100, from 192.168.1.1
    AS path: 65101 65104 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:19, localpref 100, from 192.168.2.1
    AS path: 65102 65104 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:38:19, localpref 100, from 192.168.5.1
    AS path: 65105 65104 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
6:192.168.5.1:9::4::233.252.0.1::192.168.5.1/520
*[BGP/170] 3d 07:37:40, localpref 100, from 192.168.5.1
    AS path: 65105 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:37:39, localpref 100, from 192.168.1.1
    AS path: 65101 65105 I, validation-state: unverified
>   to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:37:39, localpref 100, from 192.168.2.1

```



```

AS path: 65102 65105 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
  to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 07:37:39, localpref 100, from 192.168.4.1
AS path: 65104 65105 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
  to 10.3.8.1 via xe-0/0/1:1.0

```

9. `show route table evpn-instance-table-name match-prefix 6:*0.0.0.0*`—Verify an OISM PEG device advertises its PEG role in EVPN type 6 route advertisements across the interconnection (as [Figure 119 on page 1202](#) illustrates). The PEG device advertises the EVPN Type 6 route as a multicast route applicable for all multicast groups and multicast sources (*,*), so with this command we match on output for multicast group address 0.0.0.0 in addition to the device's route distinguisher.
- a. Verify iGW-11 in DC-1 (192.168.1.1) advertises its PEG role in DC-1 and across the WAN by originating EVPN Type 6 routes using its DC-1 RD (192.168.1.1:9) and its interconnect RD (111:100):

```

user@iGW-11> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*0.0.0.0*192.168.1.1*

evpn-vxlan-A.evpn.0: 586 destinations, 1883 routes (581 active, 0 holddown, 312 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.1.1:9::2::0.0.0.0::192.168.1.1/520
    *[EVPN/170] 3d 08:15:17
    Indirect
6:192.168.1.1:9::4::0.0.0.0::192.168.1.1/520
    *[EVPN/170] 3d 08:15:17
    Indirect
6:192.168.1.1:9::5::0.0.0.0::192.168.1.1/520
    *[EVPN/170] 3d 08:15:17
    Indirect
6:111:100::2::0.0.0.0::192.168.1.1/520
    *[EVPN/170] 3d 08:15:17
    Indirect
6:111:100::4::0.0.0.0::192.168.1.1/520
    *[EVPN/170] 3d 08:15:17
    Indirect
6:111:100::5::0.0.0.0::192.168.1.1/520
    *[EVPN/170] 3d 08:15:17
    Indirect

```

- b. Verify Leaf-11 in DC-1 (192.168.3.1) receives the EVPN Type 6 route with the PEG role advertisement from iGW-11 (192.168.1.1):

```

user@Leaf-11> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*0.0.0.0*192.168.1.1*

evpn-vxlan-A.evpn.0: 272 destinations, 986 routes (272 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.1.1:9::2::0.0.0.0::192.168.1.1/520
    *[BGP/170] 3d 08:17:53, localpref 100, from 192.168.1.1
        AS path: 65101 I, validation-state: unverified
    > to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
    [BGP/170] 3d 08:17:25, localpref 100, from 192.168.2.1
        AS path: 65102 65000 I, validation-state: unverified
    > to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
    [BGP/170] 3d 08:17:41, localpref 100, from 192.168.4.1
        AS path: 65104 65101 I, validation-state: unverified
    > to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
    [BGP/170] 3d 08:16:58, localpref 100, from 192.168.5.1
        AS path: 65105 65101 I, validation-state: unverified
    > to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0

6:192.168.1.1:9::4::0.0.0.0::192.168.1.1/520
    *[BGP/170] 3d 08:17:53, localpref 100, from 192.168.1.1
        AS path: 65101 I, validation-state: unverified
    > to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
    [BGP/170] 3d 08:17:25, localpref 100, from 192.168.2.1
        AS path: 65102 65000 I, validation-state: unverified
    > to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
    [BGP/170] 3d 08:17:41, localpref 100, from 192.168.4.1
        AS path: 65104 65101 I, validation-state: unverified
    > to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0
    [BGP/170] 3d 08:16:58, localpref 100, from 192.168.5.1
        AS path: 65105 65101 I, validation-state: unverified
    > to 10.3.7.1 via xe-0/0/1:0.0
        to 10.3.8.1 via xe-0/0/1:1.0

```

```

6:192.168.1.1:9::5::0.0.0.0::192.168.1.1/520
*[BGP/170] 3d 08:17:53, localpref 100, from 192.168.1.1
  AS path: 65101 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
  to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 08:17:25, localpref 100, from 192.168.2.1
  AS path: 65102 65000 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
  to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 08:17:41, localpref 100, from 192.168.4.1
  AS path: 65104 65101 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
  to 10.3.8.1 via xe-0/0/1:1.0
[BGP/170] 3d 08:16:58, localpref 100, from 192.168.5.1
  AS path: 65105 65101 I, validation-state: unverified
> to 10.3.7.1 via xe-0/0/1:0.0
  to 10.3.8.1 via xe-0/0/1:1.0

```

- c. Verify iGW-21 in DC-2 (192.168.21.1) receives the EVPN Type 6 route with the PEG role advertisement from the WAN, and re-originates the route with the DC-2 RD toward the OISM PE devices in DC-2:

```

user@iGW-21> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*0.0.0.0*

evpn-vxlan-A.evpn.0: 571 destinations, 1796 routes (566 active, 0 holddown, 317 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.21.1:9::4::0.0.0.0::192.168.21.1/520
*[EVPN/170] 3d 08:19:47
  Indirect
6:192.168.22.1:9::4::0.0.0.0::192.168.22.1/520
*[BGP/170] 3d 08:19:47, localpref 100, from 192.168.50.50
  AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 08:19:47, localpref 100, from 192.168.22.1
  AS path: 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 08:12:47, localpref 100, from 192.168.23.1
  AS path: 65203 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 08:19:47, localpref 100, from 192.168.24.1
  AS path: 65204 65202 I, validation-state: unverified

```

```

> to 10.21.53.2 via et-0/0/1:1.0
[BGP/170] 3d 08:19:41, localpref 100, from 192.168.25.1
AS path: 65205 65202 I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:111:100::2::0.0.0.0::192.168.1.1/520
*[BGP/170] 3d 08:19:45, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:111:100::4::0.0.0.0::192.168.1.1/520
*[BGP/170] 3d 08:19:45, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:111:100::5::0.0.0.0::192.168.1.1/520
*[BGP/170] 3d 08:19:45, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:112:100::2::0.0.0.0::192.168.2.1/520
*[BGP/170] 3d 08:19:47, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:112:100::4::0.0.0.0::192.168.2.1/520
*[BGP/170] 3d 08:19:47, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0
6:112:100::5::0.0.0.0::192.168.2.1/520
*[BGP/170] 3d 08:19:47, localpref 100, from 192.168.50.50
AS path: I, validation-state: unverified
> to 10.21.53.2 via et-0/0/1:1.0

```

- d. Verify Leaf-21 in DC-1 (192.168.23.1) receives EVPN Type 6 route with the PEG advertisement from iGW-21 (192.168.21.1):

```

user@Leaf-21> show route table evpn-vxlan-A.evpn.0 match-prefix 6:*0.0.0.0*

evpn-vxlan-A.evpn.0: 257 destinations, 935 routes (257 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.21.1:9::4::0.0.0.0::192.168.21.1/520
*[BGP/170] 3d 08:16:04, localpref 100, from 192.168.21.1
AS path: 65201 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
[BGP/170] 3d 08:16:01, localpref 100, from 192.168.22.1

```

```

        AS path: 65202 65000 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
[BGP/170] 3d 08:15:57, localpref 100, from 192.168.24.1
        AS path: 65204 65201 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
[BGP/170] 3d 08:15:45, localpref 100, from 192.168.25.1
        AS path: 65205 65201 I, validation-state: unverified
> to 10.23.21.1 via xe-0/0/0:3.0
6:192.168.22.1:9::4::0.0.0.0::192.168.22.1/520
*[BGP/170] 3d 08:16:01, localpref 100, from 192.168.22.1
        AS path: 65202 I, validation-state: unverified
> to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 08:16:04, localpref 100, from 192.168.21.1
        AS path: 65201 65000 I, validation-state: unverified
> to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 08:15:57, localpref 100, from 192.168.24.1
        AS path: 65204 65202 I, validation-state: unverified
> to 10.23.22.1 via xe-0/0/1:0.0
[BGP/170] 3d 08:15:45, localpref 100, from 192.168.25.1
        AS path: 65205 65202 I, validation-state: unverified
> to 10.23.22.1 via xe-0/0/1:0.0

```

10. `show evpn igmp-snooping proxy` **<extensive>**—Verify IGMP snooping forwarding paths for the expected multicast groups, multicast sources, and VLANs (including the tenant VLANs 2, 3, and 5, and the SBD VLAN 4) are established across the DCI. For example, in our configuration:

iGW-11:

```

user@iGW-11> show evpn igmp-snooping proxy
Instance: evpn-vxlan-A
  VN Identifier: 2
    Group IP: 0.0.0.0, Source IP: 0.0.0.0
    Interconn advertisement route status: DCI route created
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
    Interconn advertisement route status: Route creation pending
    Group IP: 233.252.0.2, Source IP: 0.0.0.0
    Interconn advertisement route status: Route creation pending
  VN Identifier: 4
    Group IP: 0.0.0.0, Source IP: 0.0.0.0
    Interconn advertisement route status: DCI route created
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
    Interconn advertisement route status: DCI route created

```

```

Group IP: 233.252.0.1, Source IP: 172.20.1.15
Interconn advertisement route status: Not created
Group IP: 233.252.0.2, Source IP: 0.0.0.0
Interconn advertisement route status: DCI route created
Group IP: 233.252.0.2, Source IP: 172.20.1.15
Interconn advertisement route status: Not created
VN Identifier: 5
Group IP: 0.0.0.0, Source IP: 0.0.0.0
Interconn advertisement route status: DCI route created
Group IP: 233.252.0.1, Source IP: 0.0.0.0
Interconn advertisement route status: Not created
Group IP: 233.252.0.2, Source IP: 0.0.0.0
Interconn advertisement route status: Not created

```

```
user@iGW-11> show evpn igmp-snooping proxy extensive
```

```
Instance: evpn-vxlan-A
```

```
VN Identifier: 2
```

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
0.0.0.0	0.0.0.0	1	1	0	0	0/0/0/1
233.252.0.1	0.0.0.0	1	1	0	4003	0/0/1/1
233.252.0.2	0.0.0.0	1	1	0	4003	0/0/1/1

```
VN Identifier: 4
```

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
0.0.0.0	0.0.0.0	1	1	0	0	0/0/0/1
233.252.0.1	0.0.0.0	1	4	2	4047	0/0/1/1
233.252.0.1	172.20.1.15	0	0	0	4069	0/1/0/0
233.252.0.2	0.0.0.0	1	4	2	4047	0/0/1/1
233.252.0.2	172.20.1.15	0	0	0	4069	0/1/0/0

```
VN Identifier: 5
```

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
0.0.0.0	0.0.0.0	1	1	0	0	0/0/0/1
233.252.0.1	0.0.0.0	0	1	0	4003	0/0/0/0
233.252.0.2	0.0.0.0	0	1	0	4003	0/0/0/0

iGW-12:

```
user@iGW-12> show evpn igmp-snooping proxy
```

```
Instance: evpn-vxlan-A
```

```
VN Identifier: 2
```

Group IP: 0.0.0.0, Source IP: 0.0.0.0
 Interconn advertisement route status: DCI route created
 Group IP: 233.252.0.1, Source IP: 0.0.0.0
 Interconn advertisement route status: Route creation pending
 Group IP: 233.252.0.2, Source IP: 0.0.0.0
 Interconn advertisement route status: Route creation pending

VN Identifier: 4

Group IP: 0.0.0.0, Source IP: 0.0.0.0
 Interconn advertisement route status: DCI route created
 Group IP: 233.252.0.1, Source IP: 0.0.0.0
 Interconn advertisement route status: DCI route created
 Group IP: 233.252.0.1, Source IP: 172.20.1.15
 Interconn advertisement route status: Not created
 Group IP: 233.252.0.2, Source IP: 0.0.0.0
 Interconn advertisement route status: DCI route created
 Group IP: 233.252.0.2, Source IP: 172.20.1.15
 Interconn advertisement route status: Not created

VN Identifier: 5

Group IP: 0.0.0.0, Source IP: 0.0.0.0
 Interconn advertisement route status: DCI route created
 Group IP: 233.252.0.1, Source IP: 0.0.0.0
 Interconn advertisement route status: Not created
 Group IP: 233.252.0.2, Source IP: 0.0.0.0
 Interconn advertisement route status: Not created

user@iGW-12> **show evpn igmp-snooping proxy extensive**

Instance: evpn-vxlan-A

VN Identifier: 2

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/
Lcl-Refresh						
0.0.0.0	0.0.0.0	1	1	0	0	0/0/0/1
233.252.0.1	0.0.0.0	1	1	0	4023	0/0/1/1
233.252.0.2	0.0.0.0	1	1	0	4023	0/0/1/1

VN Identifier: 4

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/
Lcl-Refresh						
0.0.0.0	0.0.0.0	1	1	0	0	0/0/0/1
233.252.0.1	0.0.0.0	1	4	2	4067	0/0/1/1
233.252.0.1	172.20.1.15	0	0	0	4081	0/1/0/0
233.252.0.2	0.0.0.0	1	4	2	4067	0/0/1/1
233.252.0.2	172.20.1.15	0	0	0	4081	0/1/0/0

VN Identifier: 5

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/
-------	--------	-------	--------	-----------	--------	----------------------

Lcl-Refresh						
0.0.0.0	0.0.0.0	1	1	0	0	0/0/0/1
233.252.0.1	0.0.0.0	0	1	0	4023	0/0/0/0
233.252.0.2	0.0.0.0	0	1	0	4023	0/0/0/0

Leaf-11:

```
user@Leaf-11> show evpn igmp-snooping proxy
```

Instance: evpn-vxlan-A

VN Identifier: 2

Group IP: 233.252.0.1, Source IP: 0.0.0.0

Group IP: 233.252.0.2, Source IP: 0.0.0.0

VN Identifier: 4

Group IP: 0.0.0.0, Source IP: 0.0.0.0

Group IP: 233.252.0.1, Source IP: 0.0.0.0

Group IP: 233.252.0.2, Source IP: 0.0.0.0

VN Identifier: 5

Group IP: 233.252.0.1, Source IP: 0.0.0.0

Group IP: 233.252.0.2, Source IP: 0.0.0.0

```
user@Leaf-11> show evpn igmp-snooping proxy extensive
```

Instance: evpn-vxlan-A

VN Identifier: 2

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
233.252.0.1	0.0.0.0	1	1	0	524311	0/0/1/1
233.252.0.2	0.0.0.0	1	1	0	524311	0/0/1/1

VN Identifier: 4

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
0.0.0.0	0.0.0.0	0	2	0	0	0/0/0/0
233.252.0.1	0.0.0.0	1	4	0	524312	0/0/1/1
233.252.0.2	0.0.0.0	1	4	0	524312	0/0/1/1

VN Identifier: 5

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
233.252.0.1	0.0.0.0	0	1	0	524311	0/0/0/0
233.252.0.2	0.0.0.0	0	1	0	524311	0/0/0/0

iGW-21:

```
user@iGW-21> show evpn igmp-snooping proxy
```

```
Instance: evpn-vxlan-A
```

```
  VN Identifier: 3
```

```
    Group IP: 0.0.0.0, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: Not created
```

```
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: Route creation pending
```

```
    Group IP: 233.252.0.2, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: Route creation pending
```

```
  VN Identifier: 4
```

```
    Group IP: 0.0.0.0, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: DC route created
```

```
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: DCI route created
```

```
    Group IP: 233.252.0.2, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: DCI route created
```

```
  VN Identifier: 5
```

```
    Group IP: 0.0.0.0, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: Not created
```

```
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: Not created
```

```
    Group IP: 233.252.0.2, Source IP: 0.0.0.0
```

```
    Interconn advertisement route status: Not created
```

```
user@iGW-21> show evpn igmp-snooping proxy extensive
```

```
Instance: evpn-vxlan-A
```

```
  VN Identifier: 3
```

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
0.0.0.0	0.0.0.0	0	1	0	0	0/0/0/0
233.252.0.1	0.0.0.0	1	1	0	4049	0/0/1/1
233.252.0.2	0.0.0.0	1	1	0	4049	0/0/1/1

```
  VN Identifier: 4
```

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
0.0.0.0	0.0.0.0	0	1	2	0	0/0/0/0
233.252.0.1	0.0.0.0	1	4	2	4079	0/0/1/1
233.252.0.2	0.0.0.0	1	4	2	4079	0/0/1/1

```
  VN Identifier: 5
```

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/Lcl-Refresh
-------	--------	-------	--------	-----------	--------	---------------------------------

```
Refresh
0.0.0.0      0.0.0.0    0      1      0      0      0/0/0/0
233.252.0.1  0.0.0.0    0      1      0      4049   0/0/0/0
233.252.0.2  0.0.0.0    0      1      0      4049   0/0/0/0
```

Leaf-21:

```
user@Leaf-21> show evpn igmp-snooping proxy
```

Instance: evpn-vxlan-A

VN Identifier: 2

Group IP: 233.252.0.1, Source IP: 0.0.0.0

Group IP: 233.252.0.2, Source IP: 0.0.0.0

VN Identifier: 4

Group IP: 0.0.0.0, Source IP: 0.0.0.0

Group IP: 233.252.0.1, Source IP: 0.0.0.0

Group IP: 233.252.0.2, Source IP: 0.0.0.0

VN Identifier: 5

Group IP: 233.252.0.1, Source IP: 0.0.0.0

Group IP: 233.252.0.2, Source IP: 0.0.0.0

```
user@Leaf-21> show evpn igmp-snooping proxy extensive
```

Instance: evpn-vxlan-A

VN Identifier: 2

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/
Lcl-Refresh						
233.252.0.1	0.0.0.0	1	1	0	524291	0/0/1/1
233.252.0.2	0.0.0.0	1	1	0	524291	0/0/1/1

VN Identifier: 4

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/
Lcl-Refresh						
0.0.0.0	0.0.0.0	0	2	0	0	0/0/0/0
233.252.0.1	0.0.0.0	1	4	0	524331	0/0/1/1
233.252.0.2	0.0.0.0	1	4	0	524331	0/0/1/1

VN Identifier: 5

Group	Source	Local	Remote	Remote DC	Corenh	Flood/Lclsrc/L3-MGM/
Lcl-Refresh						
233.252.0.1	0.0.0.0	0	1	0	524291	0/0/0/0
233.252.0.2	0.0.0.0	0	1	0	524291	0/0/0/0

11. `show evpn igmp-snooping proxy internal`—Get internal details on multicast queues and operating modes for troubleshooting. For example, in our configuration:

iGW-11 (output for peer iGW-12 and iGW-21 is similar):

```

user@iGW-11> show evpn igmp-snooping proxy internal
Instance: evpn-vxlan-A
  VN Identifier: 2
    Group IP: 0.0.0.0, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 0, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0
      MCSN Added: 1, MCSN Deleted: 0
      In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
      Peer IP: 192.168.2.1, Flags: 0x2
      Interconn advertisement route status: DCI route created
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 1, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0
      MCSN Added: 1, MCSN Deleted: 0
      In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
      Peer IP: 192.168.2.1, Flags: 0x2
      Interconn advertisement route status: Route creation pending
    Group IP: 233.252.0.2, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 1, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0
      MCSN Added: 1, MCSN Deleted: 0
      In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
      Peer IP: 192.168.2.1, Flags: 0x2
      Interconn advertisement route status: Route creation pending
  VN Identifier: 4
    Group IP: 0.0.0.0, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 0, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0
      MCSN Added: 1, MCSN Deleted: 0
      In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
      Peer IP: 192.168.2.1, Flags: 0x2
      Interconn advertisement route status: DCI route created
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 1, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0

```

MCSN Added: 1, MCSN Deleted: 0

In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N

Peer IP: 192.168.3.1, Flags: 0x0

Peer IP: 192.168.4.1, Flags: 0x0

Peer IP: 192.168.2.1, Flags: 0x2

Peer IP: 192.168.22.1, Flags: 0x1

Peer IP: 192.168.21.1, Flags: 0x1

Peer IP: 192.168.5.1, Flags: 0x0

Interconn advertisement route status: DCI route created

Group IP: 233.252.0.1, Source IP: 172.20.1.15

SMET Flood: 0, Enhanced OISM Local Source: 1

L3-MGM Notified: 0, Local Refreshed: 0

Enhanced OISM Forward on Source BD: 0

MCSN Added: 0, MCSN Deleted: 0

In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N

Interconn advertisement route status: Not created

VN Identifier: 5

Group IP: 0.0.0.0, Source IP: 0.0.0.0

SMET Flood: 0, Enhanced OISM Local Source: 0

L3-MGM Notified: 0, Local Refreshed: 1

Enhanced OISM Forward on Source BD: 0

MCSN Added: 1, MCSN Deleted: 0

In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N

Peer IP: 192.168.2.1, Flags: 0x2

Interconn advertisement route status: DCI route created

Group IP: 233.252.0.1, Source IP: 0.0.0.0

SMET Flood: 0, Enhanced OISM Local Source: 0

L3-MGM Notified: 0, Local Refreshed: 0

Enhanced OISM Forward on Source BD: 0

MCSN Added: 1, MCSN Deleted: 0

In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N

Peer IP: 192.168.2.1, Flags: 0x2

Interconn advertisement route status: Not created

Group IP: 233.252.0.2, Source IP: 0.0.0.0

SMET Flood: 0, Enhanced OISM Local Source: 0

L3-MGM Notified: 0, Local Refreshed: 0

Enhanced OISM Forward on Source BD: 0

MCSN Added: 1, MCSN Deleted: 0

In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N

Peer IP: 192.168.2.1, Flags: 0x2

Interconn advertisement route status: Not created

Leaf-11 (output for Leaf-21 is similar):

```

user@Leaf-11> show evpn igmp-snooping proxy internal
Instance: evpn-vxlan-A
  VN Identifier: 2
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 1, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0
      MCSN Added: 1, MCSN Deleted: 0
      In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
      Peer IP: 192.168.4.1, Flags: 0x0
    Group IP: 233.252.0.2, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 1, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0
      MCSN Added: 1, MCSN Deleted: 0
      In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
      Peer IP: 192.168.4.1, Flags: 0x0
  VN Identifier: 4
    Group IP: 0.0.0.0, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 0, Local Refreshed: 0
      Enhanced OISM Forward on Source BD: 0
      MCSN Added: 1, MCSN Deleted: 0
      In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
      Peer IP: 192.168.1.1, Flags: 0x0
      Peer IP: 192.168.2.1, Flags: 0x0
    Group IP: 233.252.0.1, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 1, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0
      MCSN Added: 1, MCSN Deleted: 0
      In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
      Peer IP: 192.168.4.1, Flags: 0x0
      Peer IP: 192.168.2.1, Flags: 0x0
      Peer IP: 192.168.1.1, Flags: 0x0
      Peer IP: 192.168.5.1, Flags: 0x0
    Group IP: 233.252.0.2, Source IP: 0.0.0.0
      SMET Flood: 0, Enhanced OISM Local Source: 0
      L3-MGM Notified: 1, Local Refreshed: 1
      Enhanced OISM Forward on Source BD: 0

```

```

MCSN Added: 1, MCSN Deleted: 0
In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
Peer IP: 192.168.4.1, Flags: 0x0
Peer IP: 192.168.2.1, Flags: 0x0
Peer IP: 192.168.1.1, Flags: 0x0
Peer IP: 192.168.5.1, Flags: 0x0
VN Identifier: 5
Group IP: 233.252.0.1, Source IP: 0.0.0.0
SMET Flood: 0, Enhanced OISM Local Source: 0
L3-MGM Notified: 0, Local Refreshed: 0
Enhanced OISM Forward on Source BD: 0
MCSN Added: 1, MCSN Deleted: 0
In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
Peer IP: 192.168.4.1, Flags: 0x0
Group IP: 233.252.0.2, Source IP: 0.0.0.0
SMET Flood: 0, Enhanced OISM Local Source: 0
L3-MGM Notified: 0, Local Refreshed: 0
Enhanced OISM Forward on Source BD: 0
MCSN Added: 1, MCSN Deleted: 0
In SMET Queue: N, In Add Queue: N, In Delete Queue: N, In PIM Queue: N
Peer IP: 192.168.4.1, Flags: 0x0

```

- 12.** `show igmp snooping evpn status <detail>`—Verify the configured EVPN OISM mode and role of each VLAN associated with the configured EVPN instance(s). For example, in our configuration, you can see that enhanced OISM over DCI is enabled on the iGW devices, and all devices in the EVPN network have VLAN_4 (VNI 4) configured as the OISM SBD. Also, iGW-11, iGW-12, Leaf-11, and Leaf-21 serve tenant VLANs VLAN_2 and VLAN_5, while iGW-21 and iGW-22 serve tenant VLANs VLAN_3 and VLAN_5.

iGW-11 (output for peer iGW-12 is the same):

```

user@iGW-11> show igmp snooping evpn status detail
Mode: Enhanced OISM
SMET enabled: Yes

Instance: evpn-vxlan-A
  Bridge-Domain: VLAN_2, VN Identifier: 2
    OISM          : Enabled
    Supplementary BD: No
    External VLAN  : No
    DCI status     : Enabled
  Bridge-Domain: VLAN_4, VN Identifier: 4

```

```

OISM          : Enabled
Supplementary BD: Yes
External VLAN  : No
DCI status     : Enabled
Bridge-Domain: VLAN_5, VN Identifier: 5
OISM          : Enabled
Supplementary BD: No
External VLAN  : No
DCI status     : Enabled

```

Leaf-11:

```

user@Leaf-11> show igmp snooping evpn status detail
Mode: Enhanced OISM
SMET enabled: Yes

Instance: evpn-vxlan-A
Bridge-Domain: VLAN_2, VN Identifier: 2
OISM          : Enabled
Supplementary BD: No
External VLAN  : No
DCI status     : Disabled
Bridge-Domain: VLAN_4, VN Identifier: 4
OISM          : Enabled
Supplementary BD: Yes
External VLAN  : No
DCI status     : Disabled
Bridge-Domain: VLAN_5, VN Identifier: 5
OISM          : Enabled
Supplementary BD: No
External VLAN  : No
DCI status     : Disabled

```

iGW-21:

```

user@iGW-21> show igmp snooping evpn status detail
Mode: Enhanced OISM
SMET enabled: Yes

Instance: evpn-vxlan-A
Bridge-Domain: VLAN_4, VN Identifier: 4

```

```

        OISM          : Enabled
        Supplementary BD: Yes
        External VLAN   : No
        DCI status      : Enabled
    Bridge-Domain: VLAN_5, VN Identifier: 5
        OISM          : Enabled
        Supplementary BD: No
        External VLAN   : No
        DCI status      : Enabled
    Bridge-Domain: VLAN_3, VN Identifier: 3
        OISM          : Enabled
        Supplementary BD: No
        External VLAN   : No
        DCI status      : Enabled

```

Leaf-21:

```

user@Leaf-21> show igmp snooping evpn status detail
Mode: Enhanced OISM
SMET enabled: Yes

Instance: evpn-vxlan-A
    Bridge-Domain: VLAN_4, VN Identifier: 4
        OISM          : Enabled
        Supplementary BD: Yes
        External VLAN   : No
        DCI status      : Disabled
    Bridge-Domain: VLAN_5, VN Identifier: 5
        OISM          : Enabled
        Supplementary BD: No
        External VLAN   : No
        DCI status      : Disabled
    Bridge-Domain: VLAN_2, VN Identifier: 2
        OISM          : Enabled
        Supplementary BD: No
        External VLAN   : No
        DCI status      : Disabled

```


Microsegmentation Using Group-Based Policies

IN THIS CHAPTER

- Overview | 1272
- Typical Network Use Cases | 1274
- Supported Platforms | 1278
- GBP Profiles | 1279
- GBP Processing | 1282
- Unified Access Policy | 1309
- Example: Using GBP to Segment Traffic | 1319

Overview

SUMMARY

Use group based policies (GBP) to support microsegmentation.

Microsegmentation, in the context of networking, refers to the support of granular network access policy and fine-grained network access controls. It subdivides a network into smaller isolated segments that you control through individual security policies. Microsegmentation augments the perimeter security provided by firewalls and presents a particularly useful technique in containing lateral threats in your network.

Juniper Networks Group Based Policy (GBP) implementation supports microsegmentation for wired and wireless clients by allowing you to conceptually separate clients into groups. You then create different policies to apply to the different groups of clients. Policy enforcement is not tied to traditional networking constructs used in ACLs such as MAC or IP addresses. You can apply the same security policy to users from within the same VLAN or across VLANs, or from within the same switch or across

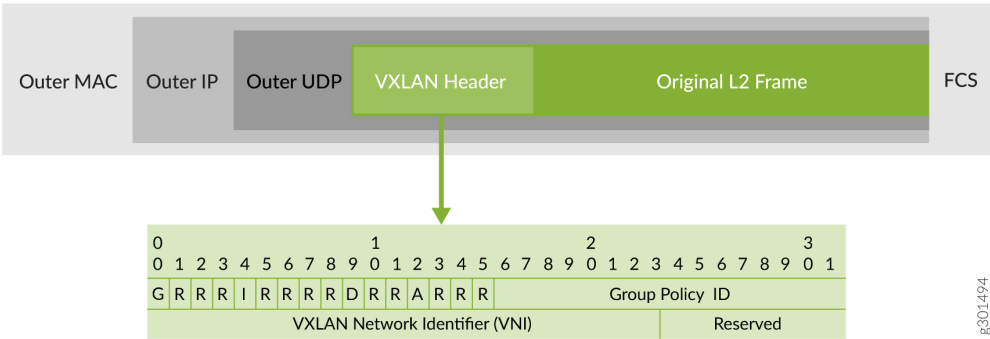
switches, all without reference to network location. Policy enforcement with GBP is truly location agnostic, untethered to where clients reside in the physical network.

GBP enforces policy based on Scalable Group Tags (SGTs), which you assign to individual clients statically through CLI configuration or dynamically through RADIUS authentication. Creating policies based on tags decouples the client from their network location. This means that unlike ACLs and firewall rules, SGTs are location-agnostic and provide natural support for mobility. A GBP policy doesn't change simply because a client has changed to a different network attachment point. This simplifies network security management, allowing seamless connectivity and consistent application of policy no matter where the client is. Moreover, GBP-based microsegmentation is highly scalable because GBP policy is applied to groups instead of to individual clients and their myriad connections.

NOTE: SGTs are also known as GBP tags. We use these terms interchangeably in this document.

Although you can enable GBP on different network infrastructure, you'll achieve the most benefit when running on an EVPN-VXLAN network. That's because the GBP tag (shown as the Group Policy ID in [Figure 120 on page 1273](#)) is carried inband alongside data in the VXLAN frame. This allows the switch at the remote end to see the GBP tag associated with the user data and apply policy based on this tag. (See [I-D.draft-smith-vxlan-group-policy](#) for more information on the VXLAN header structure when using GBP.)

Figure 120: VXLAN Header Fields with GBP



When a GBP-enabled EVPN-VXLAN switch receives a packet from a client, the switch classifies the packet to obtain the GBP tag. Then the switch encapsulates the entire packet in VXLAN and inserts the GBP tag into the VXLAN header prior to forwarding the packet through the appropriate VXLAN tunnel.

When the packet exits the other end of the tunnel, the GBP-enabled switch at the far end extracts the tag and makes a policy decision based on the extracted tag and on the tag associated with the

destination. The policy itself is based purely on GBP tags, which are associated with clients and not network location.

In the event that your network is sufficiently small that it doesn't warrant running EVPN-VXLAN, you can still take advantage of GBP capabilities. In this situation, there is no VXLAN tunnel to carry the GBP tag. Therefore you have to enforce policy locally at the ingress. As before, the GBP-enabled switch at the access receives the packet from the client and classifies it to retrieve the assigned tag. The same switch then enforces policy locally based on this tag along with other packet header information.

Whether you run with EVPN-VXLAN or not, there are three distinct aspects to GBP support:

- Tag assignment configures the mapping between the client and the group.
- Packet classification classifies the incoming packet in the dataplane to obtain the GBP tags (source tag at ingress, destination tag at egress).
- Policy enforcement enforces policy based on the GBP tags.

We cover all three aspects in ["GBP Processing" on page 1282](#).

Typical Network Use Cases

SUMMARY

GBP is supported on different network topologies.

IN THIS SECTION

- [GBP in Large Campus Networks | 1274](#)
- [GBP in Branch and Small Campus Networks | 1276](#)

We support GBP on different topologies on IPv4 and IPv6 overlay networks, and on IPv4 (and IPv6 starting in Junos OS Release 25.4R1) underlay networks. Whether you're deploying a large campus network or a branch or small campus network, you can use GBP to provide a common microsegmentation solution.

GBP in Large Campus Networks

A large campus network is typically constructed as an IP Clos fabric consisting of an access layer, a distribution layer, and in the largest deployments, a distinct core layer. This arrangement allows high scale, non-blocking connectivity between end user devices. When coupled with an overlay technology such as EVPN-VXLAN, you can extend layer 2 connectivity across campus with no constraint on physical location while limiting excessive broadcasts and flooding.

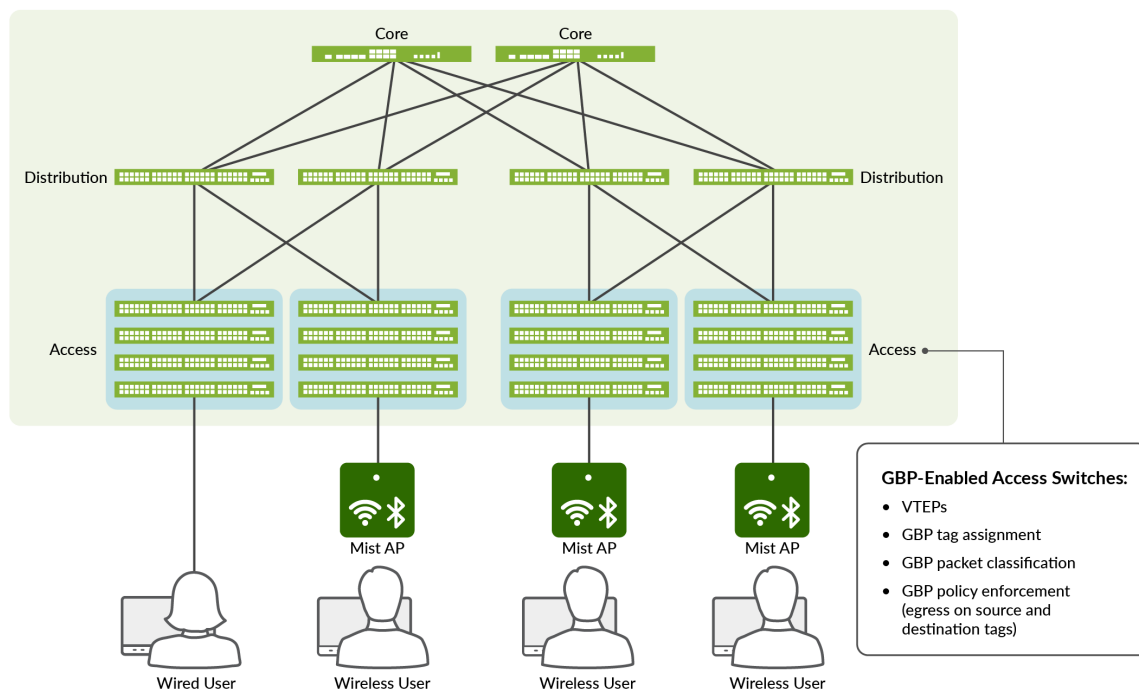
EVPN-VXLAN delivers scalability and flexibility to your network by decoupling the virtual (overlay) topology from the physical (underlay) topology. This reduces operational overhead and time to deployment and increases portability. The IP Clos fabric provides the physical any-to-any connectivity while the EVPN-VXLAN network controls the virtual connectivity between end user equipment. More specifically, in the context of microsegmentation, EVPN-VXLAN allows you to extend Group Based Policies (GBP) to provide microsegmentation across the entire campus network.

The access switches act as VXLAN tunnel endpoints (VTEPs). On ingress, packets from the access link are encapsulated in VXLAN and forwarded through the appropriate tunnels. On egress, packets exiting from the VTEP are decapsulated and sent out the access link.

As the home to VTEPs, the access switches are perfectly positioned to support microsegmentation using GBP. The access switch classifies an incoming packet to obtain the GBP tag and inserts this tag into the VXLAN header prior to forwarding the VXLAN-encapsulated frame through the appropriate tunnel. The access switch at the remote end of the tunnel decapsulates the VXLAN-encapsulated frame and extracts the tag. The remote access switch can then use the local and remote tags to enforce policy.

This is shown in [Figure 121 on page 1275](#).

Figure 121: Large Campus IP Clos Network



This type of architecture is ideal for large campus sites. By delegating all policy enforcement to the access switches, you can support layer 2 / layer 3 segmentation at scale. You enjoy the benefit of a fully-featured GBP microsegmentation solution on a highly scalable EVPN-VXLAN infrastructure.

While you can configure tag assignment using either the CLI or RADIUS authentication, using RADIUS authentication is the preferred approach for large networks. With RADIUS, you don't require prior knowledge of client networking configuration as you do for CLI. You simply configure your RADIUS server to offer the desired GBP tag to each client. The tag is conveyed in a special vendor-specific attribute (VSA) as part of authentication, compatible with any RADIUS server that supports VSAs. See ["Assigning Tags Using RADIUS" on page 1288](#) for more information on the VSAs used for GBP configuration.

There are two situations to consider when assigning tags using RADIUS:

- The client device is directly connected to the access switch. In this situation, the access switch acts as the authenticator and processes RADIUS messages to and from the authentication server. The access switch can therefore extract the assigned GBP tag from the authentication server's response and automatically configure it locally on the switch.
- The client device connects to a wireless access point (AP). The wireless AP then connects to the access switch. In this situation, we can authenticate the AP as if it were a client device directly connected to the access switch. However, this only provides authentication (and GBP tag assignment) at the AP level, which means that all wireless clients of the AP are authenticated and grouped together.

In order to provide granularity down to the client level, we must perform authentication and assign GBP tags at the client level. In order to do this, the wireless AP must act as the authenticator, not the access switch. As the authenticator, the wireless AP has knowledge of the GBP tag assigned to each of its clients, but it needs a way to convey this tag to the upstream access switch.

Starting in Junos OS Release 25.4R1, if the wireless access point is a Mist AP, the Mist AP can use proprietary messaging to convey the assigned GBP tag to the upstream access switch. Upon receiving this notification, the access switch configures the assigned tag as if it were configured or learned locally. See ["Unified Access Policy" on page 1309](#) for more information on this scenario.

GBP in Branch and Small Campus Networks

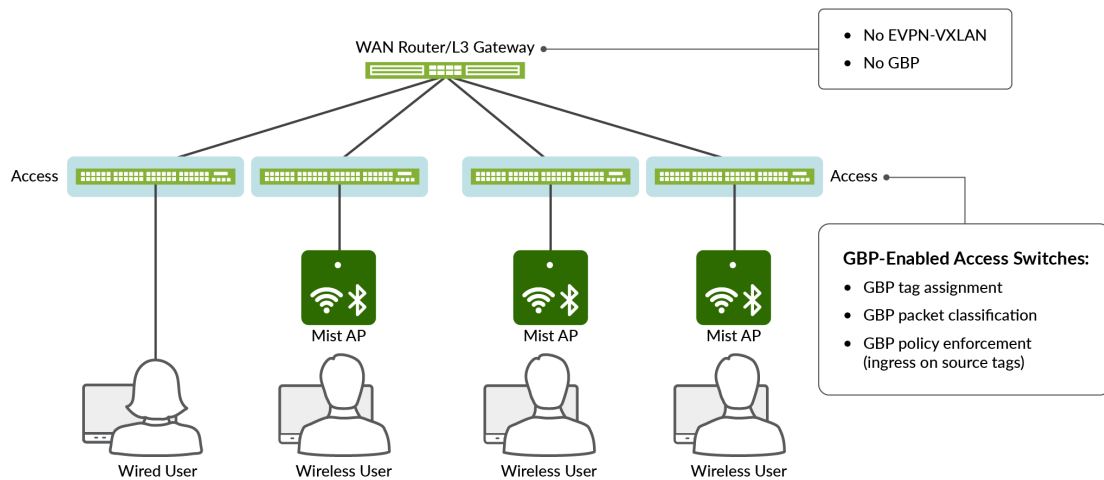
A branch or small campus environment may not have the infrastructure to run a full EVPN-VXLAN implementation. The network may simply consist of VLANs running on a collection of layer 2 switches. Fortunately, this does not preclude you from using Juniper's GBP solution to segment your network. Starting in Junos OS Release 25.4R1, Juniper's GBP solution can work (with some restrictions) in this type of pure layer 2 environment.

[Figure 122 on page 1277](#) shows layer 2 access switches connected to external networks through a WAN router. You configure the access switches to assign GBP tags based on layer 2 fields. Since there is no VXLAN encapsulation or EVPN signaling, the assigned GBP tags are not propagated through the

network. Policy enforcement, therefore, takes place purely using locally available layer 2 and layer 3 information at the ingress.

Even with this restriction, the ability to use GBP to segment traffic in small networks like this provides a lot of the same benefits that the full GBP solution offers: operational simplicity, consistent policy application, and high scalability.

Figure 122: GBP in Layer 2 Branch Network



Similar to GBP in large campus networks, you configure tag assignment using the CLI or RADIUS authentication.



NOTE: There are restrictions on what GBP microsegmentation filters you can configure using the CLI for this use case. See ["Using the GBP Pure L2 Profile"](#) on page 1316 for more details.

When using RADIUS authentication, the behavior is the same as for large campus networks:

- The access switch acts as the authenticator for wired users and configures the tag assignment automatically when the wired user is authenticated.
- The Mist AP acts as the authenticator for wireless users and notifies the access switch of the tag assignment using proprietary messaging. Upon receiving this notification, the access switch configures the tag assignment into its MAC address tables and internal data structures.

For more information on deploying GBP in branch and small campus networks, see ["Unified Access Policy"](#) on page 1309.

Supported Platforms

SUMMARY

Here are the platforms that support group based policies (GBP).

Table 65 on page 1278 lists the minimum release for platform support.



NOTE: The minimum release refers to the release in which GBP support was introduced. The minimum release does not necessarily equate to the current recommended release for network deployment. See <https://supportportal.juniper.net/s/article/Junos-Software-Versions-Suggested-Releases-to-Consider-and-Evaluate> for the recommended releases for each platform.

Table 65: VXLAN-GBP Supported Switches

Junos Release	VXLAN-GBP Supported Switches
Starting with Junos OS release 21.1R1	EX4400-24P, EX4400-24T, EX4400-48F, EX4400-48P, and EX4400-48T
Starting with Junos OS release 21.2R1	EX4400-24MP and EX4400-48MP
Starting with Junos OS release 21.4R1	<ul style="list-style-type: none">• QFX5120-32C and QFX5120-48Y• EX4650
Starting with Junos OS release 22.4R1	<ul style="list-style-type: none">• EX4100 Series
Starting with Junos OS release 23.2R1	<ul style="list-style-type: none">• EX4400-24X• EX9204/9208/9214 (with EX9200-15C)

Table 65: VXLAN-GBP Supported Switches *(Continued)*

Junos Release	VXLAN-GBP Supported Switches
Starting with Junos OS release 24.2R1	<ul style="list-style-type: none"> • MX240/480/960 (with MPC-10E) • MX304 • MX10004 • MX10008
Starting with Junos OS release 24.4R1	<ul style="list-style-type: none"> • EX4100-H Series • EX4400-48XP and EX4400-48MXP • QFX5120-48T and QFX5120-48YM

GBP Profiles

SUMMARY

GBP profiles allow you to allocate forwarding table resources appropriately for your deployment.

Before running GBP, you must select a GBP profile to use.

GBP profiles are unified forwarding table (UFT) profiles that allow you to allocate forwarding table resources tailored to your network when you run GBP. This enables you to allocate a higher percentage of memory for one type of address over another depending on your needs. The GBP profile determines the table sizes to allocate for the various GBP filters. Set one of the available profiles at the [edit chassis forwarding-options] hierarchy level that best meets your network needs.

[Table 66 on page 1280](#) shows the supported GBP profiles.

Table 66: Supported VXLAN-GBP Profiles (Junos OS Release 23.2R1 and Later)

Profiles	Description	Supported Devices ¹	Release Introduced
vxlan-gbp-profile	Suitable for a balanced configuration that contains a mix of L2 and L3 networks.	<ul style="list-style-type: none"> • EX4100 • EX4400 • EX4650 • QFX5120 	Junos OS Release 21.1R1
vxlan-gbp-l2-profile	Suitable when running an edge device in a virtualized network comprised of multiple virtual servers and VMs. You can also use this profile for direct reachability if the edge device needs more space in the mac-table.	<ul style="list-style-type: none"> • EX4400 • EX4650 • QFX5120 	Junos OS Release 23.2R1
vxlan-gbp-l3-profile	Suitable for an edge device running in an L3-intensive network where more space in the host-table is required.	<ul style="list-style-type: none"> • EX4400 • EX4650 • QFX5120 	Junos OS Release 23.2R1

Table 66: Supported VXLAN-GBP Profiles (Junos OS Release 23.2R1 and Later) *(Continued)*

Profiles	Description	Supported Devices ¹	Release Introduced
vxlan-gbp-mc-profile	<p>Required for the network to seamlessly support GBP unicast traffic flows together with multicast flows that use enhanced OISM across the VXLAN tunnels.</p> <p>NOTE: In a GBP-enabled network, we support OISM only in enhanced mode (not regular mode). Also, only unicast traffic carries GBP tags in the VXLAN headers; we don't assign GBP tags to multicast traffic.</p>	<ul style="list-style-type: none"> • EX4100 • EX4400 • EX4650 • QFX5120 	Junos OS Release 24.2R2-S2
gbp-pure-l2-profile	<p>Configure when running GBP in a pure layer 2 deployment. A pure layer 2 network does not support EVPN-VXLAN. See "Using the GBP Pure L2 Profile" on page 1316.</p>	<ul style="list-style-type: none"> • EX4100 • EX4400 • EX4650 • QFX5120 	Junos OS Release 25.4R1
¹ Specific variants are listed in "Supported Platforms" on page 1278.			



NOTE: If you set or delete a GBP profile, the Packet Forwarding Engine (PFE) restarts automatically.



NOTE: If you set or delete a GBP profile in a virtual chassis, you must reboot all members of the virtual chassis: `request system reboot all-members`

GBP Processing

SUMMARY

GBP processing includes tag assignment, packet classification, and policy enforcement.

IN THIS SECTION

- [Tag Assignment | 1282](#)
- [Packet Classification | 1294](#)
- [Policy Enforcement | 1294](#)

Tag Assignment

SUMMARY

Assign GBP tags statically using the CLI or dynamically using RADIUS authentication.

IN THIS SECTION

- [Assigning Tags Using the CLI | 1284](#)
- [Assigning Tags Using RADIUS | 1288](#)
- [Tag Assignment Priority | 1292](#)

Tag assignment refers to configuring the mapping between a user and their group and programming this mapping into MAC address tables and internal data structures. Note that you don't configure group membership per se. Different users with the same tag assignment implicitly belong to the same group.

You can configure tag assignment statically through the CLI or dynamically through RADIUS authentication. Tag assignment through RADIUS authentication is the preferred approach as this gives you the greatest flexibility.

- When you configure tag assignment statically using the CLI, you're creating a mapping between a user and a GBP tag, or more accurately, between a proxy of that user (such as their MAC or IP address) and a GBP tag. To make this approach less cumbersome, especially if you have many users, we give you the flexibility to configure tag assignment at different levels, from the interface down to the individual MAC addresses. For example, if you want all users connected to the same interface to be assigned the same tag, you would configure the tag for that interface to apply to all directly connected users. On the other hand, if you want a specific user to be assigned a specific tag, then you would configure tag assignment based on their MAC or IP address.

The downside of static CLI configuration is that it requires apriori knowledge of the user (where the user is connected, what their MAC or IP address is, etc.). You also need to reconfigure this mapping if the user changes location or device in the future.

For example, let's say you want to configure GBP tag 10 for a user named John and you want to use his MAC address as the proxy for that user. In this case, you'll need to know John's MAC address before you can map it to GBP tag 10. Once you configure this mapping, the GBP-enabled switch can then program it into its MAC address tables and internal data structures for data plane processing. If John changes device to another MAC address, you'll need to manually reconfigure this mapping.

We leverage familiar firewall CLI statements to configure tag assignment. Specifically, you create a firewall filter that configures a specific GBP tag when specific criteria (for example, MAC address match) are met. To distinguish between a regular firewall filter and a tag assignment firewall filter, we introduce a new type of filter specific to microsegmentation.

- When you configure tag assignment dynamically using RADIUS authentication, you don't need prior knowledge of the user's networking configuration. You simply configure RADIUS to offer the desired GBP tag when you authenticate the user.

In our example above, when John authenticates with the RADIUS server, he is offered GBP tag 10. There is no need for you to determine John's MAC address at all. The GBP-enabled switch, acting as the authenticator, sees both John's MAC address and the offered GBP tag as part of the authentication exchange. The switch can therefore program this mapping into its MAC address tables and internal data structures directly without requiring CLI intervention. If John changes devices, he will re-authenticate with the RADIUS server, and the GBP-enabled switch will automatically reprogram its MAC address tables and internal data structures with the new association.

Tag assignment using RADIUS is also supported for wireless users when connecting to a Mist AP. In this situation, the Mist AP acts as the authenticator and becomes aware of the mapping between the user's MAC address and their GBP tag. Armed with this information, the Mist AP notifies the upstream GBP-enabled switch of this mapping using proprietary messaging. Upon reception of this notification, the GBP-enabled switch programs its MAC address tables and internal data structures with the new mapping.



NOTE: When you use RADIUS to dynamically configure tag assignment on a directly connected wired user, the GBP-enabled switch creates a MAC or an interface-based assignment depending on how you configure RADIUS.

When you use RADIUS to dynamically configure tag assignment on a wireless user attached to a Mist AP, the GBP-enabled switch creates a MAC-based assignment.

When you use the CLI to statically configure tag assignment, you can configure different types of assignments to suit your requirements. This may reduce the amount of configuration needed in some situations.

Assigning Tags Using the CLI

IN THIS SECTION

- [Example of Matching on MAC Address | 1286](#)
- [Example of Matching on IP Address | 1287](#)
- [Example of Matching on VLANs and Interfaces | 1287](#)

To assign GBP tags using the CLI, you use a special type of firewall filter called a microsegmentation filter. Just like regular firewall filters, you specify the match conditions and the subsequent actions:

```
set firewall family any filter <filter-name> micro-segmentation
set firewall family any filter <filter-name> term <term-name> from <match-condition>
set firewall family any filter <filter-name> term <term-name> then gbp-tag <tag>
```

For the microsegmentation filter, the supported match conditions are shown in [Table 67 on page 1284](#) and the single supported action is to assign a GBP tag.

Table 67: Match Conditions (Junos OS Release 22.4R1 and Later)

Match Conditions	Description
ip-version ipv4 address <ip address> prefix-list <prefix-list> ip-version ipv6 address <ip address> prefix-list <prefix-list>	Match IPv4/IPv6 address/prefix-lists. This match is applied to the source IP address on packets arriving on the access link from the attached client. NOTE: Starting in Junos OS Release 24.4R1, you can specify whether you want IP address terms to be evaluated in term order or by longest prefix match. By default, IP address terms are evaluated by longest prefix match in Junos OS Release 24.4R1 and later. In releases prior to Junos OS Release 24.4R1, IP address terms are evaluated in term order only. See " Longest Prefix Match versus Strict Firewall Term Order " on page 1293.
mac-address <mac address>	Match MAC address. This match is applied to the source MAC address on packets arriving on the access link from the attached client.

Table 67: Match Conditions (Junos OS Release 22.4R1 and Later) *(Continued)*

Match Conditions	Description
<p>interface <interface_name></p>	<p>Match interface name.</p> <p>NOTE: Junos OS Release 23.4R1 and later supports multiple interface <interface_name>match conditions within a single firewall filter term. For example:</p> <pre>set firewall family any filter test term t1 from interface ge-0/0/0 set firewall family any filter test term t1 from interface ge-0/0/1 set firewall family any filter test term t1 from interface ge-0/0/2</pre> <p>NOTE: Junos OS Release 23.4R1 and later also allows you to configure this match condition alongside the vlan-id match condition in a single firewall filter term when you use Service Provider-style configuration on supported EX4400, EX4650, and QFX5120 switches. For example:</p> <pre>set firewall family any filter f1 term t1 from interface ge-0/0/0.1 set firewall family any filter f1 term t1 from vlan-id 2000</pre> <p>where VLAN 2000 is a VLAN created using Service Provider-style configuration. This is not supported if you use Enterprise-style configuration.</p> <p>For more information on Service Provider-style and Enterprise-style configuration, see <i>Flexible Ethernet Services Encapsulation</i>.</p>

Table 67: Match Conditions (Junos OS Release 22.4R1 and Later) *(Continued)*

Match Conditions	Description
vlan-id <vlan id> [<vlan_list>] <vlan_range>	<p>Match VLAN IDs.</p> <p>NOTE: Not supported on the EX4100 switches</p> <p>NOTE: Junos OS Release 23.4R1 and later supports the <vlan_list> and <vlan_range> options. For example:</p> <pre>set firewall family any filter test term t1 from vlan-id 2000-2100 set firewall family any filter test term t1 from vlan-id [3000 3010 3020]</pre> <p>NOTE: Junos OS Release 23.4R1 and later also allows you to configure this match condition alongside the interface match condition in a single firewall filter term when you use Service Provider-style configuration on supported EX4400, EX4650, and QFX5120 switches. This is not supported if you use Enterprise-style configuration.</p> <p>For more information on Service Provider-style and Enterprise-style configuration, see <i>Flexible Ethernet Services Encapsulation</i>.</p>

To ensure consistent application of policy, we recommend that you configure the same GBP tag assignments everywhere (at both the ingress and egress).

Example of Matching on MAC Address

Here's an example of GBP tag assignment using MAC addresses:

```
set firewall family any filter f1 micro-segmentation
set firewall family any filter f1 term tag100 from mac-address 00:00:5E:00:53:10
set firewall family any filter f1 term tag100 then gbp-tag 100
set firewall family any filter f1 term tag200 from mac-address 00:00:5E:00:53:20
set firewall family any filter f1 term tag200 then gbp-tag 200
set firewall family any filter f1 term tag300 from mac-address 00:00:5E:00:53:30
set firewall family any filter f1 term tag300 then gbp-tag 300
```

At the ingress in the above example, packets from MAC address 00:00:5E:00:53:10 are assigned tag 100, packets from MAC address 00:00:5E:00:53:20 are assigned tag 200, and packets from MAC address 00:00:5E:00:53:30 are assigned tag 300.

Example of Matching on IP Address

Here's an example of GBP tag assignment using IP addresses:

```
set firewall family any filter f2 micro-segmentation
set firewall family any filter f2 term t1 from ip-version ipv4 172.16.1.0/24
set firewall family any filter f2 term t1 then gbp-tag 400
```



NOTE: Matching on IP addresses is not supported in pure layer 2 deployments.



NOTE: Note that microsegmentation filters cannot have both IPv4 and IPv6 matches together as part of the same filter.

Example of Matching on VLANs and Interfaces

Here's an example of matching on a VLAN ID (not supported on EX4100 switches):

```
set firewall family any filter f1 micro-segmentation
set firewall family any filter f1 term t1 from vlan-id 100
set firewall family any filter f1 term t1 then gbp-tag 1
```

Starting in Junos OS Release 23.4R1:

- The EX4400, EX4650, and QFX5120 switches in ["Supported Platforms" on page 1278](#) support:
 - VLAN lists and ranges in a GBP filter
 - multiple VLAN entries in a single term in a GBP filter
 - multiple interface entries in a single term in a GBP filter
 - an interface and VLAN combination in a single term in a GBP filter when using Service Provider-style configuration. For more information on Service Provider-style and Enterprise-style configuration, see *Flexible Ethernet Services Encapsulation*.

- The EX4100 switches in ["Supported Platforms" on page 1278](#) support multiple interface entries in a single term in a GBP filter.

Here's an example of matching on VLAN ranges (not supported on EX4100 switches):

```
set firewall family any filter f1 micro-segmentation
set firewall family any filter f1 term t1 from vlan-id [10-30]
set firewall family any filter f1 term t1 then gbp-tag 100
```

Here's an example on matching on interfaces:

```
set firewall family any filter f2 micro-segmentation
set firewall family any filter f2 term t1 from interface xe-0/0/1
set firewall family any filter f2 term t1 from interface xe-0/0/2
set firewall family any filter f2 term t1 from interface xe-0/0/3
set firewall family any filter f2 term t1 from interface xe-0/0/4
set firewall family any filter f2 term t1 then gbp-tag 200
```

Assigning Tags Using RADIUS

You can configure your RADIUS server to assign GBP tags to users during authentication. RADIUS servers are commonly used in campus environments for access control and other functions such as the assignment of VLANs.



NOTE: Only MAC and interface-based GBP tag assignments are supported when using RADIUS:

- If you configure RADIUS authentication with single-secure or multiple supplicant mode, then GBP tagging is MAC-based.
- If you configure RADIUS authentication with single supplicant mode, then GBP tagging is interface-based.

To accommodate the distribution of GBP tags during authentication, we leverage the use of vendor specific attributes (VSAs), as supported by the AAA service framework. These VSAs are carried as part of the standard RADIUS request reply message, and provide a built-in extension to handle implementation-specific information such as our GBP tags. The exact syntax on the RADIUS server varies according to whether the authentication scheme is MAC or EAP-based.

For MAC-based clients, the configuration looks like this:

```
001094001199 Cleartext-Password := "001094001199"
Juniper-Switching-Filter = "apply action gbp-tag 100"
```

For EAP-based clients, the GBP tag is pushed from RADIUS server at the time of authentication. The configuration looks like this:

```
PermEmp01 Auth-Type = EAP, Cleartext-Password := "gbp"
Juniper-Switching-Filter = "apply action gbp-tag 100"
```

Starting with Junos OS Release 23.4R1, in addition to the existing Juniper-Switching-Filter, we support a new VSA called Juniper-Group-Based-Policy-Id on the EX4100, EX4400, EX4650, and QFX5120 switches.



NOTE: You should not use both the Juniper-Group-Based-Policy-Id VSA and the Juniper-Switching-Filter VSA together for the same client.

The client will not be authenticated if both VSAs exist and contain different GBP tag values.

You can assign GBP tags dynamically from RADIUS through either one of these VSAs:

- Juniper-Switching-Filter carries the GBP filter and other filter match and action conditions.
- The Juniper-Group-Based-Policy-Id carries only the GBP tag (available on EX4100, EX4400, EX4650, and QFX5120 switches only).

The Juniper-Group-Based-Policy-Id VSA for MAC and interface-based GBP tag filter looks like this:

```
PermEmp01 Auth-Type = EAP, Cleartext-Password := "gbp"
Juniper-Group-Based-Policy-Id = "100"
```

The configured GBP tag is a non-zero positive value in the range (1-65535) for GBP tags specified in a VSA (vendor specific attribute) from a RADIUS server.

Starting with Junos OS Release 23.4R1 and later, GBP feature support is also added to the following dot1x configuration statements on the EX4100, EX4400, EX4650, and QFX5120 switches:

Table 68: Dot1x Configuration Statements for GBP

CLI	Description
<pre>set protocols dot1x authenticator interface [interface-names] server-fail gbp-tag <i>gbp-tag</i></pre>	<p>Specify the GBP tag to apply on the interface when the server is inaccessible. If you configure the gbp-tag <i>gbp-tag</i> and the client authenticates in server-fail <i>vlan-name</i> or server-fail permit, then the configured gbp-tag <i>gbp-tag</i> filter is also installed for the client.</p> <p>You can only configure this option when the server-fail <i>vlan-name</i> or server-fail permit option is configured.</p>
<pre>set protocols dot1x authenticator interface [interface-names] server-reject-vlan gbp-tag <i>gbp-tag</i></pre>	<p>Specify the GBP tag to apply when RADIUS rejects the client authentication. If you configure the gbp-tag <i>gbp-tag</i> and the client authenticates in server-reject <i>vlan</i>, then the configured gbp-tag filter is also installed for the client.</p> <p>You can only configure the server-reject gbp-tag <i>gbp-tag</i> when the server-reject-vlan <i>vlan-id</i> option is configured.</p>
<pre>set protocols dot1x authenticator interface [interface-names] guest-gbp-tag <i>gbp-tag</i></pre>	<p>Specify the GBP tag to apply when an interface is moved to a guest VLAN. If the guest-gbp-tag is configured and the client authenticates in guest VLAN, then the configured guest-gbp-tag filter is also installed for the client.</p> <p>You can only configure the guest-gbp-tag when the guest-vlan <i>vlan-id</i> option is configured.</p> <p>For more information on guest VLANs, see <i>802.1X Authentication</i>.</p>

You can use the `show dot1x interface detail` or the `show ethernet-switching table` command to verify which GBP tag is received from RADIUS.

Here is example output from the `show dot1x interface detail` command:

```
root@access1> show dot1x interface mge-0/0/3 detail
mge-0/0/3.0
```

```

Role: Authenticator
Administrative state: Auto
Supplicant mode: Single
Number of retries: 3
Quiet period: 60 seconds
Transmit period: 30 seconds
Mac Radius: Enabled
Mac Radius Restrict: Enabled
Mac Radius Authentication Protocol: PAP
Reauthentication: Enabled
Reauthentication interval: 3600 seconds
Supplicant timeout: 30 seconds
Server timeout: 30 seconds
Maximum EAPOL requests: 2
Guest VLAN member: not configured
Number of connected supplicants: 1
  Supplicant: 525400cb93dd, 52:54:00:CB:93:DD
    Operational state: Authenticated
    Backend Authentication state: Idle
    Authentication method: Mac Radius
    Authenticated VLAN: vlan1099
    Dynamic Filter: apply action gbp-tag 300
    Session Reauth interval: 3600 seconds
    Reauthentication due in 2595 seconds
    Session Accounting Interim Interval: 36000 seconds
    Accounting Update due in 34995 seconds
    Eapol-Block: Not In Effect
    Domain: Data

```

Here is example output from the show ethernet-switching table command:

```
root@access2> show ethernet-switching table
```

```

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC,
          B - Blocked MAC)

```

```
Ethernet switching table : 2 entries, 2 learned
```

```
Routing instance : default-switch
```

Vlan	MAC	MAC	GBP	Logical	SVLBNH/	Active
name	address	flags	tag	interface	VENH Index	source

vlan200	00:10:94:00:00:05	D	100	xe-0/2/2.0	
vlan200	00:10:94:00:00:06	DR	100	vtep.32769	21.2.0.1

Tag Assignment Priority

IN THIS SECTION

- [Assignment Using Both CLI and RADIUS | 1292](#)
- [Conflicting Match Conditions | 1292](#)
- [Concurrent Matches | 1293](#)
- [Longest Prefix Match versus Strict Firewall Term Order | 1293](#)

This section clarifies the behavior when you create potentially ambiguous match conditions.

Assignment Using Both CLI and RADIUS

You typically configure tag assignment either using the CLI or using RADIUS, but not both. In the event that you do use both and there's a conflicting assignment (for example, CLI assigns GBP tag 10 to a user while RADIUS assigns GBP tag 20 to the same user), then the RADIUS assignment prevails.

Conflicting Match Conditions

Here's an example of a conflicting match condition:

```
set firewall family any filter f4 micro-segmentation
set firewall family any filter f4 term t1 from ip-version ipv4 address 172.16.0.0/24
set firewall family any filter f4 term t1 then gbp-tag 10
set firewall family any filter f4 term t2 from ip-version ipv4 address 172.16.0.0/24
set firewall family any filter f4 term t2 then gbp-tag 20
```

In the above example, both t1 and t2 match on IP address 172.16.0.0/24 but assign different GBP tags. In this situation, only the first matching term is evaluated. The second and subsequent matching terms are ignored. This is true regardless of whether you configure the filter for longest prefix match or not. Term t1 takes effect and any matching packet will be assigned GBP tag 10.

Concurrent Matches

When an incoming packet matches more than one condition, the match priorities shown in [Table 69 on page 1293](#) take effect:

Table 69: Match Priorities

Priority	Match Condition
1 (Highest)	ip-version ipv4 <ip address> <prefix-list>
2	ip-version ipv6 <ip address> <prefix-list>
3	mac-address <mac address>
4	interface <interface_name> vlan-id <vlan id>
5	vlan-id <vlan id>
6 (Lowest)	interface <interface_name>
<p>NOTE: If you enable MAC/IP inter-tagging, and an incoming packet matches on both the MAC and IP addresses, the result is indeterminate. The IP address match does not necessarily take priority over the MAC address match in that situation. See "GBP MAC/IP Inter-tagging" on page 1307.</p>	

Longest Prefix Match versus Strict Firewall Term Order

You have the option of choosing longest prefix match versus strict firewall term order. For example, here's a microsegmentation filter that contains overlapping IP addresses:

```
set firewall family any filter f3 micro-segmentation
set firewall family any filter f3 term t1 from ip-version ipv4 address 10.0.0.0/22
set firewall family any filter f3 term t1 then gbp-tag 10
set firewall family any filter f3 term t2 from ip-version ipv4 address 10.0.0.0/24
set firewall family any filter f3 term t2 then gbp-tag 20
```

Prior to Junos OS Release 24.4R1, an incoming packet with IP address 10.0.0.231 (for example) would be assigned GBP tag 10 because the incoming packet matches the first term (t1) in the filter.

However, starting in Junos OS Release 24.4R1, that same incoming packet would be assigned GBP tag 20 because the second term (t2) provides a more specific match.

If you want to retain the legacy behavior of strictly following firewall term order, then configure the filter as follows:

```
set firewall family any filter f3 no-longest-prefix-match
```



NOTE: Set the `no-longest-prefix-match` parameter when you first create the microsegmentation filter. Don't toggle this parameter on an existing microsegmentation filter.

Packet Classification

Packet classification in the context of GBP refers to retrieving the configured GBP tag during data plane processing of an incoming packet.

Packet classification occurs at both the ingress and the egress switch. At ingress, the switch classifies the incoming packet from the access link to obtain the GBP tag of the sender (source). At egress, the switch classifies the incoming packet from the network to obtain the GBP tag of the recipient (destination).

Once source and destination tags are retrieved, policy enforcement can take place.

Policy Enforcement

SUMMARY

Create microsegmentation policies using GBP tags.

IN THIS SECTION

- [Enforcement Using GBP Tags | 1295](#)
- [Enforcement Using Destination IP Address | 1295](#)
- [Enforcement Using L4 Fields | 1296](#)
- [Explicit Default Discard | 1297](#)
- [Policy Enforcement at the Ingress and Tag Propagation for EVPN-VXLAN | 1298](#)
- [Filter-Based Forwarding | 1302](#)
- [GBP MAC/IP Inter-tagging | 1307](#)

The GBP-enabled switch enforces policy based on rules that you create for the assigned GBP tags. In order to support GBP policy enforcement, we've extended regular firewall filter capability to include matching and acting on GBP tags. Regular firewall actions such as accept or deny are supported.

When running with EVPN-VXLAN, policy enforcement typically takes place at the egress using both the source and destination GBP tags. Ingress enforcement is also supported (see ["Policy Enforcement at the Ingress and Tag Propagation for EVPN-VXLAN" on page 1298](#)), but we recommend you use egress enforcement in high scale networks. Egress enforcement is the default behavior.

When running without EVPN-VXLAN, policy enforcement typically takes place at the ingress with either source GBP tag and destination IP address or with both source and destination GBP tags depending on your network. See ["Unified Access Policy" on page 1309](#) for more information.

Enforcement Using GBP Tags



NOTE: By default, policy enforcement is done at the egress. If you want to enforce policy at the ingress, see ["Policy Enforcement at the Ingress and Tag Propagation for EVPN-VXLAN" on page 1298](#).

Here's an example of GBP policy enforcement:

```
set firewall family any filter gbp-policy term t100-200 from gbp-src-tag 100
set firewall family any filter gbp-policy term t100-200 from gbp-dst-tag 200
set firewall family any filter gbp-policy term t100-200 then accept
set firewall family any filter gbp-policy term t100-300 from gbp-src-tag 100
set firewall family any filter gbp-policy term t100-300 from gbp-dst-tag 300
set firewall family any filter gbp-policy term t100-300 then discard
```

Packets with GBP source tag 100 and GBP destination tag 200 will match on term *t100-200* and be accepted. Packets with GBP source tag 100 and GBP destination tag 300 will match on term *t100-300* and be discarded.

Enforcement Using Destination IP Address

Starting in Junos OS Release 25.4R1, we allow you to enforce policy based on destination IP address when using the `gbp-pure-l2-profile`.

In pure L2 deployments, the access switches typically connect up to a WAN router as a layer 3 gateway. The WAN router is not GBP-aware and cannot pass GBP tags. Therefore, policy can only be enforced at

the ingress access switch based on the source GBP tag. In this situation, we allow you to augment the policy to include the destination IP address. For example:

```
set chassis forwarding-options gbp-pure-l2-profile
set firewall family any filter gbp-policy term t1 from gbp-src-tag 100
set firewall family any filter gbp-policy term t1 from ip-version ipv4 ip-destination-address
10.1.45.0/24
set firewall family any filter gbp-policy term t1 then accept
```

For more information on this use case, see ["Using the GBP Pure L2 Profile" on page 1316](#).

Enforcement Using L4 Fields

Starting in Junos OS Release 23.2R1, we've extended match conditions in GBP filters to include L4 matches ([Table 70 on page 1296](#)). This provides you with additional granularity to control application traffic.

Table 70: Support for Additional L4 Policy Matches (Junos OS Release 23.2R1 and Later)

Policy Enforcement Matches for MAC and IP GBP-Tagged Packets	Description
ip-version ipv4 destination-port <i>dst_port</i>	Match TCP/UDP destination port.
ip-version ipv4 source-port <i>src_port</i>	Match TCP/UDP source port.
ip-version ipv4 ip-protocol <i>ip-protocol</i>	Match IP protocol type.
ip-version ipv4 is-fragment	Match if the packet is a fragment.
ip-version ipv4 fragment-flags <i>flags</i>	Match the fragment flags (in symbolic or hex formats).
ip-version ipv4 ttl <i>value</i>	Match the MPLS/IP TTL value.
ip-version ipv4 tcp-flags <i>flags</i>	Match the TCP flags (in symbolic or hex formats) - (Ingress only).

Table 70: Support for Additional L4 Policy Matches (Junos OS Release 23.2R1 and Later) *(Continued)*

Policy Enforcement Matches for MAC and IP GBP-Tagged Packets	Description
<code>ip-version ipv4 tcp-initial</code>	Match the initial packet of a TCP connection - (Ingress only).
<code>ip-version ipv4 tcp-established</code>	Match the packet of an established TCP connection.
<code>ip-version ipv6 destination-port <i>dst_port</i></code>	Match the TCP/UDP destination port.
<code>ip-version ipv6 source-port <i>src_port</i></code>	Match the TCP/UDP source port.
<code>ip-version ipv6 next-header <i>protocol</i></code>	Match the next header protocol type.
<code>ip-version ipv6 tcp-flags <i>flags</i></code>	Match the TCP flags (in symbolic or hex formats)Ingress only.
<code>ip-version ipv6 tcp-initial</code>	Match the initial packet of a TCP connection.
<code>ip-version ipv6 tcp-established</code>	Match the packet of an established TCP connection.
NOTE: L4 filters are supported on the EX4100, EX4400, EX4650, and QFX5120 Series switches shown in "Supported Platforms" on page 1278 . These match conditions are not supported on the EX92xx switches.	
NOTE: L4 filters are supported by default but can reduce the supported GBP scale. To disable L4 filters on the EX4650, QFX5120-32C, and QFX5120-48Y switches: set <code>forwarding-options evpn-vxlan gbp tag-only-policy</code> . When you use this set (and corresponding delete) command, the Packet Forwarding Engine (PFE) restarts.	

Explicit Default Discard

When no conditions are matched, the default action is to accept the packet. Starting in Junos OS Release 24.2R1, you can specify an explicit default discard action for packets that don't match any conditions. See [Table 71 on page 1298](#).

Table 71: Explicit Default Discard Action (Junos OS Release 24.2R1 and Later)

Explicit Default Discard	Description
<pre>set firewall family any filter f1 term t1 from gbp- src-tag 100 set firewall family any filter f1 term t1 from gbp- dst-tag 200 set firewall family any filter f1 term t1 then accept set firewall family any filter f1 term t2 then discard</pre>	<p>You can create a filter term (for example, t2) that contains a discard action but no match conditions. This is useful as a catch-all for packets that do not match any of the conditions in the earlier terms in the sequence.</p> <p>This explicit default discard action does not apply to broadcast, multicast, host-originated, or unknown unicast packets. These types of traffic are always accepted.</p> <p>If you don't configure the explicit discard action, then the default action is to accept the packet as is the case in previous releases.</p>
<p>NOTE: Explicit default discard is supported on the EX4100, EX4400, EX4650, and QFX5120 Series switches shown in "Supported Platforms" on page 1278. Explicit default discard is not supported on the EX92xx switches or MX Series routers</p>	

Policy Enforcement at the Ingress and Tag Propagation for EVPN-VXLAN

IN THIS SECTION

- [Tag Propagation for IP Prefix Routes Using EVPN Type 5 Advertisements](#) | 1299

Starting with Junos Release 22.4R1, you can perform policy enforcement closer to the ingress. Ingress enforcement saves network bandwidth by discarding tagged packets at the ingress that would otherwise be discarded at the egress. To support policy enforcement at or closer to the ingress, we propagate the MAC and IP-MAC based tags across the network using extended BGP communities within EVPN Type 2 and Type 5 routes. See ["EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN" on page 65](#) for information on these types of routes.

EVPN route advertisement is triggered by the installation (or a change) of an EVPN route, such as through MAC-IP learning when receiving a packet from a new host. In this case, the source IP route is

installed in the evpn.0 database and an EVPN Type 2 advertisement (that includes the GBP tag if assigned) is sent to all eBGP peers.

After these advertisements propagate through the network to the remote endpoints, the remote endpoints have sufficient information to make GBP firewall filter decisions on packets received at the remote ingress. When packets are received at their ingress, the remote endpoints can look up the destination route and obtain the destination GBP tag previously received through the EVPN Type 2 advertisement. Armed with the destination GBP tag, the remote endpoints can subsequently make GBP policy enforcement decisions on their ingress packets.

Since GBP tags are propagated using EVPN Type 2 route advertisements, tag propagation is necessarily performed per MAC or IP address. This has no bearing on tag assignment, however, which can continue to be any of the supported methods, such as VLAN or interface, among others.

For example, if you configure tag assignment based on interface, and a packet from a new host is received on that interface, then the tag assigned for that interface is propagated in a Type 2 route advertisement along with the source MAC and IP address of the incoming packet. If a packet from a different host is subsequently received on that same interface, then the same tag is propagated in another Type 2 route advertisement along with the source MAC and IP address of this different host.



NOTE: If a border leaf switch receives an EVPN Type 2 advertisement with a GBP tag, the switch installs the Type 2 route and generates an EVPN Type 5 advertisement with that GBP tag to its eBGP peers such as to the border leaf switches in other data centers (for inter-DC traffic). This Type 5 route contains a /32 IP address and a GBP tag. This Type 2 to Type 5 GBP tag propagation is supported but Type 5 to Type 2 GBP tag propagation is not supported.

For multihoming topologies, keep the configuration identical across multihoming members.

You must enable the following statement to perform the policy enforcement at the ingress node. When ingress enforcement is enabled or disabled, the Packet Forwarding Engine (PFE) restarts.

```
set forwarding-options evpn-vxlan gbp ingress-enforcement
```

Tag Propagation for IP Prefix Routes Using EVPN Type 5 Advertisements

Starting in Junos OS Release 24.2R1, we support GBP tag propagation for IP prefix routes using EVPN Type 5 advertisements. Previous to this release, GBP tag propagation was only triggered by MAC-IP learning in the dataplane, which meant that tag propagation only occurred for /32 IP routes.

With support for IP prefix routes, tag propagation can now occur, for example, when you create an interface and enable the advertisement of direct EVPN routes (set routing-instances *<instance>* protocols evpn ip-prefix-routes advertise direct-nexthop). If you also assign a GBP tag to that IP prefix, then the subsequent EVPN Type 5 advertisement includes the GBP tag, thereby propagating the tag even before MAC-IP learning takes place.

In general, GBP tag propagation within EVPN Type 5 advertisements occurs whenever you create a GBP filter that assigns a tag to an IP prefix and that IP prefix route is installed in the evpn.0 routing database. (You can create the GBP filter before or after the route is installed.)

Even though the switch generates a Type 5 advertisement, if the switch learns of a new host (for example, through MAC-IP learning in the dataplane), the switch will generate a Type 2 advertisement as well. It may be desirable in many instances to suppress these redundant /32 advertisements to reduce EVPN traffic. To do so, create a BGP policy to reject /32 routes.

For example, the following creates a policy called T5_EXPORT with term called fm_v4_host that rejects /32 routes from IPv4 hosts:

```
set policy-options policy-statement T5_EXPORT term fm_v4_host from route-filter 0.0.0.0/0 prefix-length-range /32-/32
set policy-options policy-statement T5_EXPORT term fm_v4_host then reject
```



NOTE: If a switch receives an EVPN advertisement for an IP prefix route and associated GBP tag, and if you've configured a GBP filter that assigns a different tag to that same IP prefix route, the GBP tag in the locally-configured GBP filter prevails. The switch replaces the GBP tag in the received EVPN advertisement with the locally-assigned GBP tag before re-advertising the EVPN route.

IP prefix tag propagation is automatically enabled when you create a GBP filter for an IP prefix and associate the GBP filter to a routing instance. For example:

```
set firewall family any filter f1 micro-segmentation
set firewall family any filter f1 term tag300 from ip-version ipv4 172.16.1.0/24
set firewall family any filter f1 term tag300 then gbp-tag 300
set routing-instances <routing-instance> forwarding-options evpn-vxlan gbp ingress-src-tag filter f1
```

where *<routing-instance>* is the name of the routing instance that you want the filter to apply to.

Once an IP prefix route is associated with a GBP tag, the GBP tag is displayed in the output of the show route commands for that IP prefix route. For example:

```
show route match-prefix 5:* extensive

bgp.evpn.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
Restart Complete
5:10.255.1.1 (1 entry, 1 announced)
<trimmed>
gbp-tag:0L:53 router-mac:52:54:00:00:92:f0
      Import Accepted
      Route Label: 9100
      Overlay gateway address: 0.0.0.0
      ESI 00:00:00:00:00:00:00:00:00:00
      Localpref: 100
      Router ID: 10.255.1.1
      Secondary Tables: VRF-100.evpn.0
      Thread: junos-main
      Indirect next hops: 1
<trimmed>
```

```
show route table VRF-100.inet.0 match-prefix 10.1.1* extensive

VRF-100.inet.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
Restart Complete
10.1.1.3/32 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.1.1.3/32 -> {indirect(1048574)}
Opaque data client: EVPN-Type5
Opaque data: tlv_type :32820
Type5 gbp_tag 53.
Address: 0xa15f928
Opaque-data reference count: 2
Page 0 idx 0, (group DC2_TORS type External) Type 1 val 0xa32fd40 (adv_entry)
  Advertised metrics:
    Nexthop: Self
    AS path: [65201] 65200 I
    Communities: gbp-tag:0L:53
  Advertise: 00000001
```

```
Path 10.1.1.3
<trimmed>
```

To see the binding between a routing instance and a GBP filter, use the `show evpn gbp-src-tag filter-bind routing-instance` command.

To see the IP prefix route to GBP tag mapping, use the `show evpn gbp-src-tag ip-prefix inet` command.

Limitations of this feature include the following:

- You can only associate a GBP filter to one routing instance. You cannot associate the same GBP filter to multiple routing instances.
- You cannot associate two different GBP filters with the same IP prefix match condition to the same routing instance.
- You can only associate an IP-based GBP filter to a routing instance. Associating other types of GBP filters has no effect.

Filter-Based Forwarding

Filter-based forwarding refers to the ability to forward traffic to a specified next hop if the GBP tags assigned to that traffic match the GBP tags specified in the filter. Use this feature to apply different routing treatment for the specified tagged traffic versus regular traffic.

To create a forwarding filter, specify the source and destination tags that you want to match and the next hop where you want to forward the matched traffic.

The CLI statements for filter-based forwarding are different for different products:

- Filter-based forwarding of GBP-tagged traffic on the EX4xxx Series and QFX Series Switches in ["Supported Platforms" on page 1278](#) is supported starting in Junos OS release 24.4R1. See [Table 72 on page 1303](#) for CLI configuration examples.
- Filter-based forwarding of GBP-tagged traffic on the EX92xx Series Switches and the MX Series Routers in ["Supported Platforms" on page 1278](#) is supported starting in Junos OS release 25.2R1. See [Table 73 on page 1305](#) CLI configuration examples.

Table 72: Filter-Based Forwarding Examples of GBP-Tagged Traffic (EX4xxx Series and QFX Series Switches)

Filter-Based Forwarding Configuration Examples (EX4xxx Series and QFX Series Switches)	Description
<pre>set firewall family any filter f1 term t1 from gbp- src-tag 100 set firewall family any filter f1 term t1 from gbp- dst-tag 200 set firewall family any filter f1 term t1 then next- ip 10.10.1.1</pre>	<p>Use the default routing instance to forward traffic with GBP source tag 100 and destination tag 200 to the next hop for the 10.10.1.1 route.</p>
<pre>set firewall family any filter f1 term t1 from gbp- src-tag 100 set firewall family any filter f1 term t1 from gbp- dst-tag 200 set firewall family any filter f1 term t1 then next- ip 10.10.1.0/24</pre>	<p>Use the default routing instance to forward traffic with GBP source tag 100 and destination tag 200 to the next hop for the 10.10.1.0/24 route.</p>
<pre>set firewall family any filter f1 term t1 from gbp- src-tag 100 set firewall family any filter f1 term t1 from gbp- dst-tag 200 set firewall family any filter f1 term t1 then next- ip 10.10.1.1 routing-instance VRF-100</pre>	<p>Use the VRF-100 routing instance to forward traffic with GBP source tag 100 and destination tag 200 to the next hop for the 10.10.1.1 route.</p>
<pre>set firewall family any filter f1 term t1 from gbp- src-tag 100 set firewall family any filter f1 term t1 from gbp- dst-tag 200 set firewall family any filter f1 term t1 then next- ip6 2001:db8:4136:e378:8000:63bf:3fff:fdd2</pre>	<p>Use the default routing instance to forward traffic with GBP source tag 100 and destination tag 200 to the next hop for the 2001:db8:4136:e378:8000:63bf:3fff:fdd2 route.</p>

Table 72: Filter-Based Forwarding Examples of GBP-Tagged Traffic (EX4xxx Series and QFX Series Switches) (Continued)

Filter-Based Forwarding Configuration Examples (EX4xxx Series and QFX Series Switches)	Description
<pre>set firewall family any filter f1 term t1 from gbp-src-tag 100 set firewall family any filter f1 term t1 from gbp-dst-tag 200 set firewall family any filter f1 term t1 then next-ip6 2001:db8:4136::/48</pre>	<p>Use the default routing instance to forward traffic with GBP source tag 100 and destination tag 200 to the next hop for the 2001:db8:4136::/48 route.</p>
<pre>set firewall family any filter f1 term t1 from gbp-src-tag 100 set firewall family any filter f1 term t1 from gbp-dst-tag 200 set firewall family any filter f1 term t1 then next-ip6 2001:db8:4136:e378:8000:63bf:3fff:fdd2 routing-instance VRF-100</pre>	<p>Use the VRF-100 routing instance to forward traffic with GBP source tag 100 and destination tag 200 to the next hop for the 2001:db8:4136:e378:8000:63bf:3fff:fdd2 route.</p>
<p>Limitations:</p> <ul style="list-style-type: none"> • Filter-based forwarding is only supported for tags assigned using IPv4/IPv6 address match conditions. It is not supported for tags assigned using other match conditions such as MAC address, interface, VLAN, and interface+VLAN. • No checks are made regarding the validity or presence of the next hop. If the next hop is not in the routing table or otherwise not resolved or if the outbound interface is down, filter-based forwarding will not take place and the packet will get dropped. • The only action that can coexist with next-ip and next-ip6 is count. For example: <pre>set firewall family any filter f1 term t1 from gbp-src-tag 100 set firewall family any filter f1 term t1 from gbp-dst-tag 200 set firewall family any filter f1 term t1 then count num_100_200_packets set firewall family any filter f1 term t1 then next-ip 10.10.1.1</pre>	

Table 73: Filter-Based Forwarding Examples of GBP-Tagged Traffic (EX92xx Series Switches and MX Series Routers)

Filter-Based Forwarding Configuration Examples (EX92xx Series Switches and MX Series Routers)	Description
<pre>set firewall family any filter f1 term t1 from ip- version ipv4 set firewall family any filter f1 term t1 from gbp- src-tag 100 set firewall family any filter f1 term t1 from gbp- dst-tag 200 set firewall family any filter f1 term t1 then next- ip 10.10.1.1/32</pre>	<p>Forward traffic with GBP source tag 100 and destination tag 200 to the next hop for the 10.10.1.1/32 route.</p>
<pre>set firewall family any filter f1 term t1 from ip- version ipv6 set firewall family any filter f1 term t1 from gbp- src-tag 100 set firewall family any filter f1 term t1 from gbp- dst-tag 200 set firewall family any filter f1 term t1 then next- ip6 2001:db8:4136:e378:8000:63bf:3fff:fdd2/128</pre>	<p>Forward traffic with GBP source tag 100 and destination tag 200 to the next hop for the 2001:db8:4136:e378:8000:63bf:3fff:fdd2/128 route.</p>
<p>Limitations:</p> <ul style="list-style-type: none"> • Only /32 IPv4 and /128 IPv6 addresses are supported as next hops. • No checks are made regarding the validity or presence of the next hop. If the next hop is not in the routing table or otherwise not resolved or if the outbound interface is down, filter-based forwarding will not take place and the packet will get dropped. • The actions that can coexist with next-ip and next-ip6 are count, policer, and accept. For example: <pre>set firewall family any filter f1 term t1 from ip-version ipv4 set firewall family any filter f1 term t1 from gbp-src-tag 100 set firewall family any filter f1 term t1 from gbp-dst-tag 200 set firewall family any filter f1 term t1 then count num_100_200_packets set firewall family any filter f1 term t1 then policer CIR-100 set firewall family any filter f1 term t1 then next-ip 10.10.1.1/32</pre>	

For EX92xx Series Switches and MX Series Routers, since you don't specify the routing table directly in the filter, you need to explicitly attach the forwarding filter to a routing table. See [Table 74 on page 1306](#) for CLI configuration examples.

Table 74: Attaching a Forwarding Filter Examples (EX92xx Series Switches and MX Series Routers)

Attaching the Forwarding Filter Configuration Examples (EX92xx Series Switches and MX Series Routers)	Description
<pre>set routing-instances VRF-100 forwarding-options family inet filter output f1</pre>	Apply the f1 forwarding filter after the route lookup in the VRF-100 IPv4 routing table.
<pre>set routing-instances VRF-100 forwarding-options family inet6 filter output f1</pre>	Apply the f1 forwarding filter after the route lookup in the VRF-100 IPv6 routing table.
<p>NOTE: You can only attach the forwarding filter to a routing instance of type vrf or virtual-router. You cannot attach the forwarding filter to other routing instances or to other attachment points.</p>	

[Table 75 on page 1306](#) summarizes the filter-based forwarding platform-specific behaviors for your platforms.

Table 75: Platform-Specific Behavior

Platform	Difference
EX4xxx Series	<ul style="list-style-type: none"> The next hop in the filter can be a prefix route. Specify the applicable routing instance directly in the filter.
EX92xx Series	<ul style="list-style-type: none"> The next hop in the filter must be a /32 IPv4 or /128 IPv6 route. Specify the applicable routing instance by explicitly attaching the filter to the applicable routing instance.

Table 75: Platform-Specific Behavior (Continued)

Platform	Difference
MX Series	<ul style="list-style-type: none"> • The next hop in the filter must be a /32 IPv4 or /128 IPv6 route. • Specify the applicable routing instance by explicitly attaching the filter to the applicable routing instance.
QFX Series	<ul style="list-style-type: none"> • The next hop in the filter can be a prefix route. • Specify the applicable routing instance directly in the filter.

GBP MAC/IP Inter-tagging

Prior to Junos OS Release 24.2R1, a MAC-based GBP filter only applied to switched traffic, and an IP-based GBP filter only applied to routed traffic.

Starting in Junos OS Release 24.2R1, MAC-based GBP filters can also apply to routed traffic, and IP-based GBP filters can also apply to switched traffic. This is called MAC/IP inter-tagging and can be enabled on the specific EX4100, EX4400, EX4650, and QFX5120 series switches shown in ["Supported Platforms" on page 1278](#).

Starting in Junos OS Release 25.4R1, MAC/IP inter-tagging is enabled by default. You don't need to explicitly configure it.

When enabled, the switch automatically adds a corresponding entry as follows:

- If you create a MAC-based GBP filter, the first packet that arrives matching that MAC address will cause the switch to automatically create a corresponding IP-based GBP assignment. This assignment will apply the same tag as the original MAC-based GBP filter but match on the source IP address instead of the MAC address.
- If you create an IP-based GBP filter, the first packet that arrives matching that IP address will cause the switch to automatically create a corresponding MAC-based GBP assignment. This assignment will apply the same tag as the original IP-based GBP filter but match on the source MAC address instead of the IP address.

Creating a corresponding entry ensures that the desired GBP tag is assigned regardless of whether the flow is subsequently switched or routed.



NOTE: If you enable this feature, be careful not to create conflicting GBP tag assignments. For example, if you enable this feature and you create two filters as follows:

- Filter 1: assign GBP tag 100 to traffic with MAC address 52:54:00:00:00:11
- Filter 2: assign GBP tag 200 to traffic with IP address 172.16.0.11

then make sure that the switch never receives traffic with that MAC/IP combination. Tagging behavior is indeterminate if the switch receives a packet with MAC address 52:54:00:00:00:11 and IP address 172.16.0.11 when you create the above filters and enable MAC/IP inter-tagging.

To enable MAC/IP inter-tagging:

```
set forwarding-options evpn-vxlan gbp mac-ip-inter-tagging
```



NOTE: You don't need to configure MAC/IP inter-tagging starting in Junos OS Release 25.4R1. It is enabled by default.



NOTE: If you set or delete the mac-ip-inter-tagging option, the Packet Forwarding Engine (PFE) restarts automatically.



NOTE: If you set or delete the mac-ip-inter-tagging option in a virtual chassis, you must reboot all members of the virtual chassis: `request system reboot all-members`

Below you can see the same GBP tag 100 appear in both the MAC and IP tables when you enable MAC/IP inter-tagging.

```
show ethernet-switching table
```

```
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, B - Blocked MAC)
```

```
Ethernet switching table : 2 entries, 2 learned
```

```
Routing instance : default-switch
```

Vlan	MAC	MAC	GBP	Logical	SVLBNH/
Active					
name	address	flags	tag	interface	VENH Index
source					
vlan100	52:54:00:22:22:22	D	100	xe-0/0/8.0	
vlan200	52:54:00:44:44:44	D		xe-0/0/9.0	

```
show ethernet-switching mac-ip-table
```

MAC IP flags (S - Static, D - Dynamic, L - Local , R - Remote, Lp - Local Proxy, Rp - Remote Proxy, K - Kernel, RT - Dest Route, (N)AD - (Not) Advt to remote, RE - Re-ARP/ND, RO - Router, OV - Override, Ur - Unresolved, B - Blocked, RTS - Dest Route Skipped, RGw - Remote Gateway, GBP - Group Based Policy, RTF - Dest Route Forced, SC - Static Config, P - Probe, NLC - No Local Config, LD - Local Down)

Routing instance : default-switch

Bridging domain : vlan100

IP	MAC	Flags	GBP
Logical	Active		
address	address		Tag
Interface	source		
10.100.100.100	52:54:00:22:22:22	DL,K,RT,AD	100 xe-0/0/8.0
10.100.100.101	52:54:00:63:0e:40	S,K	irb.100
2001:db8::9300:6463:e40	52:54:00:63:0e:40	S,K	irb.100

Unified Access Policy

SUMMARY

Unified access policy extends GBP support to Mist access points (APs).

IN THIS SECTION

- [GBP Messages | 1310](#)
- [UAP Access Links | 1311](#)
- [UAP Inter-Switch Links | 1312](#)
- [UAP Configuration | 1314](#)

Starting in Junos OS Release 25.4R1, we support unified access policy for wired and wireless clients on the EX4100, EX4400, EX4650, and QFX5120 switches listed in ["Supported Platforms" on page 1278](#).

Unified access policy enables supported switches to learn GBP tags from Mist APs, allowing both wired and wireless clients to participate in GBP microsegmentation.

GBP Messages

When you enable unified access policy on a supported switch, the switch learns GBP tag assignments from attached Mist APs through proprietary GBP messages.

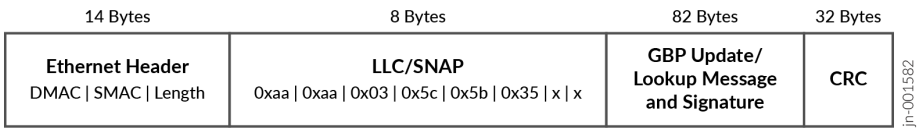
[Table 76 on page 1310](#) shows the two types of GBP messages:

Table 76: GBP Messages

GBP Message Type	Description
GBP update	<p>A GBP update message contains the mapping between a MAC address and a GBP tag. This can be a solicited or unsolicited message from the sender to convey a tag assignment to the receiver.</p> <p>This can be a unicast or multicast message.</p>
GBP lookup	<p>A GBP lookup message is a request by the sender to obtain the GBP tag assignment from the receiver for the MAC address specified within the message.</p> <p>This is a unicast message.</p>
<p>NOTE: GBP update messages contain MAC-based GBP tag assignments only and GBP lookup messages request MAC-based GBP tag assignments only.</p>	

[Figure 123 on page 1311](#) shows the format of the GBP messages. The messages are signed with a key to detect tampering.

Figure 123: GBP Message Format



The following sections describe how these messages are used.

UAP Access Links

A UAP access link refers to the link that connects a UAP-enabled access switch directly or indirectly to a Mist AP. The Mist AP conveys GBP tag assignments across this link to the access switch through a GBP message exchange.

Specifically, the Mist AP and the access switch behave as follows:

- If a wireless client is assigned a GBP tag during authentication, the Mist AP sends a multicast GBP update message containing the MAC-based GBP tag assignment to the upstream access switch. The destination multicast MAC address in the message is set to 5d:5b:35:ff:ff:01 and the source MAC address is set to the MAC address of the Mist AP.

When the access switch receives this GBP update message, the switch updates its MAC address tables and internal data structures with the specified MAC-based tag assignment.

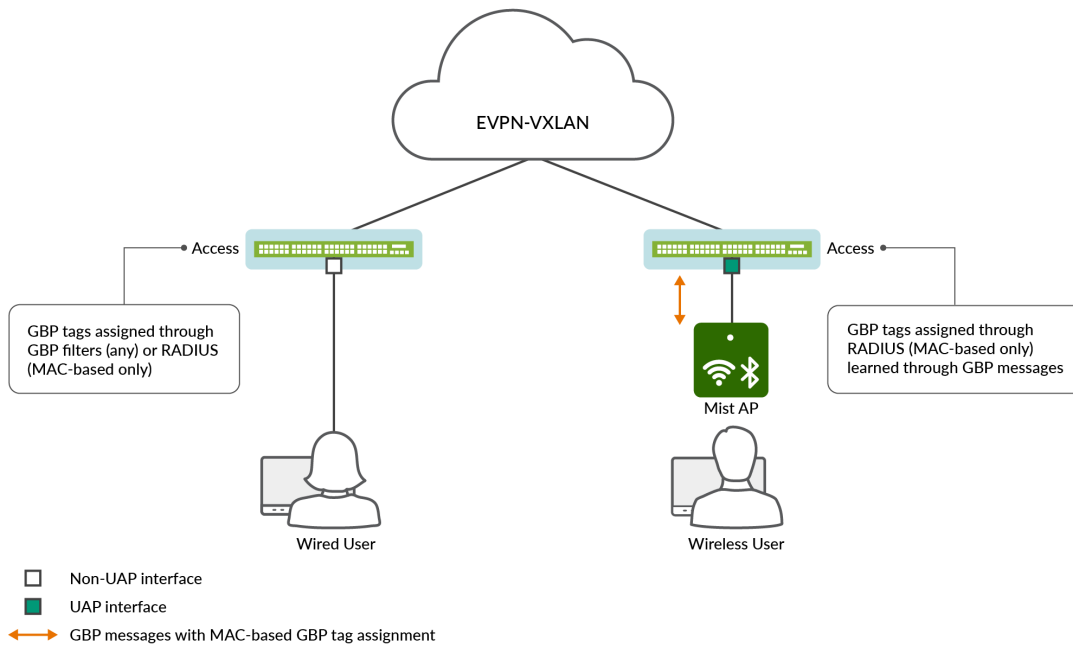
- If the GBP update message is dropped or if the access switch reboots, the tag assignment is lost. In this situation, if the access switch receives a client data frame with a source MAC address that has no GBP tag assigned, the access switch sends a GBP lookup message to the Mist AP asking for the missing tag assignment. The destination MAC address in the GBP lookup message is set to the MAC address of the Mist AP. The source MAC address is set to a derivation of the chassis MAC address of the access switch.

When the Mist AP receives this GBP lookup message, the Mist AP looks up the specified MAC address to obtain the GBP tag. If the GBP tag exists, the Mist AP responds to the GBP lookup message with a unicast GBP update message containing the tag assignment. The destination unicast MAC address in the response is set to the MAC address of the access switch (a derivation of the chassis MAC address). The source MAC address is set to the MAC address of the Mist AP.

[Figure 124 on page 1312](#) shows unified access policy in an EVPN-VXLAN network connecting wired and wireless clients. GBP tags for wired clients are assigned through GBP filters or RADIUS authentication as is usual. GBP tags for wireless clients are assigned through RADIUS authentication and the tag assignments are communicated to upstream switches in GBP update messages. The switch receiving the GBP update messages configures its MAC address tables and internal data structures with the GBP tag assignments as if the GBP tags were assigned through the CLI or learned locally. Regular

GBP tag processing then occurs. There is no restriction or limitation in functionality due to the tag being assigned through a GBP message instead of the CLI.

Figure 124: Unified Access Policy in an EVPN-VXLAN Network



Although the above example shows UAP in an EVPN-VXLAN network, you can configure UAP in a pure L2 network as well. The UAP capability works independently from the underlying infrastructure.



NOTE: Be careful not to use the CLI to assign GBP tags that conflict with GBP tags learned through GBP update messages. Unpredictable behavior occurs if the GBP tag that you assign to a MAC address through the CLI is different from the GBP tag that is learned for that same MAC address through GBP update messages.

UAP Inter-Switch Links

While a UAP access link connects an access switch to a Mist AP, a UAP inter-switch link connects UAP-enabled switches with each other in a pure layer 2 network. When you configure an interface for a UAP inter-switch link, you're allowing GBP messages to be exchanged on that interface. In this way, MAC-based GBP tags are propagated across the network to all UAP-enabled switches.



NOTE: You must enable LLDP in order for GBP messages to be sent and received across inter-switch links. LLDP allows neighboring UAP-enabled switches to discover each other.

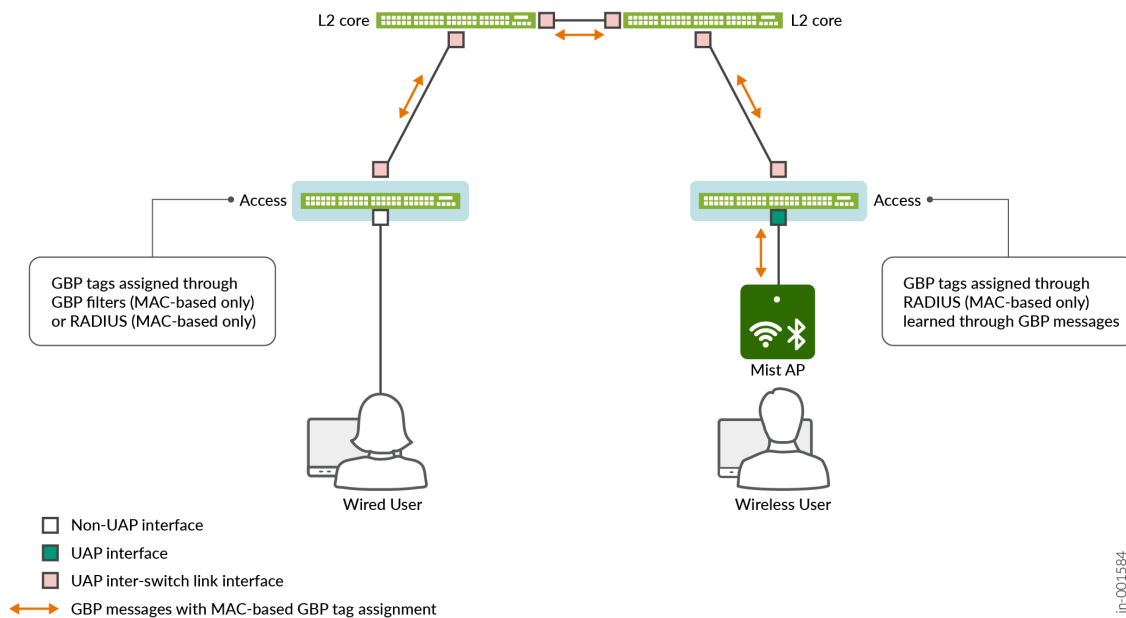
Specifically, the UAP-enabled switches behave as follows:

- **Wireless** - If an access switch receives a GBP update message from a Mist AP, the access switch processes the message as described in ["UAP Access Links" on page 1311](#) and passes this multicast message out all inter-switch link interfaces. This message then propagates across all UAP-enabled switches where this tag assignment is learned.
- **Wired** - If an access switch learns of a GBP tag assignment for a wired user (for example, through CLI configuration or RADIUS authentication), it creates a unicast GBP update message containing that tag assignment and sends it to its UAP-enabled neighbors across the inter-switch links.

Passing GBP update messages across the network in this manner allows access switches to enforce policy on both the source and destination GBP tags at the ingress. This feature, however, is limited to UAP-enabled switches in a pure L2 network.

[Figure 125 on page 1314](#) shows how GBP messages propagate over inter-switch links across a pure layer 2 network. GBP tags from both wired and wireless clients are propagated in this manner.

Figure 125: Unified Access Policy in a Pure Layer 2 Network



jn-001584

UAP Configuration

Use this procedure to configure UAP.

1. Enable unified access policy by configuring the key used to sign and verify GBP messages.

This key is used to sign and verify:

- GBP messages sent to and received from the Mist APs, and
- GBP messages sent and received on the UAP inter-switch links

```
set protocols unified-access-policy key "<key>"
```

where *<key>* is the key used to sign and verify GBP messages. This *<key>* must match the key you configure on the Mist APs and all the UAP-enabled switches.

2. Specify the interfaces where the switch sends and receives GBP messages.

There are two types of interfaces. Both can exist at the same time on the same switch.

- a. Specify the interfaces that connect to the Mist APs either directly or through switches that don't support unified access policy or through switches where unified access policy is not enabled.

```
set protocols unified-access-policy interface <if-name>
```

where *<if-name>* is the name of the interface that faces towards the Mist APs.

- b. Specify the UAP inter-switch link interfaces. These interfaces connect UAP switches together in a pure layer 2 network.



NOTE: Configure UAP inter-switch link interfaces only for pure layer 2 networks.

```
set protocols unified-access-policy interface <if-name> inter-switch-link
```

where *<if-name>* is the name of the inter-switch link interface.

Repeat this step to cover all applicable interfaces.

3. Configure the source MAC address to use in GBP messages if you have a multi-homed deployment. By default, the UAP-enabled switch uses a source MAC address that is derived from the chassis MAC address. This results in each UAP-enabled switch having a different source MAC address, which is the desired behavior in a single-homed deployment.

In a multi-homed deployment, however, where a UAP-enabled access switch is connected to two UAP-enabled upstream switches, we want both connected upstream switches to appear as one for the purpose of the GBP message exchange. In this latter situation, configure the same source MAC address on both connected upstream switches as follows:

```
set protocols unified-access-policy source-mac <source-address>
```

where *<source-address>* is the Ethernet source address you want to use for GBP messages in this multi-homed scenario.

4. Optionally, set the number of attempts to send the GBP lookup messages.

```
set protocols unified-access-policy robust-count <count>
```

where *<count>* is the number of times to send the lookup message if there is no response. The default count is 3 (at non-configurable one-second intervals).

5. After committing your configuration, check that you've configured the interfaces as intended.

For example:

```
show unified-access-policy
```

```
Unified Access Policy Status : Enabled
```

```
Destination Mac : 5d-5b-35-ff-ff-01
```

```
Chassis Mac : 00-00-5E-00-53-01
```

```
Robust Count : 3
```

Interface	Link-Type	Intf-Status	UAP-LLDP Neighbor Count
ge-0/0/5	Inter Switch Link	UP	0
ge-0/0/8	Access Point	UP	0

6. Repeat for all UAP-enabled switches.

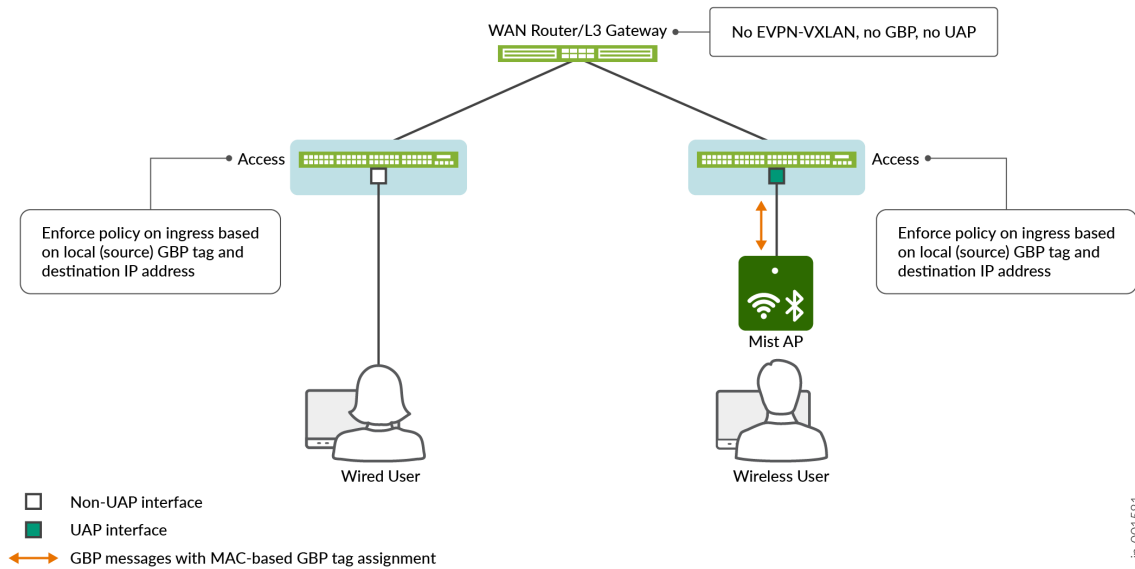
Using the GBP Pure L2 Profile

In some deployments, you may want to perform policy enforcement on access switches that are not L3 gateways. Perhaps you want to keep your access switches at layer 2 for simplicity and move the routing and gateway responsibility to a WAN router. This means that the access switches do not perform IP route lookups and cannot participate in L3 segmentation.

In this situation, if you enable the `gbp-pure-l2-profile`, we allow you to add a destination IP address/subnet match to your policy. This gives you the ability to support both L2 and L3 segmentation even on a pure L2 access switch.

[Figure 126 on page 1317](#) shows GBP-enabled layer 2 access switches connected to a WAN router. The WAN router does not support EVPN-VXLAN and does not support GBP. However, even in this scenario, the access switch can still participate in L2 and L3 segmentation if you enable the `gbp-pure-l2-profile` and enforce policy on the combination of a local (source) GBP tag and a destination IP address/subnet.

Figure 126: Microsegmentation Using GBP Pure L2 Profile



This feature allows you to incorporate third-party WAN and core/distribution routers into your network and continue to support GBP microsegmentation. This is ideal for smaller sites where policy enforcement at the access layer simplifies design and reduces dependency on core infrastructure.

[Table 77 on page 1317](#) and [Table 78 on page 1318](#) show the supported GBP tag assignment and policy enforcement features with the `gbp-pure-l2-profile`. A pure layer 2 device only supports a subset of the tagging and enforcement features.

Table 77: GBP Tag Assignment with the GBP Pure L2 Profile

Tag Assignment Based On ...	Support
MAC address only	Yes
Interface only	Yes
VLAN only	Yes ¹
Interface and VLAN	Yes ¹
IP address	No

Table 77: GBP Tag Assignment with the GBP Pure L2 Profile *(Continued)*

Tag Assignment Based On ...	Support
IEEE 802.1X	Yes
¹ Not supported on EX4100 switches.	

Table 78: GBP Policy Enforcement with the GBP Pure L2 Profile

Enforcement	Support
Egress	Yes, with destination GBP tag
Ingress	Yes, with source GBP tag and destination IP address
L4 fields	Yes
Ingress with Tag Propagation	No
Explicit Default Discard	No
MAC/IP Inter-tagging	No
Filter-Based Forwarding	No

Below is an example of configuring GBP with the gbp-pure-l2-profile.



NOTE: Although this network use case does not require EVPN-VXLAN, you'll still need to create a VTEP (for historical reasons). Create the VTEP prior to starting this procedure. Here's an example of VTEP configuration. Change this to suit your network.

```
set routing-options router-id 10.1.2.3
set interfaces lo0 unit 0 family inet address 10.1.2.3/32
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
```

```
set switch-options route-distinguisher 10.1.2.3:100
set switch-options vrf-target target:1:100
```

For all VLANS:

```
set vlans default vxlan vni 100001
set vlans default no-arp-suppression
set vlans vlan1099 vxlan vni 101099
set vlans vlan1099 no-arp-suppression
etc.
```

1. Configure the access switches for the gbp-pure-l2-profile.

```
set chassis forwarding-options gbp-pure-l2-profile
```

This not only sets the UFT table sizes appropriately for a pure L2 deployment, but it also allows you to create a policy enforcement filter that includes a destination IP address match (as shown in step 3).

2. Configure UAP on the access switch connecting to the Mist AP. See ["UAP Configuration" on page 1314](#).
3. Create a GBP policy enforcement filter.

Augment the filter with a destination IPv4 address match. For example:

```
set firewall family any filter gbp-policy term t1 from gbp-src-tag 100
set firewall family any filter gbp-policy term t1 from ip-version ipv4 ip-destination-address 192.168.0.27
set firewall family any filter gbp-policy term t1 then discard
```

Example: Using GBP to Segment Traffic

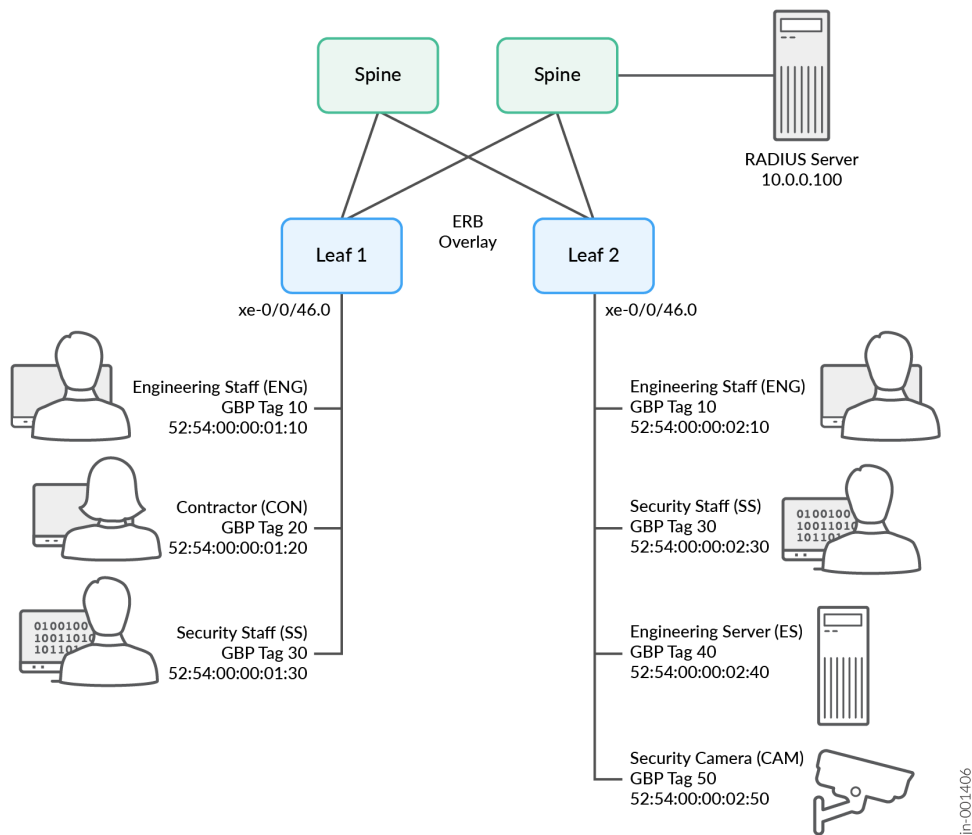
SUMMARY

This basic example shows how you can use GBP filters and policies to segment traffic.

This example shows how to configure a pair of switches for GBP-based microsegmentation. The switches (Leaf 1 and Leaf 2) control access for the following users and equipment, as shown in [Figure 127 on page 1320](#):

- 2 Engineering Staff (ENG)
- 1 Contractor (CON)
- 2 Security Staff (SS)
- 1 Engineering Server (ES)
- 1 Security Camera (CAM)

Figure 127: GBP Example



[Table 79 on page 1321](#) shows the GBP tag values that we'll assign. We'll show how to assign these tags using both the CLI and RADIUS.

Table 79: GBP Tag Assignments

Endpoint	GBP Tag
Engineering Staff (ENG)	10
Contractor (CON)	20
Security Staff (SS)	30
Engineering Server (ES)	40
Security Cam (CAM)	50

[Table 80 on page 1321](#) shows the microsegmentation policies that we'll apply. We use **Y** to indicate where access is permitted, **N** to indicate where access is blocked, and **-** to indicate not applicable (since there is only one contractor, one engineering server, and one security camera).

Table 80: Microsegmentation Policy

	ENG (Tag 10)	CON (Tag 20)	SS (Tag 30)	ES (Tag 40)	CAM (Tag 50)
ENG (Tag 10)	Y	Y	N	Y	N
CON (Tag 20)	Y	-	N	N	N
SS (Tag 30)	N	N	Y	N	Y
ES (Tag 40)	Y	N	N	-	N
CAM (Tag 50)	N	N	Y	N	-

1. Follow this step if you're assigning GBP tags using the CLI.

Add the following assignments to both Leaf 1 and Leaf 2. Instead of tailoring the assignments to each switch individually, we'll use the same assignments on both switches for convenience and consistency.

```
set firewall family any filter assign-tags micro-segmentation
```

```

set firewall family any filter assign-tags term t1 from mac-address 54:52:00:00:01:10
set firewall family any filter assign-tags term t1 then gbp-tag 10
set firewall family any filter assign-tags term t2 from mac-address 54:52:00:00:02:10
set firewall family any filter assign-tags term t2 then gbp-tag 10
set firewall family any filter assign-tags term t3 from mac-address 54:52:00:00:01:20
set firewall family any filter assign-tags term t3 then gbp-tag 20
set firewall family any filter assign-tags term t4 from mac-address 54:52:00:00:01:30
set firewall family any filter assign-tags term t4 then gbp-tag 30
set firewall family any filter assign-tags term t5 from mac-address 54:52:00:00:02:30
set firewall family any filter assign-tags term t5 then gbp-tag 30
set firewall family any filter assign-tags term t6 from mac-address 54:52:00:00:02:40
set firewall family any filter assign-tags term t6 then gbp-tag 40
set firewall family any filter assign-tags term t7 from mac-address 54:52:00:00:02:50
set firewall family any filter assign-tags term t7 then gbp-tag 50

```

2. Follow this step if you're assigning GBP tags using a RADIUS server.

a. Configure both Leaf 1 and Leaf 2 to use the RADIUS server:

```

set groups dot1xgbp access radius-server 10.0.0.100 port 1812
set groups dot1xgbp access radius-server 10.0.0.100 secret "<secret-key>"
set groups dot1xgbp access profile my_radius_profile authentication-order radius
set groups dot1xgbp access profile my_radius_profile radius authentication-server
10.0.0.100
set groups dot1xgbp access profile my_radius_profile radius accounting-server 10.0.0.100
set groups dot1xgbp access profile my_radius_profile accounting order radius

```

where 10.0.0.100 is the IP address of the RADIUS server.

b. Configure the relevant interfaces on both Leaf 1 and Leaf 2 for RADIUS authentication:

```

set groups dot1xgbp protocols dot1x authenticator authentication-profile-name
radius_profile
set groups dot1xgbp protocols dot1x authenticator interface xe-0/0/46.0 supplicant multiple
set groups dot1xgbp protocols dot1x authenticator interface xe-0/0/46.0 mac-radius

```

where xe-0/0/46.0 is the interface connecting to the endpoint users and devices on both switches.

c. Configure the GBP tag assignments on the RADIUS server using the Juniper-Switching-Filter VSA or the Juniper-Group-Based-Policy-Id VSA.

Juniper-Switching-Filter VSA:

```
Engineer01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Switching-Filter = "apply action gbp-tag 10"
Engineer02 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Switching-Filter = "apply action gbp-tag 10"
Contractor01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Switching-Filter = "apply action gbp-tag 20"
SecurityStaff01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Switching-Filter = "apply action gbp-tag 30"
SecurityStaff02 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Switching-Filter = "apply action gbp-tag 30"
EngServer01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Switching-Filter = "apply action gbp-tag 40"
SecurityCam01EngServer01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Switching-Filter = "apply action gbp-tag 50"
```

Juniper-Group-Based-Policy-Id VSA:

```
Engineer01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Group-Based-Policy-Id = "10"
Engineer02 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Group-Based-Policy-Id = "10"
Contractor01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Group-Based-Policy-Id = "20"
SecurityStaff01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Group-Based-Policy-Id = "30"
SecurityStaff02 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Group-Based-Policy-Id = "30"
EngServer01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Group-Based-Policy-Id = "40"
SecurityCam01 Auth-Type = EAP, Cleartext-Password := "<password>"
Juniper-Group-Based-Policy-Id = "50"
```

3. Set the desired GBP profile on both Leaf 1 and Leaf 2.

```
set chassis forwarding-options vxlan-gbp-profile
```

4. Enable ingress policy enforcement on both Leaf 1 and Leaf 2.

```
set forwarding-options evpn-vxlan gbp ingress-enforcement
```

Set ingress enforcement on Leaf 1 to allow Leaf 1 to enforce policy for traffic in the Leaf 1 to Leaf 2 direction.

Set ingress enforcement on Leaf 2 to allow Leaf 2 to enforce policy for traffic in the Leaf 2 to Leaf 1 direction.

In our example, we're setting ingress enforcement on both switches.

5. Create firewall filter rules to enforce the microsegmentation policies shown in [Table 80 on page 1321](#).

Set the following rules on both Leaf 1 and Leaf 2.

We'll add a default discard rule at the end so that we don't have to configure any of the discard relationships explicitly. The default discard capability was introduced in Junos OS Release 24.2R1. See ["Explicit Default Discard" on page 1297](#).

If you're running a release earlier than Junos OS Release 24.2R1, then you must configure each discard rule explicitly.

```
set groups gbp-policy firewall family any filter gbp-policy term eng-to-eng from gbp-src-tag 10
set groups gbp-policy firewall family any filter gbp-policy term eng-to-eng from gbp-dst-tag 10
set groups gbp-policy firewall family any filter gbp-policy term eng-to-eng then accept
set groups gbp-policy firewall family any filter gbp-policy term eng-to-eng then count ENG-ENG

set groups gbp-policy firewall family any filter gbp-policy term eng-to-con from gbp-src-tag 10
set groups gbp-policy firewall family any filter gbp-policy term eng-to-con from gbp-dst-tag 20
set groups gbp-policy firewall family any filter gbp-policy term eng-to-con then accept
set groups gbp-policy firewall family any filter gbp-policy term eng-to-con then count ENG-CON

set groups gbp-policy firewall family any filter gbp-policy term eng-to-es from gbp-src-tag 10
set groups gbp-policy firewall family any filter gbp-policy term eng-to-es from gbp-dst-tag 40
set groups gbp-policy firewall family any filter gbp-policy term eng-to-es then accept
set groups gbp-policy firewall family any filter gbp-policy term eng-to-es then count ENG-ES

set groups gbp-policy firewall family any filter gbp-policy term con-to-eng from gbp-src-tag 20
set groups gbp-policy firewall family any filter gbp-policy term con-to-eng from gbp-dst-tag 10
set groups gbp-policy firewall family any filter gbp-policy term con-to-eng then accept
set groups gbp-policy firewall family any filter gbp-policy term con-to-eng then count ENG-CON
```

```

set groups gbp-policy firewall family any filter gbp-policy term ss-to-ss from gbp-src-tag 30
set groups gbp-policy firewall family any filter gbp-policy term ss-to-ss from gbp-dst-tag 30
set groups gbp-policy firewall family any filter gbp-policy term ss-to-ss then accept
set groups gbp-policy firewall family any filter gbp-policy term ss-to-ss then count SS-SS

set groups gbp-policy firewall family any filter gbp-policy term ss-to-cam from gbp-src-tag 30
set groups gbp-policy firewall family any filter gbp-policy term ss-to-cam from gbp-dst-tag 50
set groups gbp-policy firewall family any filter gbp-policy term ss-to-cam then accept
set groups gbp-policy firewall family any filter gbp-policy term ss-to-cam then count SS-CAM

set groups gbp-policy firewall family any filter gbp-policy term es-to-eng from gbp-src-tag 40
set groups gbp-policy firewall family any filter gbp-policy term es-to-eng from gbp-dst-tag 10
set groups gbp-policy firewall family any filter gbp-policy term es-to-eng then accept
set groups gbp-policy firewall family any filter gbp-policy term es-to-eng then count ENG-ES

set groups gbp-policy firewall family any filter gbp-policy term cam-to-ss from gbp-src-tag 50
set groups gbp-policy firewall family any filter gbp-policy term cam-to-ss from gbp-dst-tag 30
set groups gbp-policy firewall family any filter gbp-policy term cam-to-ss then accept
set groups gbp-policy firewall family any filter gbp-policy term cam-to-ss then count SS-CAM

set groups gbp-policy firewall family any filter gbp-policy term final then discard

set apply-groups gbp-policy

```

You've now configured the GBP tag assignments and the GBP policies to microsegment traffic in our example.

Configuring the Tunneling of Q-in-Q Traffic

IN THIS CHAPTER

- [Examples: Tunneling Q-in-Q Traffic in an EVPN-VXLAN Overlay Network | 1326](#)

Examples: Tunneling Q-in-Q Traffic in an EVPN-VXLAN Overlay Network

IN THIS SECTION

- [Requirements | 1328](#)
- [Overview and Topology | 1329](#)
- [Configuring Traffic Pattern 1: Popping an S-VLAN Tag | 1333](#)
- [Configuring Traffic Pattern 2: Mapping a Range of C-VLANs to an S-VLAN, and Pushing an S-VLAN Tag | 1338](#)
- [Configuring Traffic Pattern 3: Retaining S-VLAN and C-VLAN Tags | 1344](#)
- [Configuring Traffic Pattern 4: Popping and Later Pushing an S-VLAN Tag | 1347](#)

The tunneling of Q-in-Q packets in an Ethernet VPN-Virtual Extensible LAN (EVPN-VXLAN) overlay network is supported as follows:

- Starting with Junos OS Release 17.2R1, QFX5100 switches that function as Layer 2 VXLAN tunnel endpoints (VTEPs) can tunnel single- and double-tagged Q-in-Q packets in an EVPN-VXLAN bridged overlay or centrally-routed bridging (CRB) overlay (EVPN-VXLAN network with a two-layer IP fabric).
- Starting with Junos OS Release 18.2R1, QFX5110, QFX5200, and EX4600 switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets in a bridged overlay or CRB overlay.

- Starting with Junos OS Release 18.3R1, QFX10002 (except QFX10002-60C), QFX10008, and QFX10016 switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets using service-provider style interface configuration in a bridged overlay or CRB overlay.
- QFX5120 switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets in a bridged overlay or CRB overlay as follows:
 - QFX5120-48Y: Starting in Junos OS Release 18.4R2
 - QFX5120-32C: Starting in Junos OS Release 19.1R1
 - QFX5120-48T: Starting in Junos OS Release 20.2R1
 - QFX5120-48YM: Starting in Junos OS Release 20.2R1
- Starting with Junos OS Evolved Release 21.2R1, QFX5130-32CD switches operating as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets using the service-provider style interface configuration in an edge-routed bridging (ERB) overlay.
- Starting with Junos OS Evolved Release 21.3R1, PTX10001-36MR, PTX10004, PTX10008, and PTX10016 routers operating as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets using the service-provider style interface configuration.
- Starting with Junos OS Evolved Release 22.1R1, ACX7100 routers operating as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets using the service-provider style interface configuration.



NOTE: Please refer to [Feature Explorer](#) for a complete list of the products that support this feature.

In addition to tunneling Q-in-Q packets, the ingress and egress VTEPs can perform the following Q-in-Q actions:

- Delete, or pop, an outer service VLAN (S-VLAN) tag from an incoming packet.
- Add, or push, an outer S-VLAN tag onto an outgoing packet.
- Map a configured range of customer VLAN (C-VLAN) IDs to an S-VLAN.



NOTE: The QFX Series and EX4600 switches support the pop and push actions only with a specified VLAN. The switches do not support the pop and push actions with a configured range of VLANs.

The ingress and egress VTEPs support the tunneling of Q-in-Q packets and the Q-in-Q actions in the context of the traffic patterns described in this topic. Support on EX4600, QFX5100, QFX5110,

QFX5200, and QFX5120 switches is limited to these traffic patterns. Other platforms that support this feature can also handle other Q-in-Q traffic patterns.



NOTE: This topic describes and shows how to configure the VXLAN tunneling of Q-in-Q packets for each traffic pattern. One or more of the traffic patterns might apply to your environment. Perform only those configurations that apply to your environment.

The ingress and egress VTEPs can also map a single- or double-tagged packet to a specified VLAN or to any VLAN specified in a configured list, and further map the VLAN to a VXLAN network identifier (VNI).

To enable the tunneling of Q-in-Q packets, you must configure a flexible VLAN tagging interface that can transmit 802.1Q VLAN single- and double-tagged packets on ingress and egress VTEPs.

Also, Q-in-Q packets must retain the inner C-VLAN tag while tunneling between ingress and egress VTEPs. Therefore, on each VTEP:

- You need to include the `encapsulate-inner-vlan` configuration statement at the `[edit vlans vlan-name vxlan]` hierarchy level, which retains the inner tag during packet encapsulation.



NOTE: Starting in Junos OS release 23.2R2, you cannot configure the `encapsulate-inner-vlan` statement in an EVPN MAC-VRF routing instance if the `vlan` has an IRB interface associated with it.



NOTE: Preserving the original VLAN tag is implicit with VLAN bundle services. As a result, in an EVPN-VXLAN environment, you don't need to configure the `encapsulate-inner-vlan` option with EVPN MAC-VRF instances that use the `vlan-bundle` service type. Explicitly configure this option with other service types.

- On most platforms you also need to configure the `decapsulate-accept-inner-vlan` statement at the `[edit protocols l2-learning]` hierarchy level, which retains the inner tag during packet de-encapsulation.

You don't need to configure the `decapsulate-accept-inner-vlan` option with the following devices:

- QFX10002, QFX10008, or QFX10016 switches.
- ACX7100 routers.

These routers don't drop the tagged packets, and can process the packets whether you configure the `decapsulate-accept-inner-vlan` option or not.

Requirements

These examples use the following hardware and software components:

- Two QFX5100 switches. One switch functions as the ingress VTEP; and the other as the egress VTEP.
- Junos OS Release 17.2R1 or later.

Overview and Topology

IN THIS SECTION

- [Understanding Traffic Pattern 1: Popping an S-VLAN Tag | 1330](#)
- [Understanding Traffic Pattern 2: Mapping a Range of C-VLANs to an S-VLAN, and Pushing an S-VLAN Tag | 1330](#)
- [Understanding Traffic Pattern 3: Retaining S-VLAN and C-VLAN Tags | 1331](#)
- [Understanding Traffic Pattern 4: Popping and Later Pushing an S-VLAN Tag | 1332](#)

This section describes the traffic patterns in which the VXLAN tunneling of Q-in-Q traffic is supported in a EVPN-VXLAN overlay network.

This topic describes and shows how to configure the VXLAN tunneling of Q-in-Q packets for each traffic pattern. One or more of the traffic patterns might apply to your environment. Perform only those configurations that apply to your environment.

The example configurations for these use cases include service provider style interface configuration with encapsulation `extended-vlan-bridge` at the set interfaces `interface-name` hierarchy level.



NOTE: PTX10001-36MR, PTX10004, PTX10008, PTX10016, QFX10002-32Q, QFX10002-72Q, QFX10008, and QFX10016 devices support Q-in-Q tunneling using only service-provider style interface configurations. QFX10002-60C switches don't support service provider style interface configuration, so they don't support Q-in-Q tunneling.

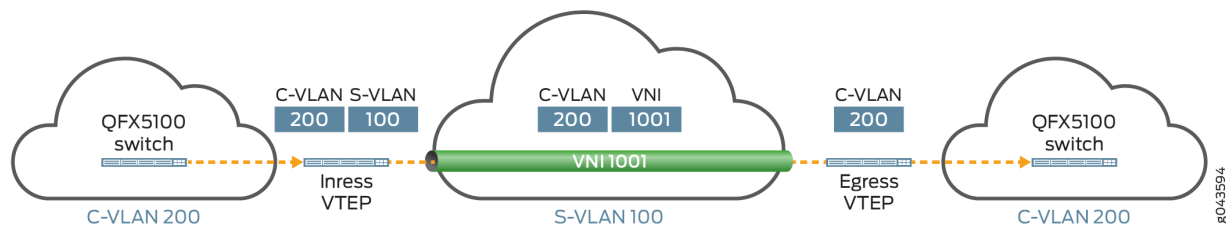


NOTE: On ACX7100 routers, you can alternatively use the flexible Ethernet services encapsulation type (encapsulation `flexible-ethernet-services`) in these examples if you need to enable the physical interface to support both service provider style and enterprise style interface configurations. See *Flexible Ethernet Services Encapsulation* for more on either of these encapsulation options.

Understanding Traffic Pattern 1: Popping an S-VLAN Tag

Figure 128 on page 1330 shows Q-in-Q traffic flowing from one dispersed C-VLAN 200 site to another by way of S-VLAN 100.

Figure 128: Popping an S-VLAN Tag



When a packet flows from C-VLAN 200 to S-VLAN 100 to C-VLAN 200, the ingress VTEP:

- Receives a packet with two tags—an inner C-VLAN tag of 200 and an outer S-VLAN tag of 100.
- Takes note of S-VLAN tag 100, which is mapped to VNI 1001, and then pops the tag.
- Encapsulates the packet with a VXLAN header that includes VNI 1001, and sends the packet with inner C-VLAN tag 200 and the VXLAN header.

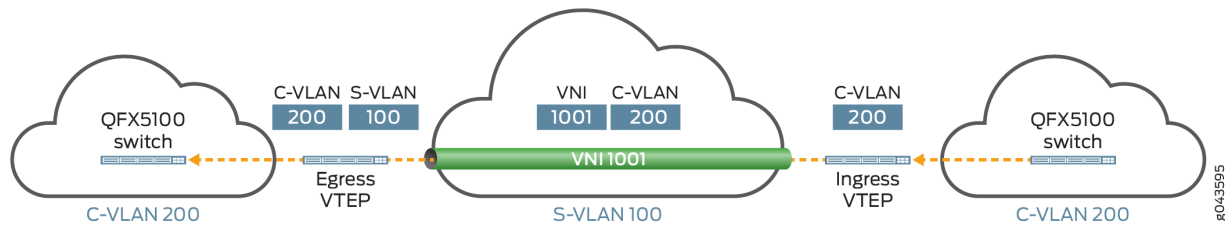
After the packet is tunneled over the Layer 3 underlay network, the egress VTEP:

- Removes the VXLAN header from the packet.
- Maps VNI 1001 back to S-VLAN 100.
- Sends the packet with C-VLAN tag 200.

Understanding Traffic Pattern 2: Mapping a Range of C-VLANs to an S-VLAN, and Pushing an S-VLAN Tag

Figure 129 on page 1331 shows Q-in-Q traffic flowing from one dispersed C-VLAN 200 site to another by way of S-VLAN 100.

Figure 129: Mapping a Range of C-VLANs to an S-VLAN



When a single-tagged packet flows from C-VLAN 200 to S-VLAN 100 to C-VLAN 200, the ingress VTEP:

- Receives a packet with a C-VLAN tag of 200.
- Takes note of C-VLAN tag 200, which is in a configured VLAN ID range 100 through 200 that is mapped to S-VLAN 100 and VNI 1001.
- Encapsulates the packet with a VXLAN header that includes VNI 1001 and sends the packet with C-VLAN tag 200 and VNI 1001.

After the packet is tunneled over the Layer 3 underlay network, the egress VTEP:

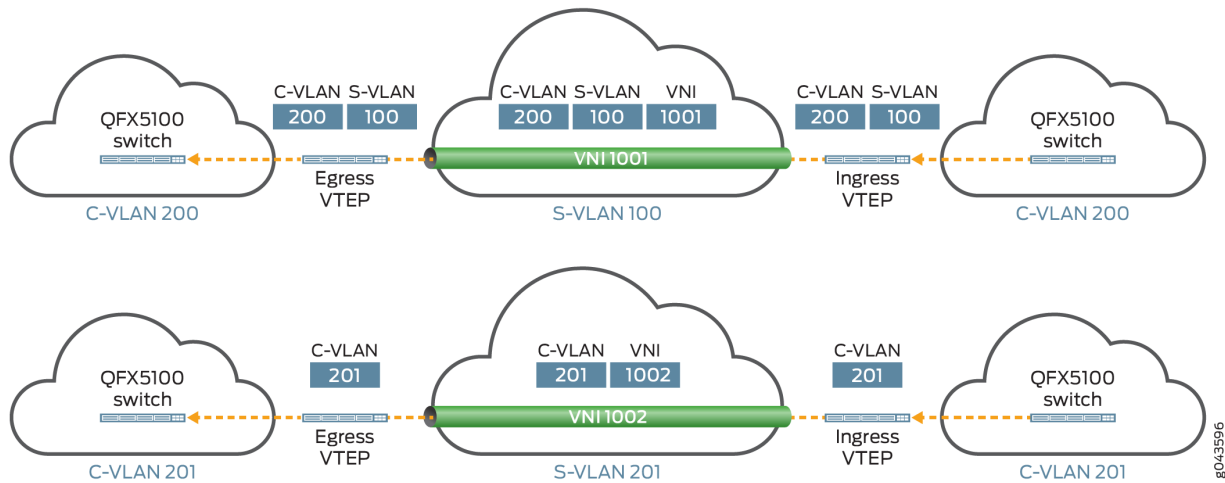
- De-encapsulates the packet.
- Maps the packet to S-VLAN 100 through its association with VNI 1001.
- Pushes S-VLAN tag 100 on the packet, and sends the packet with inner C-VLAN tag 200 and outer S-VLAN tag 100.

Understanding Traffic Pattern 3: Retaining S-VLAN and C-VLAN Tags

Figure 130 on page 1332 shows the following Q-in-Q traffic flows:

- Double-tagged packets from C-VLAN 200 to S-VLAN 100 to C-VLAN 200.
- Single-tagged packets from C-VLAN 201 to S-VLAN 201 to C-VLAN 201.

Figure 130: Retaining S-VLAN and C-VLAN Tags



When a packet flows from either C-VLAN 200 or C-VLAN 201, the ingress VTEP:

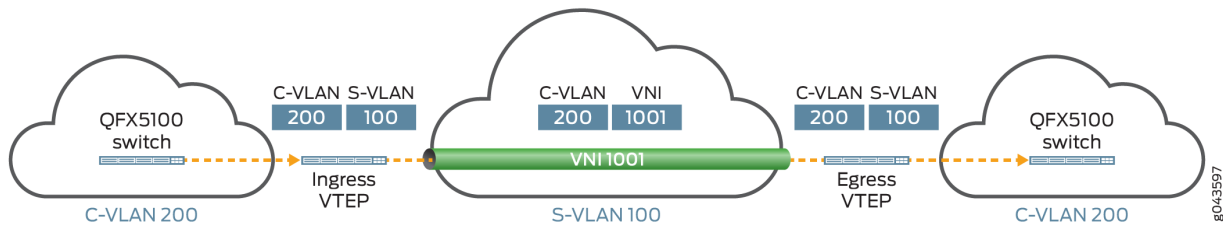
- Receives a packet—either a double-tagged packet with an inner C-VLAN tag of 200 and an outer S-VLAN tag of 100 or a single-tagged packet with a C-VLAN tag of 201.
- Takes note of outer S-VLAN tag 100, which is mapped to VNI 1001, for the double-tagged packet. For the single-tagged packet, the ingress VTEP takes note of C-VLAN tag 201, which is mapped to VNI 1002.
- Encapsulates the packet with a VXLAN header that includes VNI 1001 for the double-tagged packet and VNI 1002 for the single-tagged packet. In addition to the VXLAN header, the ingress VTEP sends the double-tagged packet with inner C-VLAN tag 200 and outer S-VLAN tag 100, and the single-tagged packet with C-VLAN tag 201.

After the packet is tunneled over the Layer 3 underlay network, the egress VTEP:

- Removes the VXLAN header from the packet.
- For the double-tagged packet, maps VNI 1001 back to S-VLAN 100, and for the single-tagged packet, maps VNI 1002 back to C-VLAN 201.
- Sends the double-tagged packet with inner C-VLAN tag 200 and outer S-VLAN tag 100 and the single-tagged packet with C-VLAN tag 201.

Understanding Traffic Pattern 4: Popping and Later Pushing an S-VLAN Tag

Figure 131 on page 1333 shows Q-in-Q traffic flowing from one dispersed C-VLAN 200 site to another by way of S-VLAN 100.

Figure 131: Popping and Later Pushing an S-VLAN Tag

When a packet flows from C-VLAN 200 to S-VLAN 100 to C-VLAN 200, the ingress VTEP:

- Receives a packet with two tags—an inner C-VLAN tag of 200 and an outer S-VLAN tag of 100.
- Takes note of S-VLAN tag 100, which is mapped to VNI 1001, then pops the tag.
- Encapsulates the packet with a VXLAN header that includes VNI 1001 and sends the packet with inner C-VLAN tag 200 and the VXLAN header.

After the packet is tunneled over the Layer 3 underlay network, the egress VTEP:

- De-encapsulates the packet.
- Maps the packet back to S-VLAN 100 through its association with VNI 1001.
- Pushes S-VLAN tag 100 on the packet, and sends the packet with inner C-VLAN tag 200 and outer S-VLAN tag 100.

Configuring Traffic Pattern 1: Popping an S-VLAN Tag

IN THIS SECTION

- [Requirements | 1334](#)
- [Introduction | 1334](#)
- [Ingress VTEP Configuration for Traffic Pattern 1 | 1334](#)
- [Egress VTEP Configuration for Traffic Pattern 1 | 1336](#)

Requirements

Introduction

For this traffic pattern, the ingress and egress VTEPs in an EVPN-VXLAN overlay network must handle double-tagged Q-in-Q traffic. The ingress VTEP retains the inner C-VLAN tag and removes, or pops, the outer S-VLAN tag. The egress VTEP also retains the inner C-VLAN tag but does not reinstate the outer S-VLAN tag.



NOTE: QFX Series and EX4600 switches support this traffic pattern on both aggregated Ethernet and non-aggregated Ethernet interfaces.



NOTE: This configuration focuses on traffic pattern 1 only. It does not provide the configuration for EVPN and all aspects of VXLAN. For a more comprehensive EVPN-VXLAN configuration for a centrally-routed bridging overlay, see ["Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric" on page 771](#).

Ingress VTEP Configuration for Traffic Pattern 1

IN THIS SECTION

- [CLI Quick Configuration | 1334](#)
- [Procedure | 1335](#)

CLI Quick Configuration

To quickly configure the ingress VTEP, 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.



NOTE: To configure QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers to retain the inner C-VLAN tag while tunneling Q-in-Q packets, you must include the encapsulate-inner-vlan configuration statement at the [edit vlans *vlan-name* vxlan] hierarchy level (for service types other than VLAN bundle). You do not need to include

the `decapsulate-accept-inner-vlan` configuration statement at the `[edit protocols l2-learning]` hierarchy level.

```
set protocols l2-learning decapsulate-accept-inner-vlan
set interfaces xe-0/0/0 flexible-vlan-tagging
set interfaces xe-0/0/0 encapsulation extended-vlan-bridge
set interfaces xe-0/0/0 unit 100 vlan-id 100
set interfaces xe-0/0/0 unit 100 input-vlan-map pop
set interfaces xe-0/0/0 unit 100 output-vlan-map push
set vlans vlan_1 interface xe-0/0/0.100
set vlans vlan_1 vxlan vni 1001
set vlans vlan_1 vxlan encapsulate-inner-vlan
```

Procedure

Step-by-Step Procedure

To configure the ingress VTEP for traffic pattern 1:

1. On all supported Juniper Networks switches except QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers, configure the VTEP to retain the inner C-VLAN tag while de-encapsulating a packet.

```
[edit protocols l2-learning]
user@switch# set decapsulate-accept-inner-vlan
```



NOTE: To support the tunneling of Q-in-Q packets, you must configure both ingress and egress VTEP to retain the inner C-VLAN tag while de-encapsulating a packet.

2. Configure the physical interface to support the simultaneous transmission of 802.1Q VLAN single-tagged and double-tagged packets on its logical interfaces and to accept packets carrying Tag Protocol Identifier (TPID) 0x8100.

```
[edit interfaces]
user@switch# set xe-0/0/0 flexible-vlan-tagging
user@switch# set xe-0/0/0 encapsulation extended-vlan-bridge
```


3. On physical interface xe-0/0/0, create logical interface 100, and associate it with S-VLAN 100. Also, assuming that the ingress VTEP receives a double-tagged packet as described in this traffic pattern, specify that the outer S-VLAN tag is popped on incoming packets.

```
[edit interfaces]
user@switch# set xe-0/0/0 unit 100 vlan-id 100
user@switch# set xe-0/0/0 unit 100 input-vlan-map pop
user@switch# set interfaces xe-0/0/0 unit 100 output-vlan-map push
```



NOTE: If you include the pop configuration statement at the [edit interfaces unit input-vlan-map] hierarchy level, you must also include the push or swap-push configuration statement at the [edit interfaces unit output-vlan-map] hierarchy level to prevent an error when committing the configuration.

4. Create a VLAN named vlan_1, and map it to logical interface xe-0/0/0.100 and VNI 1001. Also specify that the logical interface retains the inner C-VLAN tag while encapsulating a packet.



NOTE: Preserving the original VLAN tag is implicit with VLAN bundle services. As a result, in an EVPN-VXLAN environment, you don't need to configure this option with EVPN MAC-VRF instance vlan-bundle service type configurations. Explicitly configure this option with other service types.

```
[edit vlans]
user@switch# set vlan_1 interface xe-0/0/0.100
user@switch# set vlan_1 vxlan vni 1001
user@switch# set vlan_1 vxlan encapsulate-inner-vlan
```

Egress VTEP Configuration for Traffic Pattern 1

IN THIS SECTION

- [CLI Quick Configuration | 1337](#)
- [Procedure | 1337](#)

CLI Quick Configuration

To quickly configure the egress VTEP, 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.



NOTE: To configure QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers to retain the inner C-VLAN tag while tunneling Q-in-Q packets, you must include the encapsulate-inner-vlan configuration statement at the [edit vlans *vlan-name* vxlan] hierarchy level (for service types other than VLAN bundle). You do not need to include the decapsulate-accept-inner-vlan configuration statement at the [edit protocols l2-learning] hierarchy level.

```
set protocols l2-learning decapsulate-accept-inner-vlan
set interfaces xe-0/0/0 flexible-vlan-tagging
set interfaces xe-0/0/0 encapsulation extended-vlan-bridge
set interfaces xe-0/0/0 unit 100 vlan-id 100
set vlans vlan_1 interface xe-0/0/0.100
set vlans vlan_1 vxlan vni 1001
set vlans vlan_1 vxlan encapsulate-inner-vlan
```

Procedure

Step-by-Step Procedure

To configure the egress VTEP for traffic pattern 1:

1. On all Juniper Networks devices except QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers, configure the VTEP to retain the inner VLAN tag while de-encapsulating a packet.

```
[edit protocols l2-learning]
user@switch# set decapsulate-accept-inner-vlan
```

2. Configure the physical interface to support the simultaneous transmission of 802.1Q VLAN single-tagged and double-tagged packets on its logical interfaces and to accept packets carrying TPID 0x8100. Also, configure logical interface 100, and associate it with VLAN 100.

```
[edit interfaces]
user@switch# set xe-0/0/0 flexible-vlan-tagging
```

```

user@switch# set xe-0/0/0 encapsulation extended-vlan-bridge
user@switch# set xe-0/0/0 unit 100 vlan-id 100

```

3. Create a VLAN named `vlan_1`, and map it to logical interface `xe-0/0/0.100` and VNI 1001. Also specify that the logical interface retains the inner C-VLAN tag while encapsulating a packet.



NOTE: Preserving the original VLAN tag is implicit with VLAN bundle services. As a result, in an EVPN-VXLAN environment, you don't need to configure this option with EVPN MAC-VRF instance `vlan-bundle` service type configurations. Explicitly configure this option with other service types.

```

[edit vlans]
user@switch# set vlan_1 interface xe-0/0/0.100
user@switch# set vlan_1 vxlan vni 1001
user@switch# set vlan_1 vxlan encapsulate-inner-vlan

```



NOTE: To support the tunneling of Q-in-Q packets, you must configure both ingress and egress VTEPs to retain the inner C-VLAN tag while encapsulating a packet.

Configuring Traffic Pattern 2: Mapping a Range of C-VLANs to an S-VLAN, and Pushing an S-VLAN Tag

IN THIS SECTION

- [Requirements | 1339](#)
- [Introduction | 1339](#)
- [Ingress VTEP Configuration for Traffic Pattern 2 | 1339](#)
- [Egress VTEP Configuration for Traffic Pattern 2 | 1341](#)

Requirements

Introduction

For this traffic pattern, the ingress VTEP in an EVPN-VXLAN overlay network receives a packet tagged with a C-VLAN ID, one of which is included in a configured range of C-VLAN IDs that are mapped to a particular S-VLAN. After the packet is tunneled over the Layer 3 network, the egress VTEP retains the C-VLAN tag and pushes an outer tag for that particular S-VLAN on the packet.

We support this traffic pattern on both aggregated Ethernet interfaces and non-aggregated Ethernet interfaces.



NOTE: QFX Series and EX4600 switches do not support the pop and push actions with a configured range of VLANs.

Ingress VTEP Configuration for Traffic Pattern 2

IN THIS SECTION

- [CLI Quick Configuration | 1339](#)
- [Procedure | 1340](#)



NOTE: This configuration focuses on traffic pattern 2 only. It does not provide the configuration for EVPN and all aspects of VXLAN. For a more comprehensive EVPN-VXLAN configuration for a centrally-routed bridging overlay, see "[Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric](#)" on page 771.

CLI Quick Configuration

To quickly configure the ingress VTEP, 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.



NOTE: To configure QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers to retain the inner C-VLAN tag while tunneling Q-in-Q packets, you must include

the encapsulate-inner-vlan configuration statement at the [edit vlans *vlan-name* vxlan] hierarchy level (for service types other than VLAN bundle). You do not need to include the decapsulate-accept-inner-vlan configuration statement at the [edit protocols l2-learning] hierarchy level.

```
set protocols l2-learning decapsulate-accept-inner-vlan
set interfaces xe-0/0/5 flexible-vlan-tagging
set interfaces xe-0/0/5 encapsulation extended-vlan-bridge
set interfaces xe-0/0/5 unit 100 vlan-id-list 100-200
set vlans vlan_range1 interface xe-0/0/5.100
set vlans vlan_range1 vxlan vni 1001
set vlans vlan_range1 vxlan encapsulate-inner-vlan
```

Procedure

Step-by-Step Procedure

To configure the ingress VTEP for traffic pattern 2:

1. On all supported Juniper Networks switches except QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers, configure the VTEP to retain the inner VLAN tag while de-encapsulating a packet.

```
[edit protocols l2-learning]
user@switch# set decapsulate-accept-inner-vlan
```



NOTE: To support the tunneling of Q-in-Q packets, you must configure both ingress and egress VTEPs to retain the inner C-VLAN tag while de-encapsulating a packet.

2. Configure the physical interface to support the simultaneous transmission of 802.1Q VLAN single-tagged and double-tagged packets on its logical interfaces and to accept packets carrying TPID 0x8100. Also, for the physical interface, configure logical interface 100 and map it to C-VLANs 100 through 200.

```
[edit interfaces]
user@switch# set xe-0/0/5 flexible-vlan-tagging
```

```

user@switch# set xe-0/0/5 encapsulation extended-vlan-bridge
user@switch# set xe-0/0/5 unit 100 vlan-id-list 100-200

```

3. Create a VLAN named `vlan_range1`, and map it to logical interface 100 and VNI 1001. Also specify that the logical interface retains the inner VLAN tag while encapsulating a packet.



NOTE: Preserving the original VLAN tag is implicit with VLAN bundle services. As a result, in an EVPN-VXLAN environment, you don't need to configure this option with EVPN MAC-VRF instance `vlan-bundle` service type configurations. Explicitly configure this option with other service types.

```

[edit vlans]
user@switch# set vlan_range1 interface xe-0/0/5.100
user@switch# set vlan_range1 vxlan vni 1001
user@switch# set vlan_range1 vxlan encapsulate-inner-vlan

```

Egress VTEP Configuration for Traffic Pattern 2

IN THIS SECTION

- [CLI Quick Configuration | 1341](#)
- [Procedure | 1342](#)

CLI Quick Configuration

To quickly configure the egress VTEP, 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.



NOTE: To configure QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers to retain the inner C-VLAN tag while tunneling Q-in-Q packets, you must include only the `encapsulate-inner-vlan` configuration statement at the `[edit vlans vlan-name vxlan]` hierarchy level (for service types other than VLAN bundle). You do not need to include

the `decapsulate-accept-inner-vlan` configuration statement at the `[edit protocols l2-learning]` hierarchy level.

```
set protocols l2-learning decapsulate-accept-inner-vlan
set interfaces xe-0/0/0 flexible-vlan-tagging
set interfaces xe-0/0/0 encapsulation extended-vlan-bridge
set interfaces xe-0/0/0 unit 100 vlan-id 100
set interfaces xe-0/0/0 unit 100 input-vlan-map pop
set interfaces xe-0/0/0 unit 100 output-vlan-map push
set vlans v100 interface xe-0/0/0.100
set vlans v100 vxlan vni 1001
set vlans v100 vxlan encapsulate-inner-vlan
```

Procedure

Step-by-Step Procedure

To configure the egress VTEP for traffic pattern 2:

1. On all supported Juniper Networks switches except QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers, configure the VTEP to retain the inner VLAN tag while de-encapsulating a packet.

```
[edit protocols l2-learning]
user@switch# set decapsulate-accept-inner-vlan
```

2. Configure the physical interface to support the simultaneous transmission of 802.1Q VLAN single-tagged and double-tagged packets on its logical interfaces and to accept packets carrying TPID 0x8100.

```
[edit interfaces]
user@switch# set xe-0/0/0 flexible-vlan-tagging
user@switch# set xe-0/0/0 encapsulation extended-vlan-bridge
```

3. Create logical interface 100, and associate it with S-VLAN 100. Also, specify that when logical interface 100 receives a packet without an outer S-VLAN tag, the interface pushes outer S-VLAN tag 100 on the outgoing packet.

```
[edit interfaces]
user@switch# set xe-0/0/0 unit 100 vlan-id 100
user@switch# set xe-0/0/0 unit 100 input-vlan-map pop
user@switch# set xe-0/0/0 unit 100 output-vlan-map push
```



NOTE: If you include the push configuration statement at the [edit interfaces unit output-vlan-map] hierarchy level, you must also include the pop configuration statement at the [edit interfaces unit input-vlan-map] hierarchy level to prevent an error when committing the configuration.

4. Create a VLAN named v100, and map it to logical interface 100 and VNI 1001. Also specify that the logical interface retains the inner VLAN tag while encapsulating a packet.



NOTE: Preserving the original VLAN tag is implicit with VLAN bundle services. As a result, in an EVPN-VXLAN environment, you don't need to configure this option with EVPN MAC-VRF instance `vlan-bundle` service type configurations. Explicitly configure this option with other service types.

```
[edit vlans]
user@switch# set v100 interface xe-0/0/0.100
user@switch# set v100 vxlan vni 1001
user@switch# set v100 vxlan encapsulate-inner-vlan
```



NOTE: To support the tunneling of Q-in-Q packets, you must configure both ingress and egress VTEPs to retain the inner C-VLAN tag while encapsulating a packet.

Configuring Traffic Pattern 3: Retaining S-VLAN and C-VLAN Tags

IN THIS SECTION

- [Requirements | 1344](#)
- [Introduction | 1344](#)
- [Ingress and Egress VTEP Configuration for Traffic Pattern 3 | 1344](#)

Requirements

Introduction

For this traffic pattern, the ingress and egress VTEPs in an EVPN-VXLAN overlay network must handle Q-in-Q data packets that are single- or double-tagged. For both single- and double-tagged packets, the ingress and egress VTEPs encapsulate and de-encapsulate the packets without making any changes to the tag(s).



NOTE: QFX Series and EX4600 switches support this traffic pattern on both aggregated Ethernet and non-aggregated Ethernet interfaces.

Ingress and Egress VTEP Configuration for Traffic Pattern 3

IN THIS SECTION

- [CLI Quick Configuration | 1345](#)
- [Procedure | 1345](#)



NOTE: This configuration focuses on traffic pattern 3 only. It does not provide the configuration for EVPN and all aspects of VXLAN. For a more comprehensive EVPN-VXLAN configuration for a centrally-routed bridging overlay, see "[Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric](#)" on page 771.

CLI Quick Configuration

To quickly configure the ingress and egress VTEPs, 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.



NOTE: To configure QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers to retain the inner C-VLAN tag while tunneling Q-in-Q packets, you must include only the encapsulate-inner-vlan configuration statement at the [edit vlans *vlan-name* vxlan] hierarchy level (for service types other than VLAN bundle). You do not need to include the decapsulate-accept-inner-vlan configuration statement at the [edit protocols l2-learning] hierarchy level.

```
set protocols l2-learning decapsulate-accept-inner-vlan
set interfaces xe-0/0/15 flexible-vlan-tagging
set interfaces xe-0/0/15 encapsulation extended-vlan-bridge
set interfaces xe-0/0/15 unit 100 vlan-id 100
set interfaces xe-0/0/15 unit 201 vlan-id 201
set vlans vlan_100 interface xe-0/0/15.100
set vlans vlan_100 vxlan vni 1001
set vlans vlan_100 vxlan encapsulate-inner-vlan
set vlans vlan_201 interface xe-0/0/15.201
set vlans vlan_201 vxlan vni 1002
set vlans vlan_201 vxlan encapsulate-inner-vlan
```

Procedure

Step-by-Step Procedure

To configure the ingress and egress VTEP for traffic pattern 3:

1. On all supported Juniper Networks switches except QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers, configure the VTEPs to retain the inner VLAN tag while de-encapsulating a packet.

```
[edit protocols l2-learning]
user@switch# set decapsulate-accept-inner-vlan
```



NOTE: To support the tunneling of Q-in-Q packets, you must configure both ingress and egress VTEPs to retain the inner C-VLAN tag while de-encapsulating a packet.

2. Configure the physical interface to support the simultaneous transmission of 802.1Q VLAN single- and double-tagged packets on its logical interfaces and to accept packets carrying TPID 0x8100. Also, on the physical interface, create logical interfaces 100 and 201, and associate them with S-VLAN 100 and C-VLAN 201, respectively.

```
[edit interfaces]
user@switch# set xe-0/0/15 flexible-vlan-tagging
user@switch# set xe-0/0/15 encapsulation extended-vlan-bridge
user@switch# set xe-0/0/15 unit 100 vlan-id 100
user@switch# set xe-0/0/15 unit 201 vlan-id 201
```

3. Create a VLAN named `vlan_100`, and map it to logical interface 100 and VNI 1001. Also create a VLAN named `vlan_201`, and map it to logical interface 201 and VNI 1002. Also specify that the logical interfaces retain the inner VLAN tag while encapsulating a packet.



NOTE: Preserving the original VLAN tag is implicit with VLAN bundle services. As a result, in an EVPN-VXLAN environment, you don't need to configure this option with EVPN MAC-VRF instance `vlan-bundle` service type configurations. Explicitly configure this option with other service types.

```
[edit vlans]
user@switch# set vlan_100 interface xe-0/0/15.100
user@switch# set vlan_100 vxlan vni 1001
user@switch# set vlan_100 vxlan encapsulate-inner-vlan
user@switch# set vlan_201 interface xe-0/0/15.201
user@switch# set vlan_201 vxlan vni 1002
user@switch# set vlan_201 vxlan encapsulate-inner-vlan
```



NOTE: To support the tunneling of Q-in-Q packets, you must configure both ingress and egress VTEPs to retain the inner C-VLAN tag while encapsulating a packet.

Configuring Traffic Pattern 4: Popping and Later Pushing an S-VLAN Tag

IN THIS SECTION

- Requirements | 1347
- Introduction | 1347
- Configuration for Ingress VTEP for Traffic Pattern 4 | 1347
- Configuration for Egress VTEP for Traffic Pattern 4 | 1350

Requirements

Introduction

For this traffic pattern, the ingress and egress VTEPs in an EVPN-VXLAN overlay network must handle double-tagged Q-in-Q traffic. The ingress VTEP retains the inner C-VLAN tag and removes, or pops, the outer S-VLAN tag. After the packets are tunneled over the Layer 3 network, the egress VTEP pushes the S-VLAN tag back on the packet.



NOTE: QFX Series and EX4600 switches support this traffic patterns on both aggregated Ethernet and non-aggregated Ethernet interfaces.



NOTE: This configuration focuses on traffic pattern 4 only. It does not provide the configuration for EVPN and all aspects of VXLAN. For a more comprehensive EVPN-VXLAN configuration for a centrally-routed bridging overlay, see "[Example: Configure an EVPN-VXLAN Centrally-Routed Bridging Fabric](#)" on page 771.

Configuration for Ingress VTEP for Traffic Pattern 4

IN THIS SECTION

- CLI Quick Configuration | 1348
- Procedure | 1348

CLI Quick Configuration

To quickly configure the ingress VTEP, 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.



NOTE: To configure QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers to retain the inner C-VLAN tag while tunneling Q-in-Q packets, you must include only the encapsulate-inner-vlan configuration statement at the [edit vlans *vlan-name* vxlan] hierarchy level (for service types other than VLAN bundle). You do not need to include the decapsulate-accept-inner-vlan configuration statement at the [edit protocols l2-learning] hierarchy level.

```
set protocols l2-learning decapsulate-accept-inner-vlan
set interfaces xe-0/0/0 flexible-vlan-tagging
set interfaces xe-0/0/0 encapsulation extended-vlan-bridge
set interfaces xe-0/0/0 unit 100 vlan-id 100
set interfaces xe-0/0/0 unit 100 input-vlan-map pop
set interfaces xe-0/0/0 unit 100 output-vlan-map push
set vlans vlan_1 interface xe-0/0/0.100
set vlans vlan_1 vxlan vni 1001
set vlans vlan_1 vxlan encapsulate-inner-vlan
```

Procedure

Step-by-Step Procedure

To configure the ingress VTEP for traffic pattern 4:

1. On all supported Juniper Networks switches except QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers, configure the VTEP to retain the inner C-VLAN tag while de-encapsulating a packet.

```
[edit protocols l2-learning]
user@switch# set decapsulate-accept-inner-vlan
```



NOTE: To support the VXLAN tunneling of Q-in-Q packets, you must configure both ingress and egress VTEP to retain the inner C-VLAN tag while de-encapsulating a packet.

2. Configure the physical interface to support the simultaneous transmission of 802.1Q VLAN single-tagged and double-tagged packets on its logical interfaces and to accept packets carrying TPID 0x8100.

```
[edit interfaces]
user@switch# set xe-0/0/0 flexible-vlan-tagging
user@switch# set xe-0/0/0 encapsulation extended-vlan-bridge
```

3. On physical interface xe-0/0/0, create logical interface 100, and associate it with S-VLAN 100. Also, assuming that the ingress VTEP receives a double-tagged packet as described in this traffic pattern, specify that the outer S-VLAN tag is popped on incoming packets. To accommodate a scenario in which the traffic flow is reversed, and the VTEP functions as an egress VTEP that receives a single-tagged packet from C-VLAN 200, you can optionally specify that an outer S-VLAN tag is added, or pushed, on outgoing packets.

```
[edit interfaces]
user@switch# set xe-0/0/0 unit 100 vlan-id 100
user@switch# set xe-0/0/0 unit 100 input-vlan-map pop
user@switch# set xe-0/0/0 unit 100 output-vlan-map push
```

4. Create a VLAN named vlan_1, and map it to logical interface 100 and VNI 1001. Also specify that the logical interface retains the inner VLAN tag while encapsulating a packet.



NOTE: Preserving the original VLAN tag is implicit with VLAN bundle services. As a result, in an EVPN-VXLAN environment, you don't need to configure this option with EVPN MAC-VRF instance vlan-bundle service type configurations. Explicitly configure this option with other service types.

```
[edit vlans]
user@switch# set vlan_1 interface xe-0/0/0.100
user@switch# set vlan_1 vxlan vni 1001
user@switch# set vlan_1 vxlan encapsulate-inner-vlan
```

Configuration for Egress VTEP for Traffic Pattern 4

IN THIS SECTION

- [CLI Quick Configuration | 1350](#)
- [Procedure | 1350](#)

CLI Quick Configuration

To quickly configure the egress VTEP, 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.



NOTE: To configure QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers to retain the inner C-VLAN tag while tunneling Q-in-Q packets, you must include only the encapsulate-inner-vlan configuration statement at the [edit vlans *vlan-name* vxlan] hierarchy level (for service types other than VLAN bundle). You do not need to include the decapsulate-accept-inner-vlan configuration statement at the [edit protocols l2-learning] hierarchy level.

```
set protocols l2-learning decapsulate-accept-inner-vlan
set interfaces xe-0/0/5 flexible-vlan-tagging
set interfaces xe-0/0/5 encapsulation extended-vlan-bridge
set interfaces xe-0/0/5 unit 100 vlan-id 100
set interfaces xe-0/0/5 unit 100 input-vlan-map pop
set interfaces xe-0/0/5 unit 100 output-vlan-map push
set vlans vlan_1 interface xe-0/0/5.100
set vlans vlan_1 vxlan vni 1001
set vlans vlan_1 vxlan encapsulate-inner-vlan
```

Procedure

Step-by-Step Procedure

To configure the egress VTEP for traffic pattern 4:

1. On all supported Juniper Networks switches except QFX10002, QFX10008, and QFX10016 switches or ACX7100 routers, configure the VTEP to retain the inner VLAN tag while de-encapsulating a packet.

```
[edit protocols l2-learning]
user@switch# set decapsulate-accept-inner-vlan
```

2. Configure the physical interface to support the simultaneous transmission of 802.1Q VLAN single-tagged and double-tagged packets on its logical interfaces and to accept packets carrying TPID 0x8100.

```
[edit interfaces]
user@switch# set xe-0/0/5 flexible-vlan-tagging
user@switch# set xe-0/0/5 encapsulation extended-vlan-bridge
```

3. Create logical interface 100, and associate it with S-VLAN 100. Also, specify that when logical interface 100 receives a packet without an outer S-VLAN tag, the interface pushes outer S-VLAN tag 100 on the outgoing packet.

```
[edit interfaces]
user@switch# set xe-0/0/5 unit 100 vlan-id-list 100
user@switch# set xe-0/0/5 unit 100 input-vlan-map pop
user@switch# set xe-0/0/5 unit 100 output-vlan-map push
```



NOTE: If you include the push configuration statement at the [edit interfaces unit output-vlan-map] hierarchy level, you must also include the pop configuration statement at the [edit interfaces unit input-vlan-map] hierarchy level to prevent an error when committing the configuration.

4. Create a VLAN named vlan_1, and map it to logical interface 100 and VNI 1001. Also specify that the logical interface retains the inner VLAN tag while encapsulating a packet.



NOTE: Preserving the original VLAN tag is implicit with VLAN bundle services. As a result, in an EVPN-VXLAN environment, you don't need to configure this option with

EVPN MAC-VRF instance `vlan-bundle` service type configurations. Explicitly configure this option with other service types.

```
[edit vlans]
user@switch# set vlan_1 interface xe-0/0/5.100
user@switch# set vlan_1 vxlan vni 1001
user@switch# set vlan_1 vxlan encapsulate-inner-vlan
```



NOTE: To support the tunneling of Q-in-Q packets, you must configure both ingress and egress VTEPs to retain the inner C-VLAN tag while encapsulating a packet.

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
22.1R1EVO	Starting with Junos OS Evolved Release 22.1R1, ACX7100 routers operating as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets in an ERB overlay.
21.3R1EVO	Starting with Junos OS Evolved Release 21.3R1, PTX10001-36MR, PTX10004, PTX10008, and PTX10016 routers operating as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets using the service-provider style interface configuration.
21.2R1EVO	Starting with Junos OS Evolved Release 21.2R1, QFX5130-32CD switches operating as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets using service-provider style interface configuration in an ERB overlay.
20.2R1	Starting with Junos OS Release 20.2R1, QFX5120-48T switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets in a bridged overlay or CRB overlay.
20.2R1	Starting with Junos OS Release 20.2R1, QFX5120-48YM switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets in a bridged overlay or CRB overlay.
19.1R1	Starting with Junos OS Release 19.1R1, QFX5120-32C switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets in a bridged overlay or CRB overlay.
18.4R2	Starting with Junos OS Release 18.4R2, QFX5120-48Y switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets in a bridged overlay or CRB overlay.

18.3R1	Starting with Junos OS Release 18.3R1, QFX10002 (except QFX10002-60C), QFX10008, and QFX10016 switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets using service-provider style interface configuration in a bridged overlay or CRB overlay.
18.2R1	Starting with Junos OS Release 18.2R1, QFX5110, QFX5200, and EX4600 switches that function as Layer 2 VTEPs can tunnel single- and double-tagged Q-in-Q packets in a bridged overlay or CRB overlay.
17.2R1	Starting with Junos OS Release 17.2R1, QFX5100 switches that function as Layer 2 VXLAN tunnel endpoints (VTEPs) can tunnel single- and double-tagged Q-in-Q packets in an EVPN-VXLAN bridged overlay or CRB overlay.

Tunnel Traffic Inspection on SRX Series Devices

IN THIS CHAPTER

- [Tunnel Inspection for EVPN-VXLAN by SRX Series Devices | 1354](#)

Tunnel Inspection for EVPN-VXLAN by SRX Series Devices

SUMMARY

Read this topic to understand how to setup your security device to perform tunnel inspection for EVPN-VXLAN to provide embedded security.

IN THIS SECTION

- [Overview | 1354](#)
- [Example - Configure Security Policies for EVPN-VXLAN Tunnel Inspection | 1358](#)
- [Configuration for Zone-Level Inspection, IDP, Content Security and Advanced Anti-Malware for Tunnel Inspection | 1371](#)
- [Complete Device Configurations | 1381](#)

Overview

IN THIS SECTION

- [Benefits | 1357](#)

(Ethernet VPN) EVPN-(Virtual Extensible LAN) VXLAN provides enterprises a common framework used to manage their campus and data center networks.

The rapidly increasing usage of mobile and IoT devices adds a large number of endpoints to a network. Modern enterprise networks must scale rapidly to provide immediate access to devices and to extend security and control for these endpoints.

To provide endpoint flexibility, EVPN-VXLAN decouples the underlay network (physical topology) from the overlay network (virtual topology). By using overlays, you gain the flexibility of providing Layer 2/ Layer 3 connectivity between endpoints across campus and data centers, while maintaining a consistent underlay architecture.

You can use SRX Series Firewalls in your EVPN-VXLAN solution to connect end-points in your campus, data center, branch and public cloud environments while providing embedded security.

Starting in Junos OS Release 21.1R1, the SRX Series Firewall can also apply following Layer 4/Layer 7 security services to the EVPN-VXLAN tunnel traffic:

These security services provide comprehensive protection for EVPN-VXLAN traffic:

- Application Identification—Identifies applications traversing the network regardless of port, protocol, or encryption
- IDP—Provides intrusion detection and prevention capabilities to protect against known and unknown threats
- Juniper ATP Cloud—Delivers cloud-based advanced threat detection and protection against malware and other sophisticated threats
- Content Security—Offers web filtering, antivirus, and anti-spam capabilities to protect against content-based threats



NOTE: key steps required to configure tunnel inspection for EVPN-VXLAN traffic and apply relevant security services:

- Define security zones and assigns interfaces.
- Create address-book entries for VXLAN Tunnel Endpoints (VTEPs) and VLAN subnets.
- Configure security policies to permit VXLAN traffic inspection using a tunnel-inspection profile.
- Define a tunnel-inspection profile (TP-1) for VXLAN with specific VNI and policy sets.
- Associate the VXLAN Network Identifier (VNI) with the inspection profile.
- Refer the security services in a tunnel inspection policy by specifying them in a security policy permit action, when the traffic matches the policy rule.

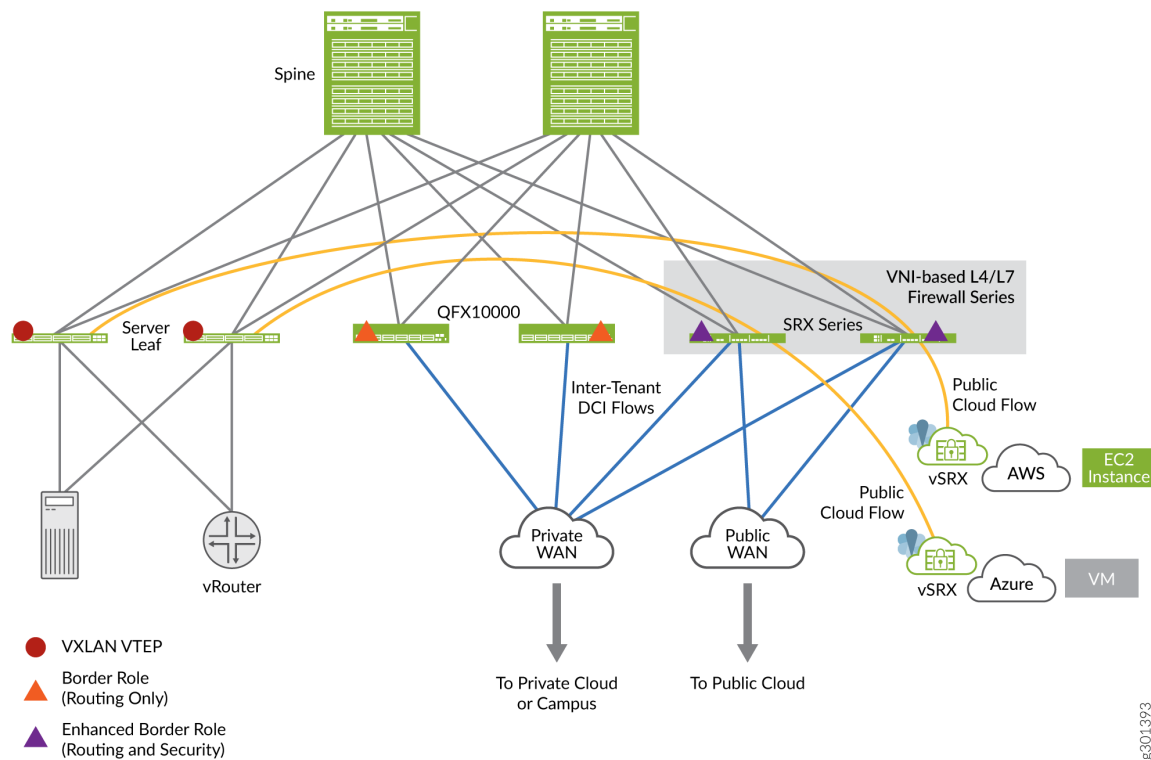
```
[edit]
user@host# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services idp-policy IDP-POLICY-NAME
user@host# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services ssl-proxy profile-name SSL-PROFILE-NAME
user@host# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services utm-policy UTM-POLICY-NAME
user@host# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services security-intelligence-policy SECINTEL-POLICY-NAME
user@host# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services advanced-anti-malware-policy AAMW-POLICY-NAME
```

For detailed information on configuring these security services, refer to the following documentation:

- [Application Security User Guide](#)
- [IDP User Guide](#)
- [Juniper ATP Cloud Administrator Guide](#)
- [Content Security User Guide](#)

[Figure 132 on page 1357](#) shows a typical deployment scenario of EVPN-VXLAN fabric based on Edge-routed bridging (ERB) with SRX Series Firewalls functioning in an enhanced border leaf (EBL) role. EBL enhances the traditional role of a border leaf with the ability to perform inspection of traffic in VXLAN tunnels.

Figure 132: EVPN-VXLAN Architecture with SRX Series Device



In the figure VXLAN traffic originating at the leaf 1 device traverses through the SRX Series Firewalls that function as EBLs. In this use case, the SRX Series Firewall is placed at the border, that is, at the entry and exit point of the campus or data center, to provide stateful inspection to the VXLAN encapsulated packets traversing through it.

In the architecture diagram, you can notice that an SRX Series Firewall is placed between two VTEP devices (devices that perform VXLAN encapsulation and decapsulation for the network traffic). The SRX Series Firewall performs stateful inspection when you enable the tunnel inspection feature with an appropriate security policy.

Benefits

Adding SRX Series Firewall in EVPN VXLAN provides:

- Added security with the capabilities of an enterprise grade firewall in the EVPN-VXLAN overlay.
- Enhanced tunnel inspection for VXLAN encapsulated traffic with Layer 4/Layer 7 security services.

Example - Configure Security Policies for EVPN-VXLAN Tunnel Inspection

IN THIS SECTION

- [Requirements | 1358](#)
- [Before you Begin | 1358](#)
- [Overview | 1359](#)
- [Configuration | 1361](#)
- [CLI Quick Configuration | 1361](#)
- [Step-by-Step Procedure | 1363](#)
- [Results | 1365](#)
- [Verification | 1368](#)

Use this example to configure the security policies that enable inspection of EVPN EVPN-VXLAN tunnel traffic on your SRX Series Firewalls.

Requirements

This example uses the following hardware and software components:

- An SRX Series Firewall or vSRX Virtual Firewall
- Junos OS Release 20.4R1

This example assumes that you already have an EVPN-VXLAN based network and want to enable tunnel inspection on SRX Series Firewall.

Before you Begin

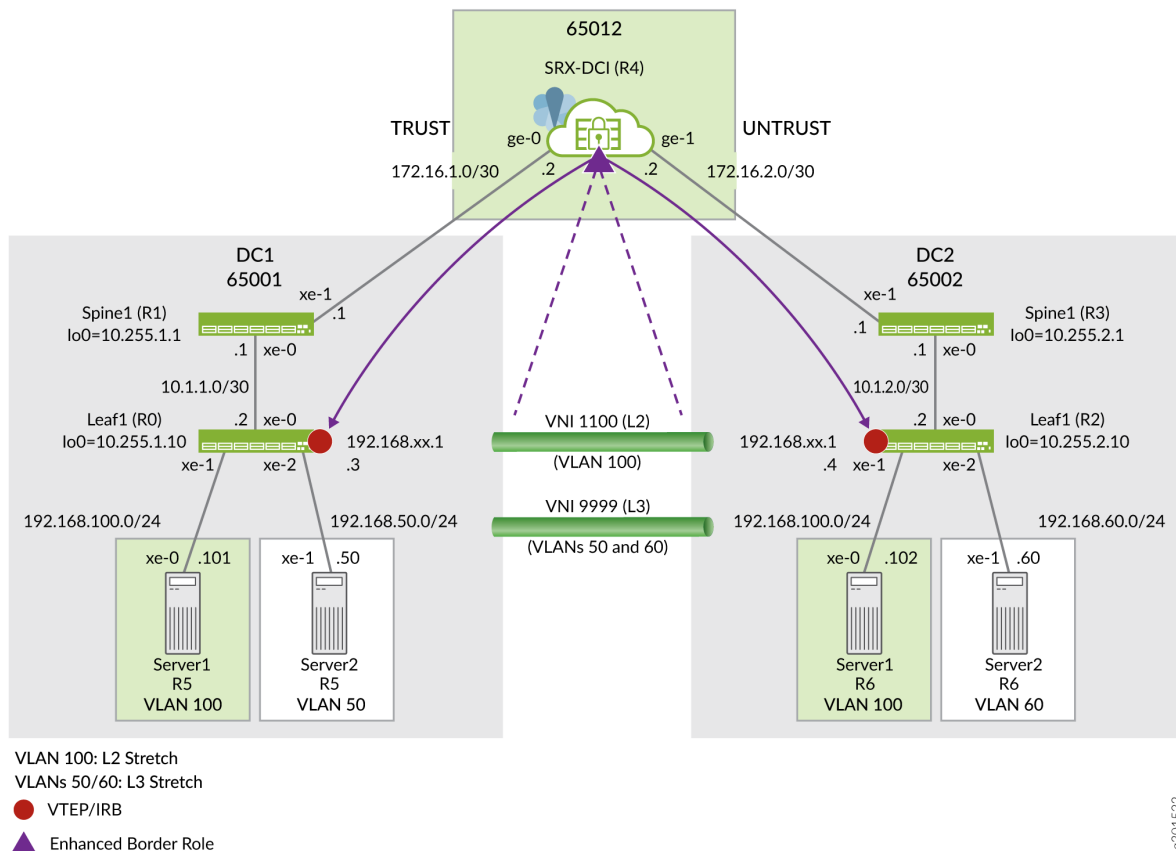
- Ensure you have a valid application identification feature license on your SRX Series Firewall and installed application signature pack on the device.
- Make sure you understand how EVPN and VXLAN works. See [EVPN-VXLAN Campus Architectures](#) to detail understanding EVPN-VXLAN
- This example assumes that you already have an EVPN-VXLAN based network fabric and want to enable tunnel inspection on the SRX Series Firewall. You can see the sample configuration of leaf and spine devices used in this example at "[Complete Device Configurations](#)" on page 1381.

Overview

In this example, we are focusing on configuring the SRX Series Firewall which is a part of a working EVPN-VXLAN network that consist of two DC locations each with an IP fabric. The SRX Series Firewall is placed in a Data Center Interconnect (DCI) role between the two DCs. In this configuration, the SRX Series Firewall performs stateful inspection of VXLAN encapsulated traffic flowing between the DCs when you enable tunnel inspection.

We are using the topology shown in [Figure 133 on page 1359](#) in this example.

Figure 133: Topology for VXLAN Tunnel Inspection



As given in the topology, the SRX Series Firewall is inspecting transit VLAN encapsulated traffic from the VXLAN tunnel endpoint (VTEP) on the leaves in both the DC-1 and DC-2 data centers. Any Juniper Networks device, both physical and virtual, that functions as a Layer 2 or Layer 3 VXLAN gateway can act as VTEP device to perform encapsulation and de-encapsulation.

Upon receipt of a Layer 2 or Layer 3 data packet from server 1, The leaf 1 VTEP adds the appropriate VXLAN header and then encapsulates the packet with a IPv4 outer header to facilitate tunneling the

packet through the IPv4 underlay network. The remote VTEP at leaf 2 then de-encapsulates the traffic and forwards the original packet towards the destination host. With the Junos software release 20.4 SRX Series Firewalls are able to perform tunnel inspection for VXLAN encapsulated overlay traffic passing through it.

In this example, you'll create a security policy to enable inspection for traffic that is encapsulated in a VXLAN tunnel . We're using the parameters described [Table 81 on page 1360](#) in this example.

Table 81: Configuration Parameters

Parameter	Description	Parameter Name
Security policy	Policy to create a flow session triggered by VXLAN overlay traffic. This policy references the outer IP source and destination address. That is, the IP addresses of the source and destination VTEPs. In this example this is the loopback address of the leaves.	P1
Policy set	Policy for the inspection of inner traffic. This policy operates on the contents of matching VXLAN tunnel traffic.	PSET-1
Tunnel inspection profile	Specifies parameters for security inspection on VXLAN tunnels.	TP-1
Name of a VXLAN network identifier (VNI) list or range	Used to uniquely identify a list or range of VXLAN tunnel IDs.	VLAN-100
VXLAN tunnel identifier name.	Used to symbolically name a VXLAN tunnel in a tunnel inspection profile.	VNI-1100

When you configure tunnel inspection security policies on the SRX Series Firewall, it decapsulates the packet to access the inner header when a packet matches a security policy. Next, it applies the tunnel inspection profile to determine if the inner traffic is permitted. The security device uses inner packet content and the applied tunnel inspection profile parameters to do a policy lookup and to then perform stateful inspection for the inner session.



TIP: In some configurations, the ingress and egress sides of a Type 5 VXLAN tunnel are placed into different VRF groups (Virtual Routing and Forwarding groups). This can cause problems in security policy matching, because the system may treat them as separate routing domains. If both sides of a Type 5 VXLAN tunnel use the same VNI, this issue is not applicable. In case if you use different VNIs on each side of the tunnel, you must change to using the same VNI to avoid issues.

To fix the issue without changing the VNI setup:

1. Put both ingress and egress VRF IDs into the same VRF group ID.
2. In the security policy, use the same source and destination VRF group ID as the matching criteria.

Configuration

In this example, you'll configure the following functionality on the SRX Series Firewall:

1. Define a trust and untrust zone to permit all host traffic. This supports the BGP session to the spine devices and allow SSH etc from either zone (DC).
2. Inspect traffic flowing from DC1 to DC2 in VNI 1100 (Layer 2 stretched for VLAN 100) for all hosts in the 192.168.100.0/24 subnet. Your policy should permit pings but deny all other traffic.
3. Allow all return traffic from DC2 to DC1 with no tunnel inspection.
4. Allow all other underlay and overlay traffic without VXLAN tunnel inspection from DC1 to DC2.

Use the following steps to enable tunnel inspection on your security device in a VXLAN-EVPN environment:



NOTE: Complete functional configurations for all devices used in this example are provided "[Complete Device Configurations](#)" on page 1381 to assist the reader in testing this example.

This example focuses on the configuration steps needed to enable and validate the VXLAN tunnel inspection feature. The SRX Series Firewall is presumed to be configured with interface addressing, BGP peering, and policy to support its DCI role.

CLI Quick Configuration

To quickly configure this example on your SRX Series Firewall, 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.

Configuration on SRX Series Device

```

set system host-name r4-dci-ebr
set security address-book global address vtep-untrust 10.255.2.0/24
set security address-book global address vtep-trust 10.255.1.0/24
set security address-book global address vlan100 192.168.100.0/24
set security policies from-zone trust to-zone untrust policy P1 match source-address vtep-trust
set security policies from-zone trust to-zone untrust policy P1 match destination-address vtep-
untrust
set security policies from-zone trust to-zone untrust policy P1 match application junos-vxlan
set security policies from-zone trust to-zone untrust policy P1 then permit tunnel-inspection
TP-1
set security policies from-zone untrust to-zone trust policy accept-all-dc2 match source-address
any
set security policies from-zone untrust to-zone trust policy accept-all-dc2 match destination-
address any
set security policies from-zone untrust to-zone trust policy accept-all-dc2 match application any
set security policies from-zone untrust to-zone trust policy accept-all-dc2 then permit
set security policies policy-set PSET-1 policy PSET-1-P1 match source-address vlan100
set security policies policy-set PSET-1 policy PSET-1-P1 match destination-address vlan100
set security policies policy-set PSET-1 policy PSET-1-P1 match application junos-icmp-all
set security policies policy-set PSET-1 policy PSET-1-P1 then permit
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/0.0
set security zones security-zone untrust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic protocols all
set security zones security-zone untrust interfaces ge-0/0/1.0
set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 policy-set PSET-1
set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 vni VLAN-100
set security tunnel-inspection vni VLAN-100 vni-id 1100
set interfaces ge-0/0/0 description "Link to DC1 Spine 1"
set interfaces ge-0/0/0 mtu 9000
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.2/30
set interfaces ge-0/0/1 description "Link to DC2 Spine 1"
set interfaces ge-0/0/1 mtu 9000
set interfaces ge-0/0/1 unit 0 family inet address 172.16.2.2/30

```

Step-by-Step Procedure

1. Configure security zones, interfaces, and address-books.

```
[edit]
user@@r4-dci-ebr# set security zones security-zone trust
user@@r4-dci-ebr# set security zones security-zone untrust
user@@r4-dci-ebr# set interfaces ge-0/0/0 description "Link to DC1 Spine 1"
user@@r4-dci-ebr# set interfaces ge-0/0/0 mtu 9000
user@@r4-dci-ebr# set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.2/30
user@@r4-dci-ebr# set interfaces ge-0/0/1 description "Link to DC2 Spine 1"
user@@r4-dci-ebr# set interfaces ge-0/0/1 mtu 9000
user@@r4-dci-ebr# set interfaces ge-0/0/1 unit 0 family inet address 172.16.2.2/30
user@@r4-dci-ebr# set security zones security-zone trust host-inbound-traffic system-services
all
user@@r4-dci-ebr# set security zones security-zone trust host-inbound-traffic protocols all
user@@r4-dci-ebr# set security zones security-zone trust interfaces ge-0/0/0.0
user@@r4-dci-ebr# set security zones security-zone untrust host-inbound-traffic system-
services all
user@@r4-dci-ebr# set security zones security-zone untrust host-inbound-traffic protocols all
user@@r4-dci-ebr# set security zones security-zone untrust interfaces ge-0/0/1.0
user@@r4-dci-ebr# set security address-book global address vtep-untrust 10.255.2.0/24
user@@r4-dci-ebr# set security address-book global address vtep-trust 10.255.1.0/24
user@@r4-dci-ebr# set security address-book global address vlan100 192.168.100.0/24
```

Note that /24 prefix lengths are used to specify the outer (VTEP) and inner (server) addresses. While you could use /32 host routes for this simple example, using a /24 will match traffic from other leaves (VTEPs) or hosts in the 192.168.100.0/24 subnet.

2. Define the tunnel-inspection profile. You can specify a range or a list of VNIs that should be inspected.

```
[edit]
user@@r4-dci-ebr# set security tunnel-inspection vni VLAN-100 vni-id 1100
user@@r4-dci-ebr# set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 vni
VLAN-100
user@@r4-dci-ebr# set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100
policy-set PSET-1
```

In this example only one VNI is needed so the vni-id keyword is used instead of the vni-range option.

The tunnel inspection profile links to both the VNI list/range as well as to the related policy that should be applied to VXLAN tunnel with matching VNIs.

3. Create a security policy to match on the outer session.

```
[edit]
user@@r4-dci-ebr# set security policies from-zone trust to-zone untrust policy P1 match
source-address vtep-trust
user@@r4-dci-ebr# set security policies from-zone trust to-zone untrust policy P1 match
destination-address vtep-untrust
user@@r4-dci-ebr# set security policies from-zone trust to-zone untrust policy P1 match
application junos-vxlan
user@@r4-dci-ebr# set security policies from-zone trust to-zone untrust policy P1 then permit
tunnel-inspection TP-1
```

This policy refers to the global address book entries you defined earlier to match source and destination VTEP addresses. These addresses are used in the underlay to support VXLAN tunnels in the overlay. Matching traffic is directed to the TP-1 tunnel inspection profile you defined in the previous step. In this example the goal is to inspect VXLAN tunnels that originate in DC1 and terminate in DC2. As a result a second policy to match on return traffic (with DC2 Leaf 1 the source VTEP) is not needed.

4. Create the policy-set for the inner session.

```
[edit]
user@@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match source-
address vlan100
user@@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match destination-
address vlan100
user@@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match application
junos-icmp-all
user@@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
```

This policy performs security inspection against the payload of matching VXLAN traffic. In this example this is the traffic sent from Server 1 on VLAN 100 in DC1 to Server 1 in DC2. By specifying the `junos-icmp-all` match condition you ensure that both ping request and replies can pass from server 1 in DC1 to server 1 in DC2. If you specify `junos-icmp-ping` only pings that originate from DC1 will be permitted.

Recall that in this example only ping is permitted to help facilitate testing of the resulting functionality. You can match on `application any` to permit all traffic, or alter the match criteria to suit your specific security needs.

5. Define the policies needed to accept all other traffic between the data centers without any tunnel inspection.

```
[edit]
user@@r4-dci-ebr# set security policies from-zone trust to-zone untrust policy accept-rest
match source-address any
user@@r4-dci-ebr# set security policies from-zone trust to-zone untrust policy accept-rest
match destination-address any
user@@r4-dci-ebr# set security policies from-zone trust to-zone untrust policy accept-rest
match application any
user@@r4-dci-ebr# set security policies from-zone trust to-zone untrust policy accept-rest
then permit
user@@r4-dci-ebr# set security policies from-zone untrust to-zone trust policy accept-all-dc2
match source-address any
user@@r4-dci-ebr# set security policies from-zone untrust to-zone trust policy accept-all-dc2
match destination-address any
user@@r4-dci-ebr# set security policies from-zone untrust to-zone trust policy accept-all-dc2
match application any
user@@r4-dci-ebr# set security policies from-zone untrust to-zone trust policy accept-all-dc2
then permit
```

Results

From configuration mode, confirm your configuration by entering the `show security` 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**

```
address-book {
  global {
    address vtep-untrust 10.255.2.0/24;
    address vtep-trust 10.255.1.0/24;
    address vlan100 192.168.100.0/24;
  }
}
policies {
  from-zone trust to-zone untrust {
    policy P1 {
```

```

        match {
            source-address vtep-trust;
            destination-address vtep-untrust;
            application junos-vxlan;
        }
        then {
            permit {
                tunnel-inspection {
                    TP-1;
                }
            }
        }
    }
}
policy accept-rest {
    match {
        source-address any;
        destination-address any;
        application any;
    }
    then {
        permit;
    }
}
}
from-zone untrust to-zone trust {
    policy accept-all-dc2 {
        match {
            source-address any;
            destination-address any;
            application any;
        }
        then {
            permit;
        }
    }
}
}
policy-set PSET-1 {
    policy PSET-1-P1 {
        match {
            source-address vlan100;
            destination-address vlan100;
            application junos-icmp-all;
        }
    }
}

```

```

        then {
            permit;
        }
    }
}
zones {
    security-zone trust {
        host-inbound-traffic {
            system-services {
                all;
            }
            protocols {
                all;
            }
        }
        interfaces {
            ge-0/0/0.0;
        }
    }
    security-zone untrust {
        host-inbound-traffic {
            system-services {
                all;
            }
            protocols {
                all;
            }
        }
        interfaces {
            ge-0/0/1.0;
        }
    }
}
tunnel-inspection {
    inspection-profile TP-1 {
        vxlan VNI-1100 {
            policy-set PSET-1;
            vni VLAN-100;
        }
    }
    vni VLAN-100 {
        vni-id 1100;
    }
}

```



```
}
}
```

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

Verification

At this time you should generate ping traffic between server 1 in DC1 to server 1 in DC2. The pings should succeed. Allow this test traffic to run in the background while you complete the verification tasks.

```
r5-dc1_server1> ping 192.168.100.102
PING 192.168.100.102 (192.168.100.102): 56 data bytes
64 bytes from 192.168.100.102: icmp_seq=0 ttl=64 time=565.451 ms
64 bytes from 192.168.100.102: icmp_seq=1 ttl=64 time=541.035 ms
64 bytes from 192.168.100.102: icmp_seq=2 ttl=64 time=651.420 ms
64 bytes from 192.168.100.102: icmp_seq=3 ttl=64 time=303.533 ms
. . .
```

Verify Inner Policy Details

Purpose

Verify the details of the policy applied for the inner session.

Action

From operational mode, enter the `show security policies policy-set PSET-1` command.

```
From zone: PSET-1, To zone: PSET-1
  Policy: PSET-1-P1, State: enabled, Index: 7, Scope Policy: 0, Sequence number: 1, Log Profile
  ID: 0
    From zones: any
    To zones: any
    Source vrf group: any
    Destination vrf group: any
    Source addresses: vlan100
    Destination addresses: vlan100
    Applications: junos-icmp-all
    Source identity feeds: any
```

```
Destination identity feeds: any
Action: permit
```

Check Tunnel Inspection Traffic

Purpose

Display the tunnel inspection traffic details.

Action

From operational mode, enter the `show security flow tunnel-inspection statistics` command.

```
Flow Tunnel-inspection statistics:
Tunnel-inspection type VXLAN:
  overlay session active:      4
  overlay session create:     289
  overlay session close:      285
  underlay session active:     3
  underlay session create:     31
  underlay session close:     28
  input packets:              607
  input bytes:                171835
  output packets:             418
  output bytes:               75627
  bypass packets:             0
  bypass bytes:               0
```

Check Tunnel Inspection Profile and VNI

Purpose

Display the tunnel inspection profile and VNI details.

Action

From operational mode, enter the `show security tunnel-inspection profiles` command.

```
Logical system: root-logical-system
Profile count: 1
Profile: TP-1
```

```
Type: VXLAN
Vxlan count: 1
Vxlan name: VXT-1
VNI count: 1
  VNI:VNI-1
  Policy set: PSET-1
  Inspection level: 1
```

From operational mode, enter the `show security tunnel-inspection vnis` command.

```
Logical system: root-logical-system
  VNI count: 2
  VNI name: VLAN-100
    VNI id count: 1
    [1100 - 1100]
  VNI name: VNI-1
    VNI id count: 1
    [1100 - 1100]
```

Check Security Flows

Purpose

Display VXLAN security flow information on the SRX to confirm that VXLAN tunnel inspection is working.

Action

From operational mode, enter the `show security flow session vxlan-vni 1100` command.

```
Session ID: 3811, Policy name: PSET-1-P1/7, State: Stand-alone, Timeout: 2, Valid
  In: 192.168.100.101/47883 --> 192.168.100.102/82;icmp, Conn Tag: 0xfcd, If: ge-0/0/0.0, Pkts:
  1, Bytes: 84,
  Type: VXLAN, VNI: 1100, Tunnel Session ID: 2193
  Out: 192.168.100.102/82 --> 192.168.100.101/47883;icmp, Conn Tag: 0xfcd, If: ge-0/0/1.0, Pkts:
  0, Bytes: 0,
  Type: VXLAN, VNI: 0, Tunnel Session ID: 0

Session ID: 3812, Policy name: PSET-1-P1/7, State: Stand-alone, Timeout: 2, Valid
  In: 192.168.100.101/47883 --> 192.168.100.102/83;icmp, Conn Tag: 0xfcd, If: ge-0/0/0.0, Pkts:
```

```

1, Bytes: 84,
  Type: VXLAN, VNI: 1100, Tunnel Session ID: 2193
  Out: 192.168.100.102/83 --> 192.168.100.101/47883;icmp, Conn Tag: 0xfcd, If: ge-0/0/1.0, Pkts:
0, Bytes: 0,
  Type: VXLAN, VNI: 0, Tunnel Session ID: 0

. . .

```

Confirm That SSH is Blocked

Purpose

Try to establish an SSH session between server 1 in DC1 and server 2 in DC2. Based on the policy that allows only ping traffic this session should be blocked at the SRX.

Action

From operational mode, enter the `show security flow session vxlan-vni 1100` command.

```

r5-dc1_server1> ssh 192.168.100.102
ssh: connect to host 192.168.100.102 port 22: Operation timed out
r5_dc1_server1>

```

Configuration for Zone-Level Inspection, IDP, Content Security and Advanced Anti-Malware for Tunnel Inspection

IN THIS SECTION

- [CLI Quick Configuration | 1372](#)
- [Create Zone-Level Inspection for Tunnel Inspection | 1373](#)
- [Create IDP, Content Security and Advanced Anti-Malware for Tunnel Inspection | 1374](#)
- [Results | 1376](#)

Use this step if you want to configure zone-level inspection, and apply layer 7 services such as IDP, Juniper ATP, Content Security, and advanced anti-malware to the tunnel traffic. This feature is supported from Junos OS Release 21.1R1 onwards.

This example uses the following hardware and software components:

- An SRX Series Firewall or vSRX Virtual Firewall
- Junos OS Release 21.1R1

We are using the same configuration of address-books, security zones, interfaces, tunnel inspection profile, and security policy for the outer session as created in ["Configuration" on page 1361](#)

This step assumes that you have Enrolled your SRX Series Firewall with Juniper ATP. For details on how to enroll your SRX Series Firewall, see [Enroll an SRX Series Firewall using Juniper ATP Cloud Web Portal](#).

In this configuration, you'll create a policy set for the inner session and apply IDP, Content Security, advanced antimalware to the tunnel traffic.

CLI Quick Configuration

To quickly configure this example on your SRX Series Firewall, 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.

Configuration on SRX Series Device

```
set system host-name r4-dci-ebr
set security address-book global address vtep-untrust 10.255.2.0/24
set security address-book global address vtep-trust 10.255.1.0/24
set security address-book global address vlan100 192.168.100.0/24
set security policies from-zone trust to-zone untrust policy P1 match source-address vtep-trust
set security policies from-zone trust to-zone untrust policy P1 match destination-address vtep-untrust
set security policies from-zone trust to-zone untrust policy P1 match application junos-vxlan
set security policies from-zone trust to-zone untrust policy P1 then permit tunnel-inspection
TP-1
set security policies from-zone untrust to-zone trust policy accept-all-dc2 match source-address
any
set security policies from-zone untrust to-zone trust policy accept-all-dc2 match destination-
address any
set security policies from-zone untrust to-zone trust policy accept-all-dc2 match application any
set security policies from-zone untrust to-zone trust policy accept-all-dc2 then permit
set security policies policy-set PSET-1 policy PSET-1-P1 match source-address vlan100
set security policies policy-set PSET-1 policy PSET-1-P1 match destination-address vlan100
set security policies policy-set PSET-1 policy PSET-1-P1 match application junos-icmp-all
set security policies policy-set PSET-1 policy PSET-1-P1 match dynamic-application any
set security policies policy-set PSET-1 policy PSET-1-P1 match url-category any
set security policies policy-set PSET-1 policy PSET-1-P1 match from-zone trust
set security policies policy-set PSET-1 policy PSET-1-P1 match to-zone untrust
```

```

set security policies policy-set PSET-1 policy PSET-1-P1 then permit
set security policies policy-set PSET-1 policy PSET-1-P1 then permit
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/0.0
set security zones security-zone untrust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic protocols all
set security zones security-zone untrust interfaces ge-0/0/1.0
set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 policy-set PSET-1
set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 vni VLAN-100
set security tunnel-inspection vni VLAN-100 vni-id 1100
set interfaces ge-0/0/0 description "Link to DC1 Spine 1"
set interfaces ge-0/0/0 mtu 9000
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.2/30
set interfaces ge-0/0/1 description "Link to DC2 Spine 1"
set interfaces ge-0/0/1 mtu 9000
set interfaces ge-0/0/1 unit 0 family inet address 172.16.2.2/30

```

Create Zone-Level Inspection for Tunnel Inspection

You can add zone-level policy control for EVPN-VXLAN tunnel inspection for the inner traffic. This policy performs security inspection against the payload of matching VXLAN traffic. In the following step, you'll specify from-zone and to-zone for the traffic.

- [edit]

```

user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match source-address vlan100
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match destination-address vlan100
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match application any
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match dynamic-application any
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match url-category any
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match from-zone trust
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match to-zone untrust
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 then permit

```

Create IDP, Content Security and Advanced Anti-Malware for Tunnel Inspection

You can add security services such as IDP, advanced anti-malware, Content Security, SSL proxy for the EVPN-VXLAN tunnel inspection for the inner traffic. This policy performs security inspection against the payload of matching VXLAN traffic.

In the following step, you'll enable service such as IDP, Content Security, SSL proxy, security-intelligence, advanced anti-malware services by specifying them in a security policy permit action, when the traffic matches the policy rule.

- ```
[edit]
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match source-
address vlan100
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match destination-
address vlan100
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 match application
any
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services ssl-proxy profile-name ssl-inspect-profile-1
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services security-intelligence-policy secintel1
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services advanced-anti-malware-policy P3
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services idp-policy idp123
user@r4-dci-ebr# set security policies policy-set PSET-1 policy PSET-1-P1 then permit
application-services utm-policy P1
```

The following steps show configuration snippets of Content Security, IDP, and advanced anti-malware policies at-glance.

- Configure advanced anti-malware policy.

```
[edit]
user@r4-dci-ebr# set services advanced-anti-malware policy P3 http inspection-profile
scripts
user@r4-dci-ebr# set services advanced-anti-malware policy P3 http action block
user@r4-dci-ebr# set services advanced-anti-malware policy P3 http notification log
user@r4-dci-ebr# set services advanced-anti-malware policy P3 http client-notify message
"AAMW Blocked!"
user@r4-dci-ebr# set services advanced-anti-malware policy P3 verdict-threshold recommended
user@r4-dci-ebr# set services advanced-anti-malware policy P3 fallback-options action permit
```

```
user@r4-dci-ebr# set services advanced-anti-malware policy P3 fallback-options notification
log
```

- Configure security intelligence profile.

```
[edit]
user@r4-dci-ebr# set services security-intelligence url https://
cloudfeeds.argonqa.junipersecurity.net/api/manifest.xml
user@r4-dci-ebr# set services security-intelligence authentication tls-profile aamw-ssl
user@r4-dci-ebr# set services security-intelligence profile cc_profile category CC
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 1
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 2
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 4
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 5
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 6
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 7
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 8
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 9
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule match
threat-level 10
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule then
action block close
user@r4-dci-ebr# set services security-intelligence profile cc_profile rule cc_rule then log
user@r4-dci-ebr# set services security-intelligence profile ih_profile category Infected-Hosts
user@r4-dci-ebr# set services security-intelligence profile ih_profile rule ih_rule match
threat-level 7
user@r4-dci-ebr# set services security-intelligence profile ih_profile rule ih_rule match
threat-level 8
user@r4-dci-ebr# set services security-intelligence profile ih_profile rule ih_rule match
threat-level 9
user@r4-dci-ebr# set services security-intelligence profile ih_profile rule ih_rule match
threat-level 10
user@r4-dci-ebr# set services security-intelligence profile ih_profile rule ih_rule then
action block close http message "Blocked!"
```



```

user@r4-dci-ebr# set services security-intelligence profile ih_profile rule ih_rule then log
user@r4-dci-ebr# set services security-intelligence policy secintel1 CC cc_profile
user@r4-dci-ebr# set services security-intelligence policy secintel1 Infected-Hosts ih_profile

```

- Configure IDP policy.

```

[edit]
user@r4-dci-ebr# set security idp idp-policy idp123 rulebase-ips rule rule1 match application
junos-icmp-all
user@r4-dci-ebr# set security idp idp-policy idp123 rulebase-ips rule rule1 then action no-
action

```

- Configure Content Security policy.

```

[edit]
user@r4-dci-ebr# set security utm default-configuration anti-virus type sophos-engine
user@r4-dci-ebr## set security utm utm-policy P1 anti-virus http-profile junos-sophos-av-
defaults

```

- Configure SSL profiles.

```

[edit]
user@r4-dci-ebr# set services ssl initiation profile aamw-ssl
user@r4-dci-ebr# set services ssl proxy profile ssl-inspect-profile-1 root-ca VJSA

```

## Results

From configuration mode, confirm your configuration by entering the `show security` 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**

```

address-book {
 global {
 address vtep-untrust 10.255.2.0/24;
 address vtep-trust 10.255.1.0/24;

```

```

 address vlan100 192.168.100.0/24;
 }
}
policies {
 from-zone trust to-zone untrust {
 policy P1 {
 match {
 source-address vtep-trust;
 destination-address vtep-untrust;
 application junos-vxlan;
 }
 then {
 permit {
 tunnel-inspection {
 TP-1;
 }
 }
 }
 }
 }
 policy accept-rest {
 match {
 source-address any;
 destination-address any;
 application any;
 }
 then {
 permit;
 }
 }
}
from-zone untrust to-zone trust {
 policy accept-all-dc2 {
 match {
 source-address any;
 destination-address any;
 application any;
 }
 then {
 permit;
 }
 }
}
policy-set PSET-1 {

```

```

policy PSET-1-P1 {
 match {
 source-address vlan100;
 destination-address vlan100;
 application junos-icmp-all;
 dynamic-application any;
 url-category any;
 from-zone trust;
 to-zone untrust;
 }
 then {
 permit {
 application-services {
 idp-policy idp123;
 ssl-proxy {
 profile-name ssl-inspect-profile-1;
 }
 utm-policy P1;
 security-intelligence-policy secintel1;
 advanced-anti-malware-policy P3;
 }
 }
 }
}

zones {
 security-zone trust {
 host-inbound-traffic {
 system-services {
 all;
 }
 protocols {
 all;
 }
 }
 interfaces {
 ge-0/0/0.0;
 }
 }
 security-zone untrust {
 host-inbound-traffic {

```

```

 system-services {
 all;
 }
 protocols {
 all;
 }
 }
 interfaces {
 ge-0/0/1.0;
 }
}
}
tunnel-inspection {
 inspection-profile TP-1 {
 vxlan VNI-1100 {
 policy-set PSET-1;
 vni VLAN-100;
 }
 }
 vni VLAN-100 {
 vni-id 1100;
 }
}
}

```

[edit]

user@host# **show services**

```

application-identification;
ssl {
 initiation {
 profile aamw-ssl;
 }
 proxy {
 profile ssl-inspect-profile-1 {
 root-ca VJSA;
 }
 }
}
advanced-anti-malware {
 policy P3 {
 http {

```

```

 inspection-profile scripts;
 action block;
 client-notify {
 message "AAMW Blocked!";
 }
 notification {
 log;
 }
 }
 verdict-threshold recommended;
 fallback-options {
 action permit;
 notification {
 log;
 }
 }
}
}
security-intelligence {
 url https://cloudfeeds.argonqa.junipersecurity.net/api/manifest.xml;
 authentication {
 tls-profile aamw-ssl;
 }
 profile cc_profile {
 category CC;
 rule cc_rule {
 match {
 threat-level [1 2 4 5 6 7 8 9 10];
 }
 then {
 action {
 block {
 close;
 }
 }
 log;
 }
 }
 }
 profile ih_profile {
 category Infected-Hosts;
 rule ih_rule {
 match {

```

```

 threat-level [7 8 9 10];
 }
 then {
 action {
 block {
 close {
 http {
 message "Blocked!";
 }
 }
 }
 }
 log;
 }
}
}
policy secintel1 {
 CC {
 cc_profile;
 }
 Infected-Hosts {
 ih_profile;
 }
}
}
}

```

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

## Complete Device Configurations

### IN THIS SECTION

- [Configuration on Leaf 1 Device | 1382](#)
- [Configuration on Spine 1 Device | 1384](#)
- [Configuration on Leaf 2 Device | 1386](#)
- [Configuration on Spine 2 Device | 1388](#)
- [Basic Tunnel Inspection Configuration on SRX Series Device | 1389](#)
- [Tunnel Inspection Configuration on SRX Series Device with Layer 7 Security Services | 1391](#)

Refer to these configurations to better understand or recreate the context of this example. They include the complete ERB based EVPN-VXLAN configurations for the QFX Series switches that form the DC fabrics, as well as the ending state of the SRX Series Firewall for both the basic and advanced VXLAN tunnel inspection examples.



**NOTE:** The provided configurations do not show user login, system logging, or management related configuration as this varies by location is not related to the VXLAN tunnel inspection feature.

For more details and example on configuring EVPN-VXLAN, see the network configuration example at [Configuring an EVPN-VXLAN Fabric for a Campus Network with ERB](#).

### Configuration on Leaf 1 Device

```
set system host-name r0_dc1_leaf1
set interfaces xe-0/0/0 mtu 9000
set interfaces xe-0/0/0 unit 0 family inet address 10.1.1.2/30
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members v100
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members v50
set interfaces irb unit 50 virtual-gateway-accept-data
set interfaces irb unit 50 family inet address 192.168.50.3/24 preferred
set interfaces irb unit 50 family inet address 192.168.50.3/24 virtual-gateway-address
192.168.50.1
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 family inet address 192.168.100.3/24 preferred
set interfaces irb unit 100 family inet address 192.168.100.3/24 virtual-gateway-address
192.168.100.1
set interfaces lo0 unit 0 family inet address 10.255.1.10/32
set interfaces lo0 unit 1 family inet address 10.255.10.10/32
set forwarding-options vxlan-routing next-hop 32768
set forwarding-options vxlan-routing overlay-ecmp
set policy-options policy-statement ECMP-POLICY then load-balance per-packet
set policy-options policy-statement FROM_Lo0 term 10 from interface lo0.0
set policy-options policy-statement FROM_Lo0 term 10 then accept
set policy-options policy-statement FROM_Lo0 term 20 then reject
set policy-options policy-statement OVERLAY_IMPORT term 5 from community comm_pod1
set policy-options policy-statement OVERLAY_IMPORT term 5 then accept
set policy-options policy-statement OVERLAY_IMPORT term 10 from community comm_pod2
set policy-options policy-statement OVERLAY_IMPORT term 10 then accept
set policy-options policy-statement OVERLAY_IMPORT term 20 from community shared_100_fm_pod2
set policy-options policy-statement OVERLAY_IMPORT term 20 from community shared_100_fm_pod1
```

```

set policy-options policy-statement OVERLAY_IMPORT term 20 then accept
set policy-options policy-statement T5_EXPORT term fm_direct from protocol direct
set policy-options policy-statement T5_EXPORT term fm_direct then accept
set policy-options policy-statement T5_EXPORT term fm_static from protocol static
set policy-options policy-statement T5_EXPORT term fm_static then accept
set policy-options policy-statement T5_EXPORT term fm_v4_host from protocol evpn
set policy-options policy-statement T5_EXPORT term fm_v4_host from route-filter 0.0.0.0/0 prefix-
length-range /32-/32
set policy-options policy-statement T5_EXPORT term fm_v4_host then accept
set policy-options policy-statement VRF1_T5_RT_EXPORT term t1 then community add target_t5_pod1
set policy-options policy-statement VRF1_T5_RT_EXPORT term t1 then accept
set policy-options policy-statement VRF1_T5_RT_IMPORT term t1 from community target_t5_pod1
set policy-options policy-statement VRF1_T5_RT_IMPORT term t1 then accept
set policy-options policy-statement VRF1_T5_RT_IMPORT term t2 from community target_t5_pod2
set policy-options policy-statement VRF1_T5_RT_IMPORT term t2 then accept
set policy-options community comm_pod1 members target:65001:1
set policy-options community comm_pod2 members target:65002:2
set policy-options community shared_100_fm_pod1 members target:65001:100
set policy-options community shared_100_fm_pod2 members target:65002:100
set policy-options community target_t5_pod1 members target:65001:9999
set policy-options community target_t5_pod2 members target:65002:9999
set routing-instances TENANT_1_VRF routing-options multipath
set routing-instances TENANT_1_VRF protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances TENANT_1_VRF protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances TENANT_1_VRF protocols evpn ip-prefix-routes vni 9999
set routing-instances TENANT_1_VRF protocols evpn ip-prefix-routes export T5_EXPORT
set routing-instances TENANT_1_VRF instance-type vrf
set routing-instances TENANT_1_VRF interface irb.50
set routing-instances TENANT_1_VRF interface irb.100
set routing-instances TENANT_1_VRF interface lo0.1
set routing-instances TENANT_1_VRF route-distinguisher 10.255.1.10:9999
set routing-instances TENANT_1_VRF vrf-import VRF1_T5_RT_IMPORT
set routing-instances TENANT_1_VRF vrf-export VRF1_T5_RT_EXPORT
set routing-instances TENANT_1_VRF vrf-table-label
set routing-options router-id 10.255.1.10
set routing-options autonomous-system 65001
set routing-options forwarding-table export ECMP-POLICY
set routing-options forwarding-table ecmp-fast-reroute
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols bgp group EVPN_FABRIC type internal
set protocols bgp group EVPN_FABRIC local-address 10.255.1.10
set protocols bgp group EVPN_FABRIC family evpn signaling
set protocols bgp group EVPN_FABRIC multipath

```



```

set protocols bgp group EVPN_FABRIC bfd-liveness-detection minimum-interval 1000
set protocols bgp group EVPN_FABRIC bfd-liveness-detection multiplier 3
set protocols bgp group EVPN_FABRIC neighbor 10.255.1.1
set protocols bgp group UNDERLAY type external
set protocols bgp group UNDERLAY family inet unicast
set protocols bgp group UNDERLAY export FROM_Lo0
set protocols bgp group UNDERLAY local-as 65510
set protocols bgp group UNDERLAY multipath multiple-as
set protocols bgp group UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group UNDERLAY neighbor 10.1.1.1 peer-as 65511
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn vni-options vni 150 vrf-target target:65001:150
set protocols evpn vni-options vni 1100 vrf-target target:65001:100
set protocols evpn extended-vni-list 1100
set protocols evpn extended-vni-list 150
set protocols lldp interface all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.255.1.10:1
set switch-options vrf-import OVERLAY_IMPORT
set switch-options vrf-target target:65001:1
set vlans v100 vlan-id 100
set vlans v100 l3-interface irb.100
set vlans v100 vxlan vni 1100
set vlans v50 vlan-id 50
set vlans v50 l3-interface irb.50
set vlans v50 vxlan vni 150

```

### Configuration on Spine 1 Device

```

set system host-name r1_dc1_spine11
set interfaces xe-0/0/0 mtu 9000
set interfaces xe-0/0/0 unit 0 family inet address 10.1.1.1/30
set interfaces xe-0/0/1 mtu 9000
set interfaces xe-0/0/1 unit 0 family inet address 172.16.1.1/30
set interfaces lo0 unit 0 family inet address 10.255.1.1/32
set policy-options policy-statement ECMP-POLICY then load-balance per-packet
set policy-options policy-statement FROM_Lo0 term 10 from interface lo0.0
set policy-options policy-statement FROM_Lo0 term 10 then accept
set policy-options policy-statement FROM_Lo0 term 20 then reject

```

```

set policy-options policy-statement UNDERLAY-EXPORT term LOOPBACK from route-filter
10.255.0.0/16 orlonger
set policy-options policy-statement UNDERLAY-EXPORT term LOOPBACK from route-filter 10.1.0.0/16
orlonger
set policy-options policy-statement UNDERLAY-EXPORT term LOOPBACK then accept
set policy-options policy-statement UNDERLAY-EXPORT term DEFAULT then reject
set policy-options policy-statement UNDERLAY-IMPORT term LOOPBACK from route-filter
10.255.0.0/16 orlonger
set policy-options policy-statement UNDERLAY-IMPORT term LOOPBACK from route-filter 10.1.0.0/16
orlonger
set policy-options policy-statement UNDERLAY-IMPORT term LOOPBACK then accept
set policy-options policy-statement UNDERLAY-IMPORT term DEFAULT then reject
set routing-options autonomous-system 65001
set routing-options forwarding-table export ECMP-POLICY
set routing-options forwarding-table ecmp-fast-reroute
set protocols bgp group EVPN_FABRIC type internal
set protocols bgp group EVPN_FABRIC local-address 10.255.1.1
set protocols bgp group EVPN_FABRIC family evpn signaling
set protocols bgp group EVPN_FABRIC cluster 10.255.1.1
set protocols bgp group EVPN_FABRIC multipath
set protocols bgp group EVPN_FABRIC bfd-liveness-detection minimum-interval 1000
set protocols bgp group EVPN_FABRIC bfd-liveness-detection multiplier 3
set protocols bgp group EVPN_FABRIC neighbor 10.255.1.10
set protocols bgp group EVPN_FABRIC vpn-apply-export
set protocols bgp group UNDERLAY type external
set protocols bgp group UNDERLAY import UNDERLAY-IMPORT
set protocols bgp group UNDERLAY family inet unicast
set protocols bgp group UNDERLAY export UNDERLAY-EXPORT
set protocols bgp group UNDERLAY local-as 65511
set protocols bgp group UNDERLAY multipath multiple-as
set protocols bgp group UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group UNDERLAY neighbor 10.1.1.2 peer-as 65510
set protocols bgp group UNDERLAY neighbor 172.16.1.2 peer-as 65012
set protocols bgp group OVERLAY_INTERDC type external
set protocols bgp group OVERLAY_INTERDC multihop no-nexthop-change
set protocols bgp group OVERLAY_INTERDC local-address 10.255.1.1
set protocols bgp group OVERLAY_INTERDC family evpn signaling
set protocols bgp group OVERLAY_INTERDC multipath multiple-as
set protocols bgp group OVERLAY_INTERDC neighbor 10.255.2.1 peer-as 65002
set protocols lldp interface all

```

## Configuration on Leaf 2 Device

```

set system host-name r2_dc2_leaf1
set interfaces xe-0/0/0 mtu 9000
set interfaces xe-0/0/0 unit 0 family inet address 10.1.2.2/30
set interfaces xe-0/0/1 unit 0 family ethernet-switching vlan members v100
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan members v60
set interfaces irb unit 60 virtual-gateway-accept-data
set interfaces irb unit 60 family inet address 192.168.60.3/24 preferred
set interfaces irb unit 60 family inet address 192.168.60.3/24 virtual-gateway-address
192.168.60.1
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 family inet address 192.168.100.4/24 preferred
set interfaces irb unit 100 family inet address 192.168.100.4/24 virtual-gateway-address
192.168.100.1
set interfaces lo0 unit 0 family inet address 10.255.2.10/32
set interfaces lo0 unit 1 family inet address 10.255.20.10/32
set forwarding-options vxlan-routing next-hop 32768
set forwarding-options vxlan-routing overlay-ecmp
set policy-options policy-statement ECMP-POLICY then load-balance per-packet
set policy-options policy-statement FROM_Lo0 term 10 from interface lo0.0
set policy-options policy-statement FROM_Lo0 term 10 then accept
set policy-options policy-statement FROM_Lo0 term 20 then reject
set policy-options policy-statement OVERLAY_IMPORT term 5 from community comm_pod1
set policy-options policy-statement OVERLAY_IMPORT term 5 then accept
set policy-options policy-statement OVERLAY_IMPORT term 10 from community comm_pod2
set policy-options policy-statement OVERLAY_IMPORT term 10 then accept
set policy-options policy-statement OVERLAY_IMPORT term 20 from community shared_100_fm_pod2
set policy-options policy-statement OVERLAY_IMPORT term 20 from community shared_100_fm_pod1
set policy-options policy-statement OVERLAY_IMPORT term 20 then accept
set policy-options policy-statement T5_EXPORT term fm_direct from protocol direct
set policy-options policy-statement T5_EXPORT term fm_direct then accept
set policy-options policy-statement T5_EXPORT term fm_static from protocol static
set policy-options policy-statement T5_EXPORT term fm_static then accept
set policy-options policy-statement T5_EXPORT term fm_v4_host from protocol evpn
set policy-options policy-statement T5_EXPORT term fm_v4_host from route-filter 0.0.0.0/0 prefix-
length-range /32-/32
set policy-options policy-statement T5_EXPORT term fm_v4_host then accept
set policy-options policy-statement VRF1_T5_RT_EXPORT term t1 then community add target_t5_pod1
set policy-options policy-statement VRF1_T5_RT_EXPORT term t1 then accept
set policy-options policy-statement VRF1_T5_RT_IMPORT term t1 from community target_t5_pod1
set policy-options policy-statement VRF1_T5_RT_IMPORT term t1 then accept

```

```

set policy-options policy-statement VRF1_T5_RT_IMPORT term t2 from community target_t5_pod2
set policy-options policy-statement VRF1_T5_RT_IMPORT term t2 then accept
set policy-options community comm_pod1 members target:65001:1
set policy-options community comm_pod2 members target:65002:2
set policy-options community shared_100_fm_pod1 members target:65001:100
set policy-options community shared_100_fm_pod2 members target:65002:100
set policy-options community target_t5_pod1 members target:65001:9999
set policy-options community target_t5_pod2 members target:65002:9999
set routing-instances TENANT_1_VRF routing-options multipath
set routing-instances TENANT_1_VRF protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances TENANT_1_VRF protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances TENANT_1_VRF protocols evpn ip-prefix-routes vni 9999
set routing-instances TENANT_1_VRF protocols evpn ip-prefix-routes export T5_EXPORT
set routing-instances TENANT_1_VRF instance-type vrf
set routing-instances TENANT_1_VRF interface irb.60
set routing-instances TENANT_1_VRF interface irb.100
set routing-instances TENANT_1_VRF interface lo0.1
set routing-instances TENANT_1_VRF route-distinguisher 10.255.1.2:9999
set routing-instances TENANT_1_VRF vrf-import VRF1_T5_RT_IMPORT
set routing-instances TENANT_1_VRF vrf-export VRF1_T5_RT_EXPORT
set routing-instances TENANT_1_VRF vrf-table-label
set routing-options router-id 10.255.2.10
set routing-options autonomous-system 65002
set routing-options forwarding-table export ECMP-POLICY
set routing-options forwarding-table ecmp-fast-reroute
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols bgp group EVPN_FABRIC type internal
set protocols bgp group EVPN_FABRIC local-address 10.255.2.10
set protocols bgp group EVPN_FABRIC family evpn signaling
set protocols bgp group EVPN_FABRIC multipath
set protocols bgp group EVPN_FABRIC bfd-liveness-detection minimum-interval 1000
set protocols bgp group EVPN_FABRIC bfd-liveness-detection multiplier 3
set protocols bgp group EVPN_FABRIC neighbor 10.255.2.1
set protocols bgp group UNDERLAY type external
set protocols bgp group UNDERLAY family inet unicast
set protocols bgp group UNDERLAY export FROM_Lo0
set protocols bgp group UNDERLAY local-as 65522
set protocols bgp group UNDERLAY multipath multiple-as
set protocols bgp group UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group UNDERLAY neighbor 10.1.2.1 peer-as 65523
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community

```

```

set protocols evpn vni-options vni 160 vrf-target target:65002:160
set protocols evpn vni-options vni 1100 vrf-target target:65002:100
set protocols evpn extended-vni-list 1100
set protocols evpn extended-vni-list 160
set protocols lldp interface all
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.255.2.10:1
set switch-options vrf-import OVERLAY_IMPORT
set switch-options vrf-target target:65002:1
set vlans v100 vlan-id 100
set vlans v100 l3-interface irb.100
set vlans v100 vxlan vni 1100
set vlans v60 vlan-id 60
set vlans v60 l3-interface irb.60
set vlans v60 vxlan vni 160

```

### Configuration on Spine 2 Device

```

set system host-name r3_dc2_spine1
set interfaces xe-0/0/0 mtu 9000
set interfaces xe-0/0/0 unit 0 family inet address 10.1.2.1/30
set interfaces xe-0/0/1 mtu 9000
set interfaces xe-0/0/1 unit 0 family inet address 172.16.2.1/30
set interfaces lo0 unit 0 family inet address 10.255.2.1/32
set policy-options policy-statement ECMP-POLICY then load-balance per-packet
set policy-options policy-statement FROM_Lo0 term 10 from interface lo0.0
set policy-options policy-statement FROM_Lo0 term 10 then accept
set policy-options policy-statement FROM_Lo0 term 20 then reject
set policy-options policy-statement UNDERLAY-EXPORT term LOOPBACK from route-filter
10.255.0.0/16 orlonger
set policy-options policy-statement UNDERLAY-EXPORT term LOOPBACK from route-filter 10.1.0.0/16
orlonger
set policy-options policy-statement UNDERLAY-EXPORT term LOOPBACK then accept
set policy-options policy-statement UNDERLAY-EXPORT term DEFAULT then reject
set policy-options policy-statement UNDERLAY-IMPORT term LOOPBACK from route-filter
10.255.0.0/16 orlonger
set policy-options policy-statement UNDERLAY-IMPORT term LOOPBACK from route-filter 10.1.0.0/16
orlonger
set policy-options policy-statement UNDERLAY-IMPORT term LOOPBACK then accept
set policy-options policy-statement UNDERLAY-IMPORT term DEFAULT then reject
set routing-options autonomous-system 65002

```

```

set routing-options forwarding-table export ECMP-POLICY
set routing-options forwarding-table ecmp-fast-reroute
set protocols bgp group EVPN_FABRIC type internal
set protocols bgp group EVPN_FABRIC local-address 10.255.2.1
set protocols bgp group EVPN_FABRIC family evpn signaling
set protocols bgp group EVPN_FABRIC cluster 10.255.2.1
set protocols bgp group EVPN_FABRIC multipath
set protocols bgp group EVPN_FABRIC bfd-liveness-detection minimum-interval 1000
set protocols bgp group EVPN_FABRIC bfd-liveness-detection multiplier 3
set protocols bgp group EVPN_FABRIC neighbor 10.255.2.10
set protocols bgp group EVPN_FABRIC vpn-apply-export
set protocols bgp group UNDERLAY type external
set protocols bgp group UNDERLAY import UNDERLAY-IMPORT
set protocols bgp group UNDERLAY family inet unicast
set protocols bgp group UNDERLAY export UNDERLAY-EXPORT
set protocols bgp group UNDERLAY local-as 65523
set protocols bgp group UNDERLAY multipath multiple-as
set protocols bgp group UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group UNDERLAY neighbor 10.1.2.2 peer-as 65522
set protocols bgp group UNDERLAY neighbor 172.16.2.2 peer-as 65012
set protocols bgp group OVERLAY_INTERDC type external
set protocols bgp group OVERLAY_INTERDC multihop no-nexthop-change
set protocols bgp group OVERLAY_INTERDC local-address 10.255.2.1
set protocols bgp group OVERLAY_INTERDC family evpn signaling
set protocols bgp group OVERLAY_INTERDC multipath multiple-as
set protocols bgp group OVERLAY_INTERDC neighbor 10.255.1.1 peer-as 65001
set protocols lldp interface all

```

### Basic Tunnel Inspection Configuration on SRX Series Device

```

set system host-name r4-dci-ebr
set security address-book global address vtep-untrust 10.255.2.0/24
set security address-book global address vtep-trust 10.255.1.0/24
set security address-book global address vlan100 192.168.100.0/24
set security policies from-zone trust to-zone untrust policy P1 match source-address vtep-trust
set security policies from-zone trust to-zone untrust policy P1 match destination-address vtep-untrust
set security policies from-zone trust to-zone untrust policy P1 match application junos-vxlan
set security policies from-zone trust to-zone untrust policy P1 then permit tunnel-inspection
TP-1

```

```

set security policies from-zone trust to-zone untrust policy accept-rest match source-address any
set security policies from-zone trust to-zone untrust policy accept-rest match destination-
address any
set security policies from-zone trust to-zone untrust policy accept-rest match application any
set security policies from-zone trust to-zone untrust policy accept-rest then permit
set security policies from-zone untrust to-zone trust policy accept-return match source-address
any
set security policies from-zone untrust to-zone trust policy accept-return match destination-
address any
set security policies from-zone untrust to-zone trust policy accept-return match application any
set security policies from-zone untrust to-zone trust policy accept-return then permit
set security policies policy-set PSET-1 policy PSET-1-P1 match source-address vlan100
set security policies policy-set PSET-1 policy PSET-1-P1 match destination-address vlan100
set security policies policy-set PSET-1 policy PSET-1-P1 match application junos-icmp-all
set security policies policy-set PSET-1 policy PSET-1-P1 then permit
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/0.0
set security zones security-zone untrust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic protocols all
set security zones security-zone untrust interfaces ge-0/0/1.0
set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 policy-set PSET-1
set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 vni VLAN-100
set security tunnel-inspection vni VLAN-100 vni-id 1100
set interfaces ge-0/0/0 description "Link to DC2 Spine 1"
set interfaces ge-0/0/0 mtu 9000
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.2/30
set interfaces ge-0/0/1 mtu 9000
set interfaces ge-0/0/1 unit 0 family inet address 172.16.2.2/30
set policy-options policy-statement ECMP-POLICY then load-balance per-packet
set policy-options policy-statement dci term 1 from protocol direct
set policy-options policy-statement dci term 1 then accept
set protocols bgp group UNDERLAY export dci
set protocols bgp group UNDERLAY multipath multiple-as
set protocols bgp group UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group UNDERLAY neighbor 172.16.1.1 peer-as 65511
set protocols bgp group UNDERLAY neighbor 172.16.2.1 peer-as 65523
set routing-options autonomous-system 65012
set routing-options forwarding-table export ECMP-POLICY

```

## Tunnel Inspection Configuration on SRX Series Device with Layer 7 Security Services

```

set system host-name r4-dci-ebr
set services application-identification
set services ssl initiation profile aamw-ssl
set services ssl proxy profile ssl-inspect-profile-1 root-ca VJSA
set services advanced-anti-malware policy P3 http inspection-profile scripts
set services advanced-anti-malware policy P3 http action block
set services advanced-anti-malware policy P3 http client-notify message "AAMW Blocked!"
set services advanced-anti-malware policy P3 http notification log
set services advanced-anti-malware policy P3 verdict-threshold recommended
set services advanced-anti-malware policy P3 fallback-options action permit
set services advanced-anti-malware policy P3 fallback-options notification log
set services security-intelligence url https://cloudfeeds.argonqa.junipersecurity.net/api/manifest.xml
set services security-intelligence authentication tls-profile aamw-ssl
set services security-intelligence profile cc_profile category CC
set services security-intelligence profile cc_profile rule cc_rule match threat-level 1
set services security-intelligence profile cc_profile rule cc_rule match threat-level 2
set services security-intelligence profile cc_profile rule cc_rule match threat-level 4
set services security-intelligence profile cc_profile rule cc_rule match threat-level 5
set services security-intelligence profile cc_profile rule cc_rule match threat-level 6
set services security-intelligence profile cc_profile rule cc_rule match threat-level 7
set services security-intelligence profile cc_profile rule cc_rule match threat-level 8
set services security-intelligence profile cc_profile rule cc_rule match threat-level 9
set services security-intelligence profile cc_profile rule cc_rule match threat-level 10
set services security-intelligence profile cc_profile rule cc_rule then action block close
set services security-intelligence profile cc_profile rule cc_rule then log
set services security-intelligence profile ih_profile category Infected-Hosts
set services security-intelligence profile ih_profile rule ih_rule match threat-level 7
set services security-intelligence profile ih_profile rule ih_rule match threat-level 8
set services security-intelligence profile ih_profile rule ih_rule match threat-level 9
set services security-intelligence profile ih_profile rule ih_rule match threat-level 10
set services security-intelligence profile ih_profile rule ih_rule then action block close http
message "Blocked!"
set services security-intelligence profile ih_profile rule ih_rule then log
set services security-intelligence policy secintel1 CC cc_profile
set services security-intelligence policy secintel1 Infected-Hosts ih_profile
set security pki ca-profile aamw-ca ca-identity deviceCA
set security pki ca-profile aamw-ca enrollment url http://ca.junipersecurity.net:8080/ejbca/publicweb/apply/scep/SRX/pkiclient.exe
set security pki ca-profile aamw-ca revocation-check disable
set security pki ca-profile aamw-ca revocation-check crl url http://va.junipersecurity.net/ca/

```



```

deviceCA.crl
set security pki ca-profile aamw-secintel-ca ca-identity JUNIPER
set security pki ca-profile aamw-secintel-ca revocation-check crl url http://
va.junipersecurity.net/ca/current.crl
set security pki ca-profile aamw-cloud-ca ca-identity JUNIPER_CLOUD
set security pki ca-profile aamw-cloud-ca revocation-check crl url http://
va.junipersecurity.net/ca/cloudCA.crl
set security idp idp-policy idp123 rulebase-ips rule rule1 match application junos-icmp-all
set security idp idp-policy idp123 rulebase-ips rule rule1 then action no-action
set security address-book global address vtep-untrust 10.255.2.0/24
set security address-book global address vtep-trust 10.255.1.0/24
set security address-book global address vlan100 192.168.100.0/24
set security utm default-configuration anti-virus type sophos-engine
set security utm utm-policy P1 anti-virus http-profile junos-sophos-av-defaults
set security policies from-zone trust to-zone untrust policy P1 match source-address vtep-trust
set security policies from-zone trust to-zone untrust policy P1 match destination-address vtep-
untrust
set security policies from-zone trust to-zone untrust policy P1 match application junos-vxlan
set security policies from-zone trust to-zone untrust policy P1 then permit tunnel-inspection
TP-1
set security policies from-zone trust to-zone untrust policy accept-rest match source-address any
set security policies from-zone trust to-zone untrust policy accept-rest match destination-
address any
set security policies from-zone trust to-zone untrust policy accept-rest match application any
set security policies from-zone trust to-zone untrust policy accept-rest then permit
set security policies from-zone untrust to-zone trust policy accept-return match source-address
any
set security policies from-zone untrust to-zone trust policy accept-return match destination-
address any
set security policies from-zone untrust to-zone trust policy accept-return match application any
set security policies from-zone untrust to-zone trust policy accept-return then permit
set security policies policy-set PSET-1 policy PSET-1-P1 match source-address vlan100
set security policies policy-set PSET-1 policy PSET-1-P1 match destination-address vlan100
set security policies policy-set PSET-1 policy PSET-1-P1 match application any
set security policies policy-set PSET-1 policy PSET-1-P1 match dynamic-application any
set security policies policy-set PSET-1 policy PSET-1-P1 match url-category any
set security policies policy-set PSET-1 policy PSET-1-P1 match from-zone trust
set security policies policy-set PSET-1 policy PSET-1-P1 match to-zone untrust
set security policies policy-set PSET-1 policy PSET-1-P1 then permit application-services idp-
policy idp123
set security policies policy-set PSET-1 policy PSET-1-P1 then permit application-services ssl-
proxy profile-name ssl-inspect-profile-1
set security policies policy-set PSET-1 policy PSET-1-P1 then permit application-services utm-

```

```

policy P1
set security policies policy-set PSET-1 policy PSET-1-P1 then permit application-services
security-intelligence-policy secintel1
set security policies policy-set PSET-1 policy PSET-1-P1 then permit application-services
advanced-anti-malware-policy P3
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/0.0
set security zones security-zone untrust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic protocols all
set security zones security-zone untrust interfaces ge-0/0/1.0
set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 policy-set PSET-1
set security tunnel-inspection inspection-profile TP-1 vxlan VNI-1100 vni VLAN-100
set security tunnel-inspection vni VLAN-100 vni-id 1100
set interfaces ge-0/0/0 description "Link to DC2 Spine 1"
set interfaces ge-0/0/0 mtu 9000
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.2/30
set interfaces ge-0/0/1 mtu 9000
set interfaces ge-0/0/1 unit 0 family inet address 172.16.2.2/30
set policy-options policy-statement ECMP-POLICY then load-balance per-packet
set policy-options policy-statement dci term 1 from protocol direct
set policy-options policy-statement dci term 1 then accept
set protocols bgp group UNDERLAY export dci
set protocols bgp group UNDERLAY multipath multiple-as
set protocols bgp group UNDERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group UNDERLAY bfd-liveness-detection multiplier 3
set protocols bgp group UNDERLAY neighbor 172.16.1.1 peer-as 65511
set protocols bgp group UNDERLAY neighbor 172.16.2.1 peer-as 65523
set routing-options autonomous-system 65012
set routing-options static route 0.0.0.0/0 next-hop 10.9.159.252
set routing-options forwarding-table export ECMP-POLICY

```

# Fault Detection and Isolation in EVPN-VXLAN Fabrics

## IN THIS CHAPTER

- [Understanding Overlay ping and traceroute Packet Support | 1394](#)
- [Example: Troubleshoot a VXLAN Overlay Network with Overlay Ping and Overlay Traceroute on QFX Series Switches | 1398](#)
- [Example: Troubleshoot a VXLAN Overlay Network with Overlay Ping and Overlay Traceroute for MX Series Routers | 1413](#)

## Understanding Overlay ping and traceroute Packet Support

### IN THIS SECTION

- [Overlay ping and traceroute Functionality | 1395](#)
- [Overlay OAM Packet Format for UDP Payloads | 1395](#)

In a virtualized overlay network, existing ping and traceroute mechanisms do not provide enough information to determine whether or not connectivity is established throughout the network. The existing ping and traceroute commands can only verify the basic connectivity between two endpoints in the underlying physical network, but not in the overlay network. For example, you can issue the existing ping command on a Juniper Networks device that functions as a virtual tunnel endpoint (VTEP) to another Juniper Networks device that also functions as a VTEP in a Virtual Extensible LAN (VXLAN) overlay. In this situation, the ping output might indicate that the connection between the source and destination VTEPs is up and running despite the fact that one of the endpoints (physical servers upon which applications directly run) is not reachable.

Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, Release 16.1 for EX9200 switches, and Release 16.2 for MX Series routers, overlay ping and traceroute are introduced as troubleshooting tools for overlay networks.

For ping and traceroute mechanisms to work in overlay networks, the ping and traceroute packets, also referred to collectively as Operations, Administration, and Management (OAM) packets, must be encapsulated with the same VXLAN UDP headers (outer headers) as the data packets forwarded over the overlay segment. This implementation ensures that transit nodes forward the OAM packets in the same way as a data packet for that particular overlay segment.

If any connectivity issues arise for a particular data flow, the overlay OAM packet corresponding to the flow would experience the same connectivity issues as the data packet for that flow.

When using ping overlay and traceroute overlay, keep the following in mind:

- The only tunnel type supported is VXLAN tunnels.
- The VTEPs in the overlay network that send and receive the overlay ping packets must be Juniper Networks devices that support overlay ping and traceroute.

## Overlay ping and traceroute Functionality

Overlay ping and traceroute packets are sent as User Datagram Protocol (UDP) echo requests and replies and are encapsulated in the VXLAN header. VTEPs, which initiate and terminate overlay tunnels, send and receive overlay OAM packets. Overlay ping and traceroute are supported only in VXLAN overlay networks in which the sending and receiving VTEPs are both Juniper Networks devices.

The overlay ping functionality validates both the data plane and the MAC address and IP address of the VTEPs. This additional validation is different from the more commonly known IP ping functionality where the actual destination replies to the echo request without the overlay segment context.

While tracing a route in a VXLAN overlay network, Juniper Networks devices that are along the route that support overlay traceroute additionally provide a timestamp. Third-party devices and Juniper Networks devices that do not support overlay traceroute do not provide this timestamp.

## Overlay OAM Packet Format for UDP Payloads

The format of overlay OAM packets depends on the type of payload that is carried in the tunnel. In the case of VXLAN tunnels, the inner packet is a Layer 2 packet.



**NOTE:** Only Layer 2 UDP payloads are supported.

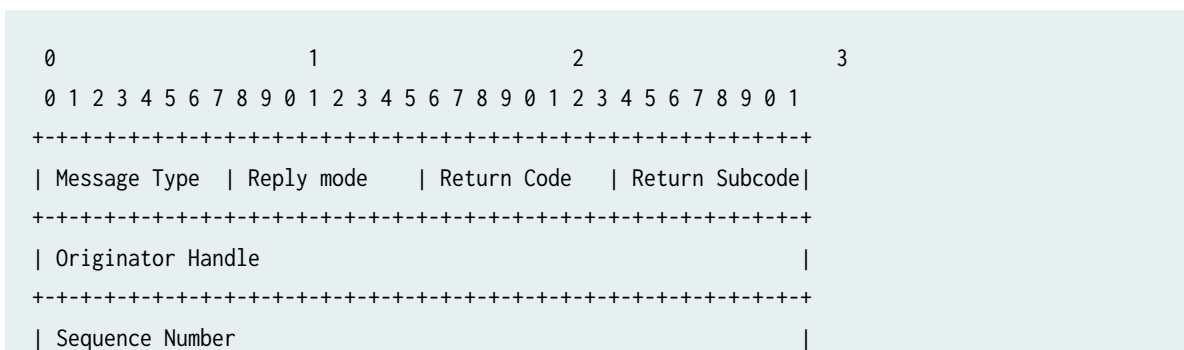
Figure 134 on page 1396 shows complete headers on a VXLAN-encapsulated overlay OAM packet.

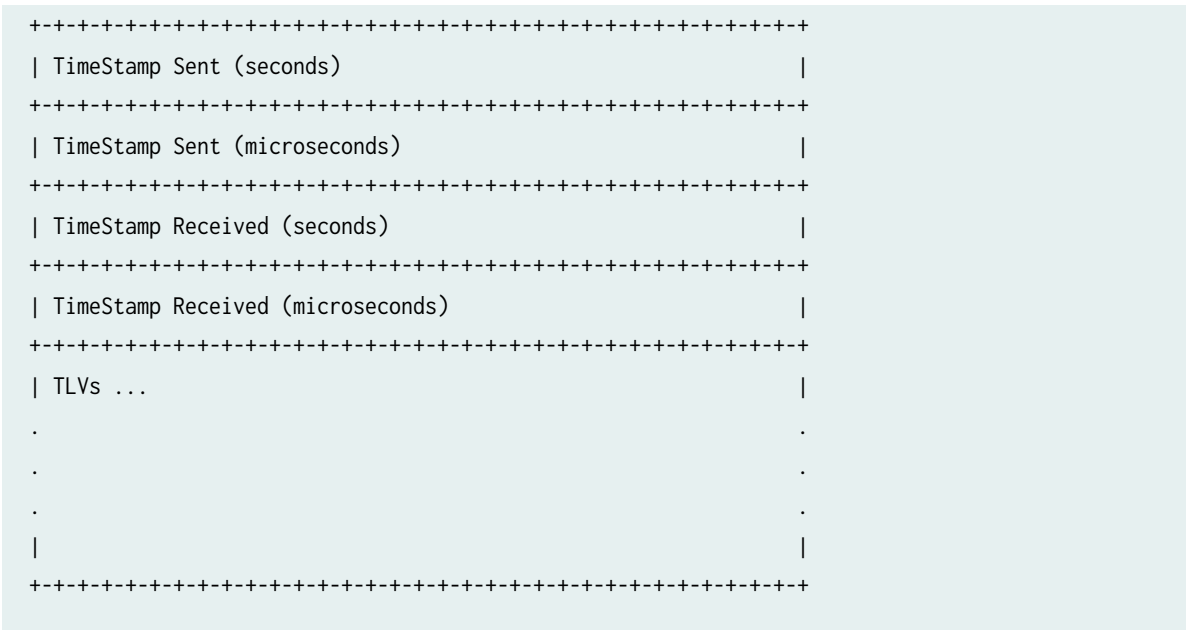
Figure 134: VXLAN-Encapsulated Overlay OAM Packet



- Outer Ethernet header—Contains the source MAC (SMAC) and destination MAC (DMAC) addresses of directly connected nodes in the physical network. These addresses change at every hop.
- Outer IP header—Contains the source and destination IP addresses of the Juniper Networks devices that function as the VTEPs that initiate and terminate the tunnel.
- Outer UDP header—Contains the source port associated with the flow entropy and destination port. The source port is an internally calculated hash value. The destination port is the standard UDP port (4789) used for VXLAN.
- VXLAN header—Contains the VXLAN Network Identifier (VNI) or the segment ID of the VXLAN, and new router alert (RA) flag bits.
- Inner Ethernet header—Contains a control MAC address (00-00-5E-90-xx-xx) for both the SMAC and DMAC. This address is not forwarded out of the VTEP. Alternatively, the SMAC can be set to a non-control MAC address. However, if a non-control MAC address is used, the VTEP must not learn the SMAC from the overlay OAM packets.
- Inner IP header—Contains the source IP address that can be set to the IP address of the endpoint or source VTEP. The destination IP address can be set to the 127/8 address, which ensures that the overlay OAM packet is not forwarded out of the ports of the Juniper Networks device that is configured as a VTEP.
- Inner UDP header—Contains a new reserved value used in the destination port field in the inner UDP header. This value identifies the incoming UDP packet as an overlay OAM packet.
- Inner UDP payload—Contains all of the overlay OAM-specific message format and type, length, and value (TLV) definitions.

The Inner UDP payload format is as follows:





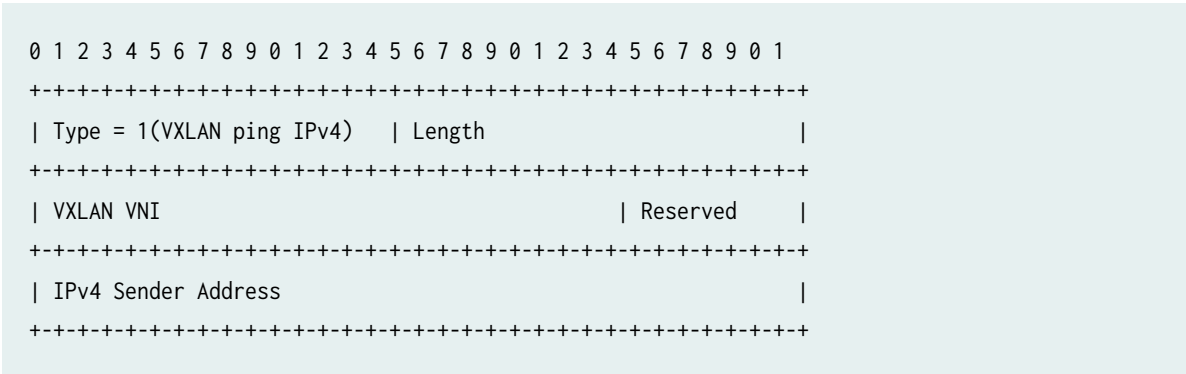
The OAM-specific message type is one of the following:

| Value | What it means |
|-------|---------------|
| 1     | Echo Request  |
| 2     | Echo Reply    |

Reply Mode Values:-

| Value | What it means                     |
|-------|-----------------------------------|
| 1     | Do not reply                      |
| 2     | Reply via an IPv4/IPv6 UDP Packet |
| 3     | Reply via Overlay Segment         |

The TLV definition for VXLAN ping is as follows:



Multiple Routing Instance Support

Starting in Junos OS Release 19.3R1, you can use the `ping overlay` and `traceroute overlay` commands to verify connectivity and detect fault in a static VxLAN tunnel with multiple routing instances. The ping and traceroute packets created for the `ping overlay` and `traceroute overlay` commands follow the same underlay network path as the data packets. This allow you to verify the connectivity between two VTEPs in the overlay VxLAN tunnel. The devices that are configured as the source and destination VTEP must both be running a Junos OS release that supports multiple routing instance, but the transit devices do not.

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                                                                                                                                                                                                                                                    |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19.3R1      | Starting in Junos OS Release 19.3R1, you can use the <code>ping overlay</code> and <code>traceroute overlay</code> commands to verify connectivity and detect fault in a static VxLAN tunnel with multiple routing instances.                                  |
| 14.1X53-D30 | Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, Release 16.1 for EX9200 switches, and Release 16.2 for MX Series routers, <code>overlay ping</code> and <code>traceroute</code> are introduced as troubleshooting tools for overlay networks. |

RELATED DOCUMENTATION

|                                                                                                                     |
|---------------------------------------------------------------------------------------------------------------------|
| <i>Example: Troubleshooting a VXLAN Overlay Network By Using Overlay Ping and Traceroute on QFX Series Switches</i> |
| <i>ping overlay</i>                                                                                                 |
| <i>traceroute overlay</i>                                                                                           |

Example: Troubleshoot a VXLAN Overlay Network with Overlay Ping and Overlay Traceroute on QFX Series Switches

IN THIS SECTION

● Requirements | 1399

- Overview and Topology | 1400
- Verification | 1403

In a Virtual Extensible LAN (VXLAN) overlay network, the existing ping and traceroute commands can verify the basic connectivity between two Juniper Networks devices that function as virtual tunnel endpoints (VTEPs) in the underlying physical network. However, in between the two VTEPs, there could be multiple routes through intermediary devices to the same destinations, and the ping and traceroute packets might successfully reach their destinations, while a connectivity issue exists in another route along which the data packets are typically forwarded.

With the introduction of the `overlay` parameter and other options in Junos OS Release 14.1X53-D30 for QFX5100 switches, you can use the ping and traceroute commands to troubleshoot a VXLAN overlay network.

For ping and traceroute mechanisms to work in a VXLAN overlay network, the ping and traceroute packets, also referred to as Operations, Administration, and Management (OAM) packets, must be encapsulated with the same VXLAN UDP headers (outer headers) as the data packets forwarded over the VXLAN segment with possible connectivity issues. If any connectivity issues arise, the overlay OAM packet would experience the same issues as the data packet.

This example shows how to use overlay ping and traceroute on a VTEP to verify the following in a VXLAN overlay network:

- Scenario 1—Verify that a particular VXLAN is configured on another VTEP.
- Scenario 2—Verify that the MAC address of a particular endpoint is associated with a VXLAN on another VTEP.
- Scenario 3—Verify that no issues exist in a particular data flow between sending and receiving endpoints.



**NOTE:** When issuing the `ping overlay` and `traceroute overlay` commands, the source VTEP on which you issue the command and the destination VTEP that receives the ping or traceroute packet must be Juniper Networks devices that support overlay ping and traceroute.

## Requirements

This example uses the following hardware and software components:

- Three physical (bare-metal) servers on which applications directly run.



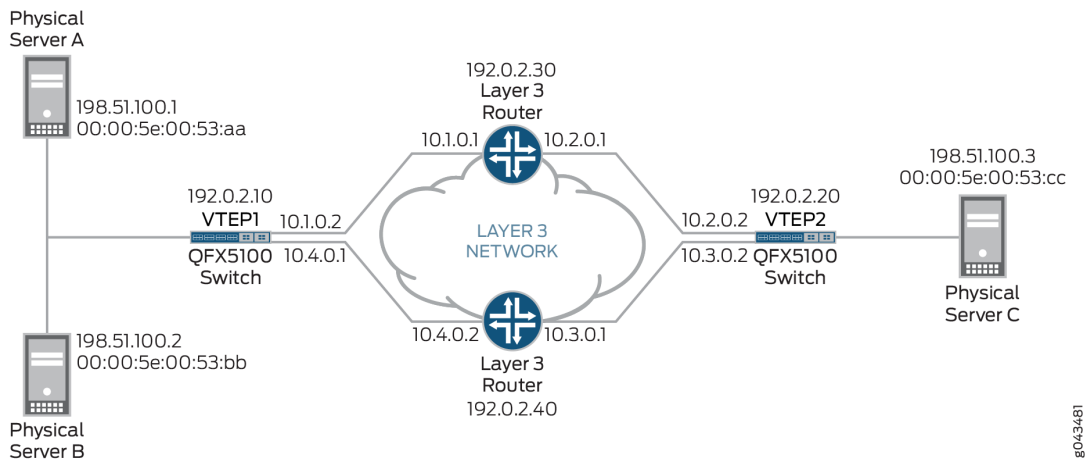
- Two QFX5100 switches running Junos OS Release 14.1X53-D30 or later software. These switches function as VTEPs.
- Two Layer 3 routers, which can be Juniper Networks routers or routers provided by another vendor.

Before issuing the `ping overlay` and `traceroute overlay` commands, gather the information needed for each parameter—for example, IP addresses or MAC addresses—used for a particular scenario. See [Table 82 on page 1401](#) to determine which parameters are used for each scenario.

## Overview and Topology

The VXLAN overlay network topology shown in [Figure 135 on page 1400](#) includes physical servers A, B, and C on which applications directly run. The applications on physical servers A and B need to communicate with the applications on physical server C. These servers are on the same subnet, so the communication between the applications occurs at the Layer 2 level, and VXLAN encapsulation or tunnels are used to transport their data packets over a Layer 3 network.

**Figure 135: Using Overlay Ping and Traceroute to Troubleshoot a VXLAN Overlay Network**



In this topology, there are two QFX5100 switches that function as VTEPs. VTEP1 initiates and terminates VXLAN tunnels for physical servers A and B, and VTEP2 does the same for physical server C. VTEP1 and VTEP2 are in VXLAN 100.

A data packet sent from physical server A is typically routed to the Layer 3 router with the IP address of 192.0.2.30 to reach physical server C.

In this VXLAN overlay network topology, a communication issue arises between physical servers A and C. To troubleshoot the issue with this data flow, you can initiate the `ping overlay` and `traceroute overlay` commands on VTEP1 (the source VTEP or `tunnel-src`) and specify that VTEP2 is the destination VTEP or `tunnel-dst`.

The ping overlay and traceroute overlay commands include several parameters. [Table 82 on page 1401](#) explains the purpose and provides a value for each of the parameters used in scenarios 1, 2, and 3.

[Table 82 on page 1401](#) does not include all available ping overlay and traceroute overlay parameters. This example uses the default values of these omitted parameters.

**Table 82: Ping and Traceroute Overlay Parameter Values For Scenarios 1, 2, and 3**

| ping overlay and traceroute overlay Parameters | Description                                                                                                                     | Scenario to Which Parameter Applies | Value             |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|-------------------|
| tunnel-type                                    | Identifies type of tunnel that you are troubleshooting.                                                                         | All                                 | vxlan             |
| vni                                            | VXLAN network identifier (VNI) of VXLAN used in this example.                                                                   | All                                 | 100               |
| tunnel-src                                     | IP address of VTEP1, on which you initiate overlay ping or traceroute.                                                          | All                                 | 192.0.2.10        |
| tunnel-dst                                     | IP address of VTEP2, which receives the overlay ping or traceroute packets.                                                     | All                                 | 192.0.2.20        |
| mac                                            | MAC address of physical server C, which is the destination endpoint.                                                            | Scenarios 2 and 3 only              | 00:00:5E:00:53:cc |
| count                                          | Number of overlay ping requests that VTEP1 sends.<br><br><b>NOTE:</b> The count parameter does not apply to overlay traceroute. | All                                 | 5                 |

**Table 82: Ping and Traceroute Overlay Parameter Values For Scenarios 1, 2, and 3 (Continued)**

| ping overlay and traceroute overlay Parameters | Description                                                                                                                                                                                                                     | Scenario to Which Parameter Applies | Value             |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|-------------------|
| hash-source-mac                                | MAC address of physical server A, which is the source endpoint.                                                                                                                                                                 | Scenario 3 only                     | 00:00:5E:00:53:aa |
| hash-destination-mac                           | <p>MAC address of physical server C, which is the destination endpoint.</p> <p><b>NOTE:</b> When specifying this parameter for scenario 3, the MAC address must be the same MAC address as specified for the mac parameter.</p> | Scenario 3 only                     | 00:00:5E:00:53:cc |
| hash-source-address                            | IP address of physical server A.                                                                                                                                                                                                | Scenario 3 only                     | 198.51.100.1      |
| hash-destination-address                       | IP address of physical server C.                                                                                                                                                                                                | Scenario 3 only                     | 198.51.100.3      |
| hash-vlan                                      | <p>VLAN ID of source endpoint.</p> <p><b>NOTE:</b> If the source endpoint is not a member of a VLAN, you do not need to use this parameter.</p>                                                                                 | Scenario 3 only                     | 150               |
| hash-input-interface                           | VTEP1 interface on which data flow originates.                                                                                                                                                                                  | Scenario 3 only                     | xe-0/0/2          |

**Table 82: Ping and Traceroute Overlay Parameter Values For Scenarios 1, 2, and 3 (Continued)**

| ping overlay and traceroute overlay Parameters | Description                                     | Scenario to Which Parameter Applies | Value |
|------------------------------------------------|-------------------------------------------------|-------------------------------------|-------|
| hash-protocol                                  | A value for the protocol used in the data flow. | Scenario 3 only                     | 17    |
| hash-source-port                               | A value for the outer TCP/UDP source port.      | Scenario 3 only                     | 4456  |
| hash-destination-port                          | A value for the outer UDP destination port.     | Scenario 3 only                     | 4540  |

Table 82 on page 1401 includes several hash parameters, which are used for scenario 3. For each of these parameters, you must specify a value associated with the data flow that you are troubleshooting. Based on the values that you specify, the system calculates a VXLAN UDP header source port hash, which is included in the VXLAN UDP header of the overlay ping and traceroute packets. Including the calculated hash in the VXLAN UDP header enables the overlay ping and traceroute packets to emulate data packets in the flow that you are troubleshooting.



**BEST PRACTICE:** When using the hash parameters, we recommend that you specify a value for each parameter. The exception to this guideline is the hash-vlan parameter, which you do not have to use if the source endpoint is not a member of a VLAN. This practice ensures that the overlay ping and traceroute processes are successful and that the output for each command is accurate. If you do not specify a value for one or more of the hash parameters, the system sends an OAM request that might include incorrect hash values and generates a warning message.

## Verification

### IN THIS SECTION

- Scenario 1: Verifying That VXLAN 100 Is Configured on VTEP2 | 1404
- Scenario 2: Verifying That the MAC Address of the Destination Endpoint Is on VTEP2 | 1407
- Scenario 3: Verifying a Data Flow | 1410

This section includes the following verification tasks:

### Scenario 1: Verifying That VXLAN 100 Is Configured on VTEP2

#### Purpose

Verify that a VXLAN with the VNI of 100 is configured on VTEP2. You can use either overlay ping or traceroute to perform this verification.

#### Action

##### Overlay Ping

On VTEP1, initiate an overlay ping:

```
user@switch> ping overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
count 5
ping-overlay protocol vxlan
```

```

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:00:00:00:00
count 5
ttl 255
```

```

WARNING: following hash-parameters are missing -
 hash computation may not succeed
```

```

end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host vlan
end-host input interface
end-host protocol
end-host l4-src-port
end-host l4-dst-port
```

```
Request for seq 1, to 192.0.2.20, at 09-24 22:03:16 PDT.033 msecs
```

```
Response for seq 1, from 192.0.2.20, at 09-24 22:03:16 PDT.036 msecs, rtt 10 msecs
```

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 2, to 192.0.2.20, at 09-24 22:03:16 PDT.044 msec

Response for seq 2, from 192.0.2.20, at 09-24 22:03:16 PDT.046 msec, rtt 10 msec

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 3, to 192.0.2.20, at 09-24 22:03:16 PDT.054 msec

Response for seq 3, from 192.0.2.20, at 09-24 22:03:16 PDT.057 msec, rtt 10 msec

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 4, to 192.0.2.20, at 09-24 22:03:16 PDT.065 msec

Response for seq 4, from 192.0.2.20, at 09-24 22:03:16 PDT.069 msec, rtt 10 msec

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 5, to 192.0.2.20, at 09-24 22:03:16 PDT.076 msec

Response for seq 5, from 192.0.2.20, at 09-24 22:03:16 PDT.079 msec, rtt 10 msec

Overlay-segment not present at RVTEP 192.0.2.20

## Overlay Traceroute

On VTEP1, initiate an overlay traceroute:

```
user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20
traceroute-overlay protocol vxlan

 vni 100
 tunnel src ip 192.0.2.10
 tunnel dst ip 192.0.2.20
 mac address 00:00:00:00:00:00
```

```
t1 255
```

```
WARNING: following hash-parameters are missing -
hash computation may not succeed
```

```
end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host vlan
end-host input interface
end-host protocol
end-host l4-src-port
end-host l4-dst-port
```

| t1 | Address    | Sender Timestamp             | Receiver Timestamp           | Response Time |
|----|------------|------------------------------|------------------------------|---------------|
| 1  | 10.1.0.1   | 09-25 00:51:10 PDT.599 msecs | *                            | 10 msecs      |
| 2  | 192.0.2.20 | 09-25 00:51:10 PDT.621 msecs | 09-25 00:51:10 PDT.635 msecs | 21 msecs      |

```
Overlay-segment not present at RVTEP 192.0.2.20
```

## Meaning

The sample overlay ping output indicates the following:

- VTEP1 sent five ping requests to VTEP2, and VTEP2 responded to each request.
- VTEP2 indicated that the VNI of 100 is not configured (Overlay-segment not present at RVTEP 192.0.2.20) and included this information in its response to VTEP1.

The sample overlay traceroute output indicates the following:

- Upon receiving an overlay traceroute packet with a time-to-live (TTL) value of 1 hop, the Layer 3 router responds to VTEP1.
- Upon receiving an overlay traceroute packet with a TTL value of 2 hops, VTEP2 responds to VTEP1.
- VTEP2 indicated that the VNI of 100 is not configured (Overlay-segment not present at RVTEP 192.0.2.20) and included this information in its response to VTEP1.



**NOTE:** The asterisk (\*) in the Receiver Timestamp column of the overlay traceroute output indicates that the Layer 3 router that received the overlay traceroute packet is

not a Juniper Networks device or is a Juniper Networks device that does not support overlay traceroute.

Given that the output of both overlay ping and traceroute indicates that VXLAN 100 is not present, check for this configuration on VTEP2. If you must configure a VNI of 100 on VTEP2, use the `vni` configuration statement at the `[edit vlans vlan-id vxlan]` hierarchy level, and reissue the `ping overlay` or `traceroute overlay` command to verify that VXLAN 100 is now recognized.

## Scenario 2: Verifying That the MAC Address of the Destination Endpoint Is on VTEP2

### Purpose

Verify that the MAC address (00:00:5E:00:53:cc) of physical server C, which is the destination endpoint, is in the forwarding table of VTEP2. You can use either overlay ping or traceroute to perform this verification.

### Action

#### Overlay Ping

On VTEP1, initiate an overlay ping:

```
user@switch> ping overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
mac 00:00:5E:00:53:cc count 5
ping-overlay protocol vxlan

 vni 100
 tunnel src ip 192.0.2.10
 tunnel dst ip 192.0.2.20
 mac address 00:00:5E:00:53:cc
 count 5
 ttl 255

 WARNING: following hash-parameters are missing -
 hash computation may not succeed

 end-host smac
 end-host dmac
 end-host src ip
 end-host dst ip
 end-host vlan
```



end-host input interface  
end-host protocol  
end-host l4-src-port  
end-host l4-dst-port

Request for seq 1, to 192.0.2.20, at 09-24 23:53:54 PDT.089 msec

Response for seq 1, from 192.0.2.20, at 09-24 23:53:54 PDT.089 msec, rtt 6 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 2, to 192.0.2.20, at 09-24 23:53:54 PDT.096 msec

Response for seq 2, from 192.0.2.20, at 09-24 23:53:54 PDT.100 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 3, to 192.0.2.20, at 09-24 23:53:54 PDT.107 msec

Response for seq 3, from 192.0.2.20, at 09-24 23:53:54 PDT.111 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 4, to 192.0.2.20, at 09-24 23:53:54 PDT.118 msec

Response for seq 4, from 192.0.2.20, at 09-24 23:53:54 PDT.122 msec, rtt 11 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 5, to 192.0.2.20, at 09-24 23:53:54 PDT.129 msec

Response for seq 5, from 192.0.2.20, at 09-24 23:53:54 PDT.133 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

## Overlay Traceroute

On VTEP1, initiate an overlay traceroute:

```
user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20 mac 00:00:5E:00:53:cc
traceroute-overlay protocol vxlan
```

```
vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
ttl 255
```

WARNING: following hash-parameters are missing -  
hash computation may not succeed

```
end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host vlan
end-host input interface
end-host protocol
end-host l4-src-port
end-host l4-dst-port
```

| ttl | Address    | Sender Timestamp             | Receiver Timestamp           | Response Time |
|-----|------------|------------------------------|------------------------------|---------------|
| 1   | 10.1.0.1   | 09-25 00:56:17 PDT.663 msecs | *                            | 10 msecs      |
| 2   | 192.0.2.20 | 09-25 00:56:17 PDT.684 msecs | 09-25 00:56:17 PDT.689 msecs | 11 msecs      |

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

## Meaning

The sample overlay ping output indicates the following:

- VTEP1 sent five ping requests to VTEP2, and VTEP2 responded to each request.
- VTEP2 verified that the VNI of 100 is configured (Overlay-segment present at RVTEP 192.0.2.20) but that the MAC address of physical server C is not in the forwarding table (End-System Not Present). VTEP2 included this information in its response to VTEP1.

The sample overlay traceroute output indicates the following:

- Upon receiving an overlay traceroute packet with a TTL value of 1 hop, the Layer 3 router responds to VTEP1.
- Upon receiving an overlay traceroute packet with a TTL value of 2 hops, VTEP2 responds to VTEP1.
- VTEP2 verified that the VNI of 100 is configured (Overlay-segment present at RVTEP 192.0.2.20) but that the MAC address of physical server C is not in the forwarding table (End-System Not Present). VTEP2 included this information in its response to VTEP1.



**NOTE:** The asterisk (\*) in the Receiver Timestamp column of the overlay traceroute output indicates that the Layer 3 router that received the overlay traceroute packet is not a Juniper Networks device or is a Juniper Networks device that does not support overlay traceroute.

Given that the output of both overlay ping and traceroute indicates that the MAC address of physical server C is not known by VTEP2, you must further investigate to determine why this MAC address is not in the forwarding table of VTEP2.

## Scenario 3: Verifying a Data Flow

### Purpose

Verify that there are no issues that might impede the flow of data from physical server A to physical server C. The networking devices that support this flow include VTEP1, the Layer 3 router with the IP address of 192.0.2.30, and VTEP2 (see [Figure 135 on page 1400](#)).

Initially, use overlay ping, and if the overlay ping results indicate an issue, then use overlay traceroute to determine in which segment of the path the issue exists.

With both overlay ping and traceroute, use the hash parameters to specify information about the devices in this data flow so that the system can calculate a VXLAN UDP header source port hash, which is included in the VXLAN UDP header of the overlay ping and traceroute packets. With the calculated

hash included in the VXLAN UDP header, the overlay ping and traceroute packets can emulate data packets in this flow, which should produce more accurate ping and traceroute results.



**BEST PRACTICE:** When using the hash parameters, we recommend specifying a value for each parameter. The exception to this guideline is the hash-vlan parameter, which you do not have to use if the source endpoint is not a member of a VLAN. This practice ensures that the overlay ping and traceroute processes are successful and that the output for each command is accurate. If you do not specify a value for one or more of the hash parameters, the system sends an OAM request that might include incorrect hash values and generates a warning message.

## Action

### Overlay Ping

On VTEP1, initiate an overlay ping:

```
user@switch> ping overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
mac 00:00:5E:00:53:cc count 5 hash-source-mac 00:00:5E:00:53:aa hash-destination-mac
00:00:5E:00:53:cc hash-source-address 198.51.100.1 hash-destination-address 198.51.100.3 hash-
vlan 150 hash-input-interface xe-0/0/2 hash-protocol 17 hash-source-port 4456 hash-destination-
port 4540
ping-overlay protocol vxlan
```

```
vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
count 5
ttl 255

hash-parameters:
 input-ifd-idx 653
 end-host smac 00:00:5E:00:53:aa
 end-host dmac 00:00:5E:00:53:cc
 end-host src ip 198.51.100.1
 end-host dst ip 198.51.100.3
 end-host protocol 17
 end-host l4-src-port 4456
 end-host l4-dst-port 4540
 end-host vlan 150
```

Request for seq 1, to 192.0.2.20, at 09-24 19:15:33 PDT.352 msecs

Request for seq 2, to 192.0.2.20, at 09-24 19:15:33 PDT.363 msecs

Request for seq 3, to 192.0.2.20, at 09-24 19:15:33 PDT.374 msecs

Request for seq 4, to 192.0.2.20, at 09-24 19:15:33 PDT.385 msecs

Request for seq 5, to 192.0.2.20, at 09-24 19:15:33 PDT.396 msecs

## Overlay Traceroute

If needed, on VTEP1, initiate an overlay traceroute:

```
user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20 mac 00:00:5E:00:53:cc hash-source-mac 00:00:5E:00:53:aa hash-destination-mac
00:00:5E:00:53:cc hash-source-address 198.51.100.1 hash-destination-address 198.51.100.3 hash-
vlan 150 hash-input-interface xe-0/0/2 hash-protocol 17 hash-source-port 4456 hash-destination-
port 4540
```

```
traceroute-overlay protocol vxlan
```

```
 vni 100
 tunnel src ip 192.0.2.10
 tunnel dst ip 192.0.2.20
 mac address 00:00:5E:00:53:cc
 ttl 255
```

```
 hash-parameters:
```

```
 input-ifd-idx 653
 end-host smac 00:00:5E:00:53:aa
 end-host dmac 00:00:5E:00:53:cc
 end-host src ip 198.51.100.1
 end-host dst ip 198.51.100.3
 end-host protocol 17
 end-host l4-src-port 4456
 end-host l4-dst-port 4540
 end-host vlan 150
```

| tth | Address  | Sender Timestamp             | Receiver Timestamp | Response Time |
|-----|----------|------------------------------|--------------------|---------------|
| 1   | 10.1.0.1 | 09-25 00:56:17 PDT.663 msecs | *                  | 10 msecs      |

## Meaning

The sample overlay ping output indicates that VTEP1 sent five ping requests to VTEP2, but VTEP2 did not respond to any of the requests. The lack of response from VTEP2 indicates that a connectivity issue exists along the path between VTEP1 and the Layer 3 router or the path between the Layer 3 router and VTEP2.

To further troubleshoot in which path the issue lies, overlay traceroute is used. The sample overlay traceroute output indicates the following:

- Upon receiving an overlay traceroute packet with a TTL value of 1 hop, the Layer 3 router responds to VTEP1, which indicates that the path between VTEP1 and the Layer 3 router is up.
- VTEP2 does not respond to the overlay traceroute packet, which indicates that the path between the Layer 3 router and VTEP2 might be down.



**NOTE:** The asterisk (\*) in the Receiver Timestamp column of the overlay traceroute output indicates that the Layer 3 router that received the overlay traceroute packet is not a Juniper Networks device or is a Juniper Networks device that does not support overlay traceroute.

Given that the overlay traceroute output indicates that there is a connectivity issue between the Layer 3 router and VTEP2, you must further investigate this path segment to determine the source of the issue.

## RELATED DOCUMENTATION

*Understanding Overlay ping and traceroute Packet Support*

*ping overlay*

*traceroute overlay*

## Example: Troubleshoot a VXLAN Overlay Network with Overlay Ping and Overlay Traceroute for MX Series Routers

### IN THIS SECTION

- [Requirements | 1414](#)
- [Overview and Topology | 1415](#)

- Configuration | 1418
- Verification | 1418

In a Virtual Extensible LAN (VXLAN) overlay network, the existing ping and traceroute commands can verify the basic connectivity between two Juniper Networks devices that function as virtual tunnel endpoints (VTEPs) in the underlying physical network. However, in between the two VTEPs, there could be multiple routes through intermediary devices, and the ping and traceroute packets might successfully reach their destinations, while a connectivity issue exists in another route along which the data packets are typically forwarded to reach their destination.

With the introduction of the overlay parameter and other options in Junos OS Release 16.2 for MX Series routers, you can use the ping and traceroute commands to troubleshoot a VXLAN.

For ping and traceroute mechanisms to work in a VXLAN, the ping and traceroute packets, also referred to as Operations, Administration, and Management (OAM) packets, must be encapsulated with the same VXLAN headers (outer headers) as the data packets forwarded over the VXLAN segment with possible connectivity issues. If any connectivity issues arise, the overlay OAM packet would experience the same issues as the data packet.

This example shows how to use overlay ping and traceroute on a VTEP to verify the following in a VXLAN:

- Scenario 1—Verify that a particular VXLAN is configured on another VTEP.
- Scenario 2—Verify that the MAC address of a particular endpoint is associated with a VXLAN on the remote VTEP.
- Scenario 3—Verify that no issues exist in a particular data flow between sending and receiving endpoints.



**NOTE:** When issuing the ping overlay and traceroute overlay commands, the source VTEP on which you issue the command and the destination VTEP that receives the ping packet must be Juniper Networks devices that support overlay ping and traceroute.

## Requirements

This example uses the following hardware and software components:

- Three physical servers on which applications directly run.
- Two MX Series routers running Junos OS Release 16.2 or later software. These routers function as VTEPs.

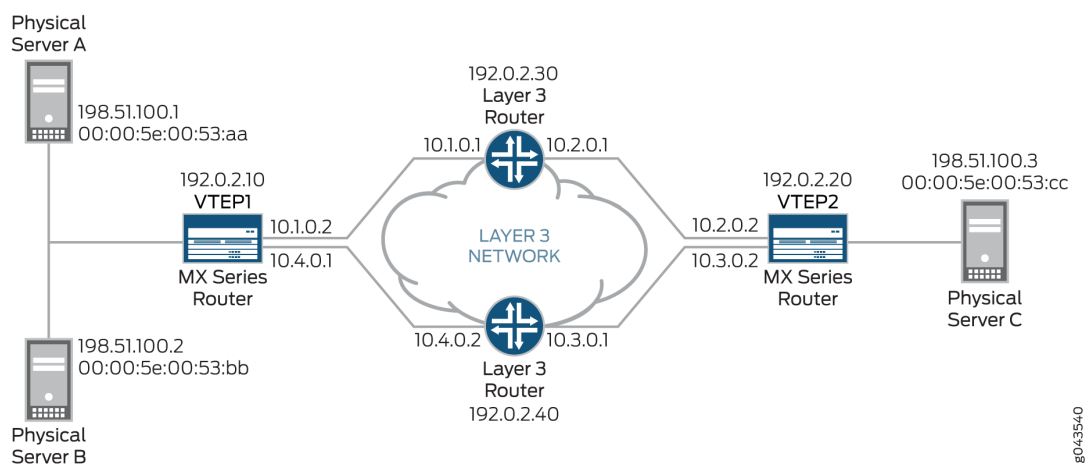
- Two Layer 3 routers, which can be Juniper Networks routers or routers provided by another vendor.

Before issuing the ping overlay and traceroute overlay commands, gather the information needed for each parameter— for example, IP addresses or MAC addresses— used for a particular scenario. See [Table 83 on page 1416](#) to determine which parameters are used for each scenario.

## Overview and Topology

The VXLAN topology shown in [Figure 136 on page 1415](#) includes physical servers A, B, and C on which applications directly run. The applications on physical servers A and B need to communicate with the applications on physical server C. These servers are on the same subnet, so the communication between the applications occurs at the Layer 2 level, and VXLAN encapsulation or tunnels are used to transport their data packets over a Layer 3 network.

**Figure 136: Using Overlay Ping and Traceroute to Troubleshoot a VXLAN**



In this topology, there are two MX Series routers that function as VTEPs. VTEP1 initiates and terminates VXLAN tunnels for physical servers A and B, and VTEP2 does the same for physical server C. VTEP1 and VTEP2 are in VXLAN 100.

A data packet sent from physical server A is typically routed to the Layer 3 router with the IP address of 192.0.2.30 to reach physical server C.

In this VXLAN topology, a communication issue arises between physical servers A and C. To troubleshoot the issue with this data flow, you can initiate the ping overlay and traceroute overlay commands on VTEP1 (the source VTEP or tunnel-src) and specify that VTEP2 is the destination VTEP or tunnel-dst.

The ping overlay and traceroute overlay commands include several parameters. [Table 83 on page 1416](#) explains the purpose and provides a value for each of the parameters used in the three scenarios.



Table 83 on page 1416 does not include all available ping overlay and traceroute overlay parameters. This example uses the default values of these omitted parameters.

**Table 83: Ping and Traceroute Overlay Parameter Values For Scenarios 1, 2, and 3**

| ping overlay and traceroute overlay Parameters | Description                                                                                                                            | Scenario to Which Parameter Applies | Value             |
|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|-------------------|
| tunnel-type                                    | Identifies type of tunnel that you are troubleshooting.                                                                                | All                                 | vxlan             |
| vni                                            | VXLAN network identifier (VNI) of VXLAN used in this example.                                                                          | All                                 | 100               |
| tunnel-src                                     | IP address of VTEP1, on which you initiate overlay ping or traceroute.                                                                 | All                                 | 192.0.2.10        |
| tunnel-dst                                     | IP address of VTEP2, which receives the overlay ping or traceroute packets.                                                            | All                                 | 192.0.2.20        |
| mac                                            | MAC address of physical server C, which is the destination endpoint.                                                                   | Scenarios 2 and 3                   | 00:00:5E:00:53:cc |
| count                                          | <p>Number of overlay ping requests that VTEP1 sends.</p> <p><b>NOTE:</b> The count parameter does not apply to overlay traceroute.</p> | All                                 | 5                 |

**Table 83: Ping and Traceroute Overlay Parameter Values For Scenarios 1, 2, and 3 (Continued)**

| ping overlay and traceroute overlay Parameters | Description                                                                                                                                                                                                              | Scenario to Which Parameter Applies | Value             |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|-------------------|
| hash-source-mac                                | MAC address of physical server A, which is the source endpoint.                                                                                                                                                          | Scenario 3                          | 00:00:5E:00:53:aa |
| hash-destination-mac                           | MAC address of physical server C, which is the destination endpoint.<br><br><b>NOTE:</b> When specifying this parameter for scenario 3, the MAC address must be the same MAC address as specified for the mac parameter. | Scenario 3                          | 00:00:5E:00:53:cc |
| hash-source-address                            | IP address of physical server A.                                                                                                                                                                                         | Scenario 3                          | 198.51.100.1      |
| hash-destination-address                       | IP address of physical server C.                                                                                                                                                                                         | Scenario 3                          | 198.51.100.3      |
| hash-protocol                                  | A value for the protocol used in the data flow.                                                                                                                                                                          | Scenario 3                          | 17                |
| hash-source-port                               | A value for the outer TCP/UDP source port.                                                                                                                                                                               | Scenario 3                          | 4456              |
| hash-destination-port                          | A value for the outer UDP destination port.                                                                                                                                                                              | Scenario 3                          | 4540              |

Table 83 on page 1416 includes several hash parameters, which are used for scenario 3. For each of these parameters, you must specify a value associated with the data flow that you are troubleshooting. Based on the values that you specify, the system calculates a VXLAN UDP header source port hash, which is included in the VXLAN UDP header of the overlay ping and traceroute packets. Including the

calculated hash in the VXLAN UDP header enables the overlay ping and traceroute packets to emulate data packets in the flow that you are troubleshooting.



**BEST PRACTICE:** When using the hash parameters, we recommend that you specify a value for each parameter. This practice ensures that the overlay ping and traceroute processes are successful and that the output for each command is accurate. If you do not specify a value for one or more of the hash parameters, the system sends an OAM request that might include incorrect hash values and generates a warning message.

## Configuration

## Verification

### IN THIS SECTION

- [Scenario-1: Verifying That VXLAN 100 Is Configured on VTEP2 | 1418](#)
- [Scenario 2: Verifying That the MAC Address of the Destination Endpoint Is on VTEP2 | 1421](#)
- [Scenario 3: Verifying a Data Flow | 1425](#)

This section includes the following verification tasks:

### Scenario-1: Verifying That VXLAN 100 Is Configured on VTEP2

#### Purpose

Verify that a VXLAN with the VNI of 100 is configured on VTEP2. You can use either overlay ping or traceroute to perform this verification.

#### Action

##### Overlay Ping

On VTEP1, initiate an overlay ping:

```
user@switch> ping overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
count 5
ping-overlay protocol vxlan
```

```

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:00:00:00:00
count 5
ttl 255

```

WARNING: following hash-parameters are missing -  
hash computation may not succeed

```

end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host protocol
end-host l4-src-port
end-host l4-dst-port

```

Request for seq 1, to 192.0.2.20, at 09-24 22:03:16 PDT.033 msecs

Response for seq 1, from 192.0.2.20, at 09-24 22:03:16 PDT.036 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 2, to 192.0.2.20, at 09-24 22:03:16 PDT.044 msecs

Response for seq 2, from 192.0.2.20, at 09-24 22:03:16 PDT.046 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 3, to 192.0.2.20, at 09-24 22:03:16 PDT.054 msecs

Response for seq 3, from 192.0.2.20, at 09-24 22:03:16 PDT.057 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 4, to 192.0.2.20, at 09-24 22:03:16 PDT.065 msecs

Response for seq 4, from 192.0.2.20, at 09-24 22:03:16 PDT.069 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 5, to 192.0.2.20, at 09-24 22:03:16 PDT.076 msecs

Response for seq 5, from 192.0.2.20, at 09-24 22:03:16 PDT.079 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

## Overlay Traceroute

On VTEP1, initiate an overlay traceroute:

```
user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20
```

```
traceroute-overlay protocol vxlan
```

```

 vni 100
 tunnel src ip 192.0.2.10
 tunnel dst ip 192.0.2.20
 mac address 00:00:00:00:00:00
 ttl 255
```

WARNING: following hash-parameters are missing -  
hash computation may not succeed

```

end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host protocol
end-host l4-src-port
end-host l4-dst-port
```

| ttl | Address    | Sender Timestamp             | Receiver Timestamp           | Response Time |
|-----|------------|------------------------------|------------------------------|---------------|
| 1   | 10.1.0.2   | 09-25 00:51:10 PDT.599 msecs | *                            | 10 msecs      |
| 2   | 192.0.2.20 | 09-25 00:51:10 PDT.621 msecs | 09-25 00:51:10 PDT.635 msecs | 21 msecs      |

Overlay-segment not present at RVTEP 192.0.2.20

## Meaning

The sample overlay ping output indicates the following:

- VTEP1 sent five ping requests to VTEP2, and VTEP2 responded to each request.



```
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
count 5
ttl 255
```

WARNING: following hash-parameters are missing -  
hash computation may not succeed

```
end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host protocol
end-host l4-src-port
end-host l4-dst-port
```

Request for seq 1, to 192.0.2.20, at 09-24 23:53:54 PDT.089 msec

Response for seq 1, from 192.0.2.20, at 09-24 23:53:54 PDT.089 msec, rtt 6 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 2, to 192.0.2.20, at 09-24 23:53:54 PDT.096 msec

Response for seq 2, from 192.0.2.20, at 09-24 23:53:54 PDT.100 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 3, to 192.0.2.20, at 09-24 23:53:54 PDT.107 msec

Response for seq 3, from 192.0.2.20, at 09-24 23:53:54 PDT.111 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 4, to 192.0.2.20, at 09-24 23:53:54 PDT.118 msec

Response for seq 4, from 192.0.2.20, at 09-24 23:53:54 PDT.122 msec, rtt 11 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 5, to 192.0.2.20, at 09-24 23:53:54 PDT.129 msec

Response for seq 5, from 192.0.2.20, at 09-24 23:53:54 PDT.133 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

## Overlay Traceroute

On VTEP1, initiate an overlay traceroute:

```
user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20 mac 00:00:5E:00:53:cc
traceroute-overlay protocol vxlan
```

```

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
ttl 255
```

WARNING: following hash-parameters are missing -  
hash computation may not succeed

```

end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host protocol
end-host l4-src-port
end-host l4-dst-port
```

| ttl | Address | Sender Timestamp | Receiver Timestamp | Response Time |
|-----|---------|------------------|--------------------|---------------|
|-----|---------|------------------|--------------------|---------------|



```

1 10.1.0.1 09-25 00:56:17 PDT.663 msecs * 10 msecs
2 192.0.2.20 09-25 00:56:17 PDT.684 msecs 09-25 00:56:17 PDT.689 msecs 11 msecs

```

Overlay-segment present at RVTEP 192.0.2.20

End-System not Present

## Meaning

The sample overlay ping output indicates the following:

- VTEP1 sent five ping requests to VTEP2, and VTEP2 responded to each request.
- VTEP2 verified that the VNI of 100 is configured (Overlay-segment present at RVTEP 192.0.2.20) but that the MAC address of physical server C is not in the forwarding table (End-System Not Present). VTEP2 included this information in its response to VTEP1.

The sample overlay traceroute output indicates the following:

- Upon receiving an overlay traceroute packet with a TTL value of 1 hop, the Layer 3 router responds to VTEP1.
- Upon receiving an overlay traceroute packet with a TTL value of 2 hops, VTEP2 responds to VTEP1.
- VTEP2 verified that the VNI of 100 is configured (Overlay-segment present at RVTEP 192.0.2.20) and that the MAC address of physical server C is in the forwarding table (End-System Present). VTEP2 included this information in its response to VTEP1.



**NOTE:** The asterisk (\*) in the Receiver Timestamp column of the overlay traceroute output indicates that the Layer 3 router that received the overlay traceroute packet is not a Juniper Networks device or is a Juniper Networks device that does not support overlay traceroute.

Given that the output of both overlay ping and traceroute indicates that the MAC address of physical server C is not known by VTEP2, you must further investigate to determine why this MAC address is not in the forwarding table of VTEP2.

### Scenario 3: Verifying a Data Flow

#### Purpose

Verify that there are no issues that might impede the flow of data from physical server A to physical server C. The networking devices that support this flow include VTEP1, the Layer 3 router with the IP address of 192.0.2.30, and VTEP2 (see [Figure 136 on page 1415](#)).

Initially, use overlay ping, and if the overlay ping results indicate an issue, use overlay traceroute to determine which device in the path has an issue.

With both overlay ping and traceroute, use the hash parameters to specify information about the devices in this data flow so that the system can calculate a VXLAN UDP header source port hash, which is included in the VXLAN UDP header of the overlay ping and traceroute packets. With the calculated hash included in the VXLAN UDP header, the overlay ping and traceroute packets can emulate data packets in this flow, which should produce more accurate ping and traceroute results.

#### Action

##### Overlay Ping

On VTEP1, initiate an overlay ping:

```
user@switch> ping overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
mac 00:00:5E:00:53:cc count 5 hash-source-mac 00:00:5E:00:53:aa hash-destination-mac
00:00:5E:00:53:cc hash-source-address 198.51.100.1 hash-destination-address 198.51.100.3 hash-
protocol 17 hash-source-port 4456 hash-destination-port 4540
ping-overlay protocol vxlan

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
count 5
ttl 255

hash-parameters:
 input-ifd-idx 653
 end-host smac 00:00:5E:00:53:aa
 end-host dmac 00:00:5E:00:53:cc
 end-host src ip 198.51.100.1
 end-host dst ip 198.51.100.3
 end-host protocol 17
```

```

end-host l4-src-port 4456
end-host l4-dst-port 4540end-host vlan 150
Request for seq 1, to 192.0.2.20, at 09-24 19:15:33 PDT.352 msec
Request for seq 2, to 192.0.2.20, at 09-24 19:15:33 PDT.363 msec
Request for seq 3, to 192.0.2.20, at 09-24 19:15:33 PDT.374 msec
Request for seq 4, to 192.0.2.20, at 09-24 19:15:33 PDT.385 msec
Request for seq 5, to 192.0.2.20, at 09-24 19:15:33 PDT.396 msec

```

## Overlay Traceroute

If needed, on VTEP1, initiate an overlay traceroute:

```

user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20 mac 00:00:5E:00:53:cc hash-source-mac 00:00:5E:00:53:aa hash-destination-mac
00:00:5E:00:53:cc hash-source-address 198.51.100.1 hash-destination-address 198.51.100.3 hash-
protocol 17 hash-source-port 4456 hash-destination-port 4540
traceroute-overlay protocol vxlan

```

```

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
ttl 255

hash-parameters:
input-ifd-idx 653
end-host smac 00:00:5E:00:53:aa
end-host dmac 00:00:5E:00:53:cc
end-host src ip 198.51.100.1
end-host dst ip 198.51.100.3
end-host protocol 17
end-host l4-src-port 4456
end-host l4-dst-port 4540

```

| ttl | Address  | Sender Timestamp            | Receiver Timestamp | Response Time |
|-----|----------|-----------------------------|--------------------|---------------|
| 1   | 10.1.0.1 | 09-25 00:56:17 PDT.663 msec | *                  | 10 msec       |

## Meaning

The sample overlay ping output indicates that VTEP1 sent five ping requests to VTEP2, but VTEP2 did not respond to any of the requests. The lack of response from VTEP2 indicates that a connectivity issue exists along the path between VTEP1 and the Layer 3 router or the path between the Layer 3 router and VTEP2.

To further troubleshoot in which path the issue lies, overlay traceroute is used. The sample overlay traceroute output indicates the following:

- Upon receiving an overlay traceroute packet with a TTL value of 1 hop, the Layer 3 router responds to VTEP1, which indicates that the path between VTEP1 and the Layer 3 router is up.
- VTEP2 does not respond to the overlay traceroute packet, which indicates that the path between the Layer 3 router and VTEP2 might be down.



**NOTE:** The asterisk (\*) in the Receiver Timestamp column of the overlay traceroute output indicates that the Layer 3 router that received the overlay traceroute packet is not a Juniper Networks device or is a Juniper Networks device that does not support overlay traceroute.

Given that the overlay traceroute output indicates that there is a connectivity issue between the Layer 3 router and VTEP2, you must further investigate this path segment to determine the source of the issue.

## RELATED DOCUMENTATION

[Understanding Overlay ping and traceroute Packet Support | 1394](#)

*ping overlay*

*traceroute overlay*

# 3

PART

## EVPN-MPLS

---

- Overview | **1429**
  - Convergence in an EVPN MPLS Network | **1444**
  - Pseudowire Termination at an EVPN | **1446**
  - Configuring the Distribution of Routes | **1453**
  - Configuring VLAN Services and Virtual Switch Support | **1465**
  - Configuring Integrated Bridging and Routing | **1499**
  - Configuring IGMP or MLD Snooping with EVPN-MPLS | **1565**
-

# Overview

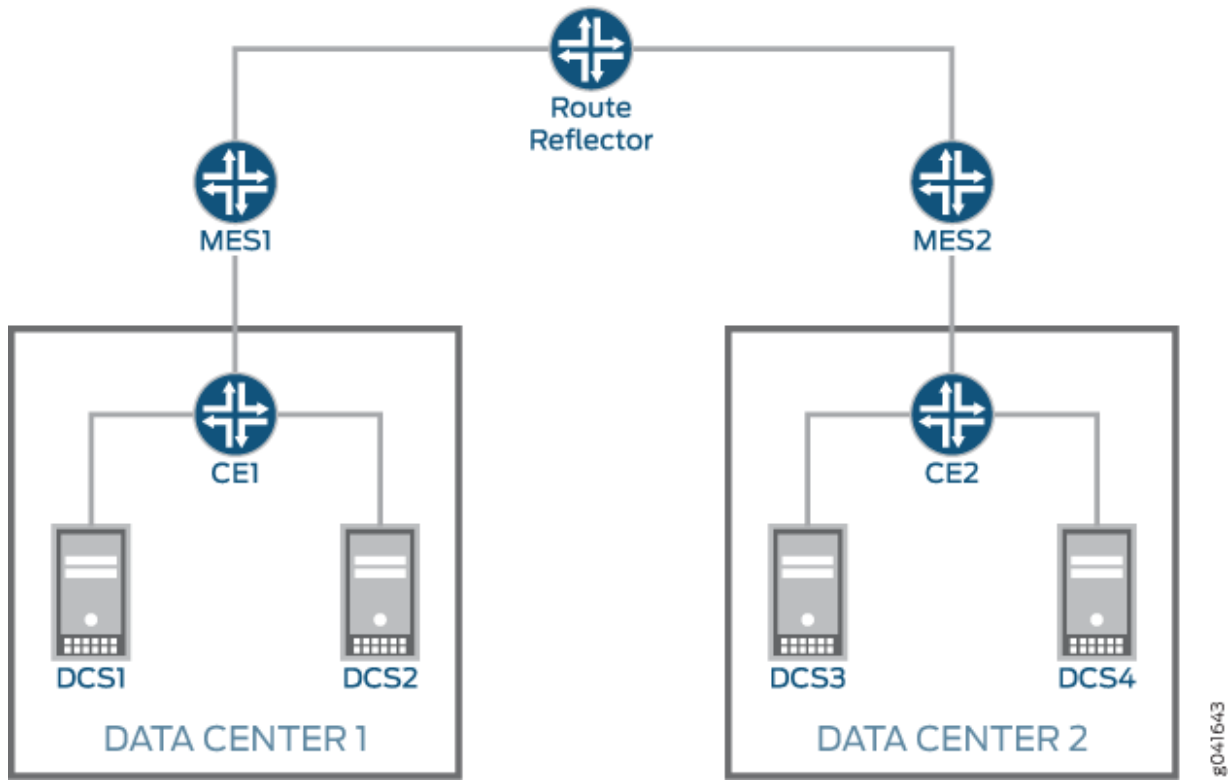
## IN THIS CHAPTER

- [EVPN Overview | 1429](#)
- [EVPN-MPLS Caveats | 1431](#)
- [EVPN Overview for Switches | 1432](#)
- [Migrating from FEC128 LDP-VPLS to EVPN Overview | 1434](#)

## EVPN Overview

An Ethernet VPN (EVPN) enables you to connect dispersed customer sites using a Layer 2 virtual bridge. As with other types of VPNs, an EVPN consists of customer edge (CE) devices (host, router, or switch) connected to provider edge (PE) routers. The PE routers can include an MPLS edge switch (MES) that acts at the edge of the MPLS infrastructure. Either an MX Series 5G Universal Routing Platform or a standalone switch can be configured to act as an MES. You can deploy multiple EVPNs within a service provider network, each providing network connectivity to a customer while ensuring that the traffic sharing on that network remains private. [Figure 137 on page 1430](#) illustrates a typical EVPN deployment. Traffic from Data Center 1 is transported over the service provider's network through MES1 to MES2 and then onto Data Center 2. DCS1, DCS2, DCS3, and DCS4 are the data center switches.

Figure 137: EVPN Connecting Data Center 1 and Data Center 2



The MESs are interconnected within the service provider's network using label-switched paths (LSPs). The MPLS infrastructure allows you to take advantage of the MPLS functionality provided by the Junos OS, including fast reroute, node and link protection, and standby secondary paths. For EVPNs, learning between MESs takes place in the control plane rather than in the data plane (as is the case with traditional network bridging). The control plane provides greater control over the learning process, allowing you to restrict which devices discover information about the network. You can also apply policies on the MESs, allowing you to carefully control how network information is distributed and processed. EVPNs utilize the BGP control plane infrastructure, providing greater scale and the ability to isolate groups of devices (hosts, servers, virtual machines, and so on) from each other.

The MESs attach an MPLS label to each MAC address learned from the CE devices. This label and MAC address combination is advertised to the other MESs in the control plane. Control plane learning enables load balancing and improves convergence times in the event of certain types of network failures. The learning process between the MESs and the CE devices is completed using the method best suited to each CE device (data plane learning, IEEE 802.1, LLDP, 802.1aq, and so on).

The policy attributes of an EVPN are similar to an IP VPN (for example, Layer 3 VPNs). Each EVPN routing instance requires that you configure a route distinguisher (RD) and one or more route targets (RTs). A CE device attaches to an EVPN routing instance on an MES through an Ethernet interface that might be configured for one or more VLANs.

See [EVPN over MPLS](#) in [Feature Explorer](#) for details on the features and sub-features we support with EVPN-MPLS on various platforms.

### 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    | Ethernet VPN (EVPN) support, including EVPN-MPLS, EVPN + VXLAN, and PBB EVPN, has been extended to logical systems running only on MX devices. The same EVPN options and performance that are available in the default EVPN instance are available in a logical system. Configure EVPN on a logical system under the [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> protocols evpn] hierarchy. |

### RELATED DOCUMENTATION

[Supported EVPN Standards | 2016](#)

[EVPN Multihoming Overview | 162](#)

[Overview of MAC Mobility | 11](#)

## EVPN-MPLS Caveats

### IN THIS SECTION

- [Caveats for ACX7100-32C and ACX7100-48L Devices | 1431](#)

### Caveats for ACX7100-32C and ACX7100-48L Devices

The following EVPN-MPLS features are not supported on ACX7100-32C and ACX7100-48L devices running Junos OS Evolved Release 21.3R1.

- VLAN-aware bundle service
- MAC address pinning
- Access and trunk interfaces



- ERPS over EVPN
- EVPN, virtual-switch, and default-switch instances types
- EVPN with P2MP (point-to-multipoint) LSPs (link-state protocols)
- PBB-EVPN
- EVPN NSR

Also, If there are multiple paths from an ingress EVPN PE (provider edge) to any remote EVPN PE, then load balancing FRR (Fast Reroute) over those paths is not supported on BUM traffic.

Check [Feature Explorer](#) to determine if the features listed here are supported in another release.

## EVPN Overview for Switches

An Ethernet VPN (EVPN) enables you to connect a group of dispersed customer sites using a Layer 2 virtual bridge. As with other types of VPNs, an EVPN comprises customer edge (CE) devices (host, router, or switch) connected to provider edge (PE) devices. The PE devices can include an MPLS edge switch (MES) that acts at the edge of the MPLS infrastructure. For the initial deployment of EVPNs using Juniper Networks equipment, you can configure an EX9200 switch to act as an MES. You can deploy multiple EVPNs within the network, each providing network connectivity to customers while ensuring that traffic sharing that network remains private.

The MESs are interconnected within the network by using label-switched paths (LSPs). The MPLS infrastructure allows you to take advantage of the MPLS functionality provided by the Junos operating system (Junos OS), including fast reroute, node and link protection, and standby secondary paths. For EVPNs, learning between MESs takes place in the control plane rather than in the data plane (as is the case with traditional network bridging). The control plane provides greater control over the learning process, allowing you to restrict which devices discover information about the network. You can also apply policies on the MESs, allowing you to carefully control how network information is distributed and processed. EVPNs utilize the BGP control plane infrastructure, providing greater scale and the ability to isolate groups of devices (hosts, servers, virtual machines, and so on) from each other.

The MESs attach an MPLS label to each MAC address learned from the CE devices. This label and MAC address combination is advertised to the other MESs in the control plane. Control plane learning enables load balancing and improves convergence times in the event of certain types of network failures. The learning process between the MESs and the CE devices is completed using the method best suited to each CE device (data plane learning, IEEE 802.1, LLDP, 802.1aq, and so on).

The policy attributes of an EVPN are similar to an IP VPN (for example, Layer 3 VPNs). Each EVPN routing instance requires that you configure a route distinguisher and one or more route targets. A CE

device attaches to an EVPN routing instance on an MES through an Ethernet interface that might be configured for one or more VLANs.

The following features are available for EVPNs:

- Ethernet connectivity between data centers spanning metropolitan area networks (*MANs*) and *WANs*
- One or more VLANs for each MAC VPN
- Automatic route distinguishers
- Dual-homed EVPN connection with active standby multihoming
- Starting with Junos OS Releases 16.1R4 and 16.2R2, the active-active mode for EVPN multihoming is supported.
- Starting with Junos OS Release 17.3R1, both pure type-5 routes and standard type-5 routes are supported on EX9200 switches. Use this feature, which advertises IP prefixes through EVPN, when the Layer 2 domain does not exist at the remote data centers or metro network peering points. For more information about how to configure, see *ip-prefix-routes*.
- Starting with Junos OS OS Release 17.3R1, VXLAN encapsulation is supported. Previously, only MPLS encapsulation is supported.
- Starting in Junos OS Evolved Release 22.4R1, you can configure nonstop active routing (NSR).

Support doesn't include Graceful restart (GRES) or VLAN-aware service.

The following features are not supported for EVPNs:

- Graceful restart and graceful Routing Engine switchover (GRES).

### 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                                                                                                                   |
|---------|-------------------------------------------------------------------------------------------------------------------------------|
| 17.3R1  | Starting with Junos OS Release 17.3R1, both pure type-5 routes and standard type-5 routes are supported on EX9200 switches.   |
| 17.3R1  | Starting with Junos OS OS Release 17.3R1, VXLAN encapsulation is supported. Previously, only MPLS encapsulation is supported. |
| 16.1R4  | Starting with Junos OS Releases 16.1R4 and 16.2R2, the active-active mode for EVPN multihoming is supported.                  |

## RELATED DOCUMENTATION

[Supported EVPN Standards | 2016](#)

[Example: Configuring EVPN Active-Active Multihoming | 375](#)

## Migrating from FEC128 LDP-VPLS to EVPN Overview

### IN THIS SECTION

- [Technology Overview and Benefits | 1434](#)
- [FEC128 LDP-VPLS to EVPN Migration | 1435](#)
- [Sample Configuration for LDP-VPLS to EVPN Migration | 1437](#)
- [Reverting to VPLS | 1441](#)
- [LDP-VPLS to EVPN Migration and Other Features | 1441](#)

For service providers with virtual private LAN service (VPLS) networks and Ethernet VPN (EVPN) networks, there is a need to interconnect these networks. Before the introduction of the Seamless Migration from LDP-VPLS to EVPN feature, a logical tunnel interface on the interconnection point of the VPLS and EVPN routing instances was used for this purpose. In this case, the provider edge (PE) devices in each network were unaware of the PE devices in the other technology network.

With the introduction of the Seamless Migration from LDP-VPLS to EVPN feature, a solution is introduced for enabling staged migration from FEC128 LDP-VPLS toward EVPN on a site-by-site basis for every VPN routing instance. In this solution, the PE devices running EVPN and VPLS for the same VPN routing instance and single-homed segments can coexist. During migration, there is minimal impact to the customer edge (CE) device-to-CE device traffic forwarding for affected customers.

See the [Seamless Migration from LDP-VPLS to EVPN](#) entry in [Feature Explorer](#) for information regarding platform and Junos OS support.

The following sections describe the migration from LDP-VPLS to EVPN:

### Technology Overview and Benefits

Virtual private LAN service (VPLS) is an Ethernet-based point-to-multipoint Layer 2 VPN. This technology allows you to connect geographically dispersed data center LANs to each other across an MPLS backbone while maintaining Layer 2 connectivity. The high availability features defined in VPLS

standards (such as LER dual homing) and topology autodiscovery features using BGP signaling make VPLS scalable and easy to deploy. Because VPLS uses MPLS as its core, it provides low latency variation and statistically bound low convergence times within the MPLS network.

Ethernet VPN (EVPN), on the other hand, is a combined Layer 2 and Layer 3 VPN solution that is more scalable, resilient, and efficient than current technologies. It provides several benefits including greater network efficiency, reliability, scalability, virtual machine (VM) mobility, and policy control for service providers and enterprises.

Although VPLS is a widely deployed Layer 2 VPN technology, service provider networks migrate to EVPN because of the scaling benefits and ease of deployment. Some of the benefits of EVPN include:

- Control plane traffic is distributed with BGP and the broadcast and multicast traffic is sent using a shared multicast tree or with ingress replication.
- Control plane learning is used for MAC and IP addresses instead of data plane learning. MAC address learning requires the flooding of unknown unicast and ARP frames, whereas IP address learning does not require any flooding.
- Route reflector is used to reduce a full mesh of BGP sessions among PE devices to a single BGP session between a PE device and the route reflector.
- Autodiscovery with BGP is used to discover PE devices participating in a given VPN, PE devices participating in a given redundancy group, tunnel encapsulation types, multicast tunnel type, and multicast members.
- All-active multihoming is used. This allows a given CE device to have multiple links to multiple PE devices, and traffic traversing to-and-from that CE device fully utilizes all of these links (Ethernet segment).
- When a link between a CE device and a PE device fails, the PE devices for that EVPN instance (EVI) are notified of the failure with the withdrawal of a single EVPN route. This allows those PE devices to remove the withdrawing PE device as a next hop for every MAC address associated with the failed link (mass withdrawal).

## **FEC128 LDP-VPLS to EVPN Migration**

Some service providers want to preserve their investments in VPLS. This leads to the need to connect the old VPLS networks to new networks that run EVPN. For this purpose, logical tunnel interfaces on the interconnection point of the VPLS and EVPN routing instances were used. However, all the other PE devices belonged either to the VPLS network or to the EVPN network and were unaware of the other technology.

Starting in Junos OS Release 17.3, EVPN can be introduced into an existing VPLS network in a staged manner, with minimal impact to VPLS services. On a VPLS PE device, some customers can be moved to EVPN, while other customers continue to use VPLS pseudowires. Other PE devices can be entirely VPLS

and switching customers on other PE devices to EVPN. This solution provides support for the seamless migration Internet draft (expires January ,2018), *(PBB-)EVPN Seamless Integration with (PBB-)VPLS*.

The seamless migration from FEC128 LDP-VPLS to EVPN solution supports the following functionality:

- Allow for staged migration toward EVPN on a site-by-site basis per VPN instance. For instance, new EVPN sites to be provisioned on EVPN PE devices.
- Allow for the coexistence of PE devices running both EVPN and VPLS for the same VPN instance and single-homed segments.

In the LDP-VPLS to EVPN migration, the PE device where some customers have been migrated to EVPN while other customers are being served using VPLS is called a super PE device. As super PE devices discover other super PE devices within a routing instance, they use EVPN forwarding to communicate with other super PE devices and VPLS pseudowires to PE devices running VPLS. The PE device with no EVPN awareness, and running only VPLS for all the customers, is called a VPLS PE device.

The CE device connected to a super PE can reach CE devices connected to EVPN-only PE devices or VPLS-only PE devices, but CE devices connected to EVPN-only PE devices cannot reach CE devices connected to VPLS-only PE devices.

Because the migration from LDP-VPLS to EVPN is supported on a per-routing instance basis, and if the routing instance is serving multiple customers on a PE device, all are migrated together. EVPN is responsible for setting up data forwarding between the PE devices upgraded to EVPN, while VPLS continues to set up data forwarding to PE devices that run VPLS. There should be zero impact for customers that still use VPLS pseudowire on all the PE devices.



**NOTE:** The following features are not supported with the LDP-VPLS to EVPN migration:

- Migration from FEC129 VPLS to EVPN.
- Migration from BGP-VPLS to EVPN.
- Migration of VPLS virtual switch to EVPN virtual switch.
- Migration of VPLS routing instance to EVPN virtual switch.
- Migration of VPLS routing instance or PBB-VPLS to PBB-EVPN.
- Seamless migration from EVPN back to VPLS.
- Enhancing EVPN to support the set of tools or statements and commands that VPLS supports.

- Active-active and active-standby multihoming. The migration to EVPN is supported only on single-homed deployments.
- Spanning all-active across EVPN and VPLS PE devices does not work, because the all-active multihoming feature is not supported on VPLS.
- Connecting EVPN-only PE devices with VPLS-only PE devices through super PE devices.
- IPv6, logical systems, multichassis support, and SNMP, because they are currently not supported on EVPN.

## Sample Configuration for LDP-VPLS to EVPN Migration

The following sections provide the sample configuration required for performing the LDP-VPLS to EVPN migration.

### LDP-VPLS Configuration

A typical static LDP-VPLS routing instance configuration is as follows:

```
user@host# show routing-instance foo
instance-type vpls;
vlan-id 100; (not needed for VLAN bundle service)
interface ge-2/0/0.590;
interface ae500.590;
routing-interface irb.0;
forwarding-options {
 family vpls {
 filter {
 input UNKNOWN-UNICAST;
 }
 }
}
protocols {
 vpls {
 control-word;
 encapsulation-type ethernet-vlan;
 enable-mac-move-action;
 mac-table-size {
 100000;
 }
 }
}
```

```

 packet-action drop;
 }
 mac-table-aging-time ;
 interface-mac-limit {
 100000;
 packet-action drop;
 }
 no-tunnel-services; (use label-switched interfaces)
 vpls-id 245015;
 mtu 1552;
 ignore-mtu-mismatch;
 mac-flush {
 any-spoke;
 }
 no-vlan-id-validate;
 neighbor 192.168.252.64 {
 psn-tunnel-endpoint 10.0.0.31;
 pseudowire-status-tlv;
 revert-time 60;
 backup-neighbor 192.168.252.65 {
 psn-tunnel-endpoint 10.0.0.32;
 hot-standby;
 }
 }
}
mesh-group Spoke { (access label-switched interface toward spoke)
local-switching;
neighbor 192.168.252.66 {
 psn-tunnel-endpoint 10.0.0.41;
 pseudowire-status-tlv;
}
neighbor 192.168.252.67 {
 psn-tunnel-endpoint 10.0.0.42;
 pseudowire-status-tlv;
}
}
connectivity-type permanent;
}

```

```

user@host# show interfaces ge-2/0/0.590
encapsulation vlan-vpls;
vlan-id 590;

```

```

output-vlan-map {
 swap;
 tag-protocol-id 0x8100;
 inner-vlan-id 590;
}
family vpls {
 filter {
 input-list [listA];
 output-list listB;
 }
}

```

## EVPN Migration Configuration

To perform the FEC128 LDP-VPLS to EVPN migration, do the following:

1. On the backup Routing Engine, load Junos OS Release 17.3R1.
2. Perform in-service software upgrade (ISSU) to acquire primary role. Ensure that the VPLS unified ISSU does not have any impact on the VPLS forwarding.
3. Identify routing instances (customers) that need to be migrated to EVPN.
4. Enable EVPN in a single routing instance.
  - Change routing instance type to `evpn`, and include the `evpn` statement at the `[edit routing-instances routing-instance-name protocols]` hierarchy level, and also include the `vpls` statement at the same hierarchy to support VPLS commands.

For example:

```

[edit routing-instances routing-instance-name]
instance-type evpn;
interface ge-2/0/0.590;
interface ae500.590;
routing-interface irb.0;
route-distinguisher 10.1.1.1:50; (add for LDP-VPLS)
vrf-target target:100:100; (add for LDP-VPLS)
forwarding-options {
 family vpls {
 filter {
 input UNKNOWN-UNICAST;
 }
 }
}

```



```

 }
}
protocols {
 vpls { (supports all existing VPLS commands)
 }
}

```

## 5. Enable family EVPN signaling in BGP.

For example:

```

protocols {
 bgp {
 local-as 64512;

 group 2mx {
 type internal;
 local-address 10.81.1.1;

 family evpn {
 signaling;
 }
 neighbor 10.81.2.2;
 neighbor 10.81.9.9;
 }
 }
}

```

After the configuration for the EVPN migration is committed, the routing protocol process and the Layer 2 address learning process start building the EVPN state to reflect interfaces, bridge domains, peers and routes. The locally learnt MAC addresses are synchronized by the Layer 2 address learning process in the instance.vpls.0 to the routing protocol process. When a local MAC ages out in the instance.vpls.0, the routing protocol process is informed by the Layer 2 address learning process.

When an EVPN peer is learnt, the routing protocol process sends a new message to the Layer 2 address learning process to remove the peer's label-switched interface or virtual tunnel logical interface from the VE mesh group and disables MAC-learning on it. The EVPN IM next-hop is then added to the VE mesh group. The EVPN behavior in the routing protocol process of learning MAC addresses over BGP and informing Layer 2 address learning process of the MPLS next hop is maintained.

The VPLS statements and commands continue to apply to the VPLS pseudowires between the PE devices and the MAC addresses learnt over them. The EVPN statements and commands apply to PE devices running EVPN.

## Reverting to VPLS

If the EVPN migration runs into issues, you can revert back to VPLS until the issue is understood. The routing instance is reverted from a super PE to a VPLS PE in a non-catastrophic manner by enabling the following configuration:

```
[edit routing-instances routing-instance-name]
user@host# set instance-type vpls
user@host# delete protocols evpn
user@host# delete route-distinguisher (if running LDP-VPLS)
user@host# delete vrf-target (if running LDP-VPLS)
```

On reverting the EVPN migration to VPLS, the following happens:

1. The EVPN state information is deleted.
2. There is a trigger for withdrawal of EVPN control plane routes.
3. The routing protocol process sends a new message to the Layer 2 address learning process with the label-switched interface or the virtual tunnel logical interface for the routing instance and peer.
4. The label-switched or virtual tunnel interface adds the new message to the flood group and MAC learning is enabled.
5. The egress IM next hop is deleted by the routing protocols process, prompting the Layer 2 address learning process to remove it from the flood group.
6. Remote MAC addresses are learned again over the label-switched interface or virtual tunnel logical interface.

## LDP-VPLS to EVPN Migration and Other Features

[Table 84 on page 1442](#) describes the functionality of some of the related features, such as multihoming and integrated routing and bridging (IRB) with the LDP-VPLS to EVPN migration.

**Table 84: EVPN Migration and Other Features Support**

| Feature  | Supported Functionality in EVPN Migration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MAC move | <p>MAC moves are supported between VPLS-only PE device and super PE devices.</p> <p>When a MAC address moves from a VPLS-only PE device to a super PE device, it is learned over BGP, and the routing protocol process informs the Layer 2 address learning process of the EVPN next hop to be updated in the foo.vpls.0 routing table.</p> <p>When a MAC address moves from a super PE device to a VPLS-only PE device, it is learned in the Packet Forwarding Engine on the label-switched interface or virtual tunnel interface. The Layer 3 address learning process updates it to VPLS or the label-switched interface next hop.</p> <p>When the type 2 route is withdrawn by EVPN BGP, the MAC address is not deleted from the forwarding table, so there is no loss of data.</p> <p>The forwarding MAC table is shared by VPLS and EVPN. Some attributes, such as <code>mac-table-size</code> and <code>mac-table-aging-time</code> could be configured under both EVPN and VPLS. When there is a conflict, the values under EVPN take precedence.</p> |
| IRB      | <p>No changes needed in IRB.</p> <p>On a super PE device, EVPN populates the /32 host routes learned over MAC+IP type 2 routes from EVPN peers in a Layer 3 virtual routing and forwarding, while VPLS IRB forwarding using subnet routes works on sites still running VPLS.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

**Table 84: EVPN Migration and Other Features Support *(Continued)***

| Feature           | Supported Functionality in EVPN Migration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Hierarchical VPLS | <p>In an H-VPLS network with hub-and-spoke PE devices, when the hub PE device is migrated to EVPN, local MAC addresses learned over the access label-switched or virtual tunnel interface need to be advertised to BGP, so that the other EVPN-only PE devices or super PE devices can reach them.</p> <p>Take the following into consideration when migrating an H-VPLS network to EVPN:</p> <ul style="list-style-type: none"> <li>• Hubs typically have local switching enabled as interspoke traffic is forwarded through the hub. If spokes alone are migrated to EVPN and spokes have Layer 3 or MPLS reachability to each other, the label-switched or virtual tunnel interface to the hub and EVPN next hop (remote spoke) is present in the VPLS edge (VE) floodgroup. This results in two copies of broadcast, unknown unicast, and multicast (BUM) traffic received by the remote spoke. One option to avoid this behavior is to migrate the hubs to EVPN too.</li> <li>• EVPN is not aware of hierarchy. All peers are considered core-facing. Once hubs and spokes are migrated to EVPN, split horizon prevents the BUM traffic from being forwarded to other core-facing PE devices.</li> </ul> |
| ESI configuration | Ethernet segment identifier (ESI) is configured at the physical interface or port level.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

**RELATED DOCUMENTATION**
[EVPN Overview](#) | 1429

# Convergence in an EVPN MPLS Network

## IN THIS CHAPTER

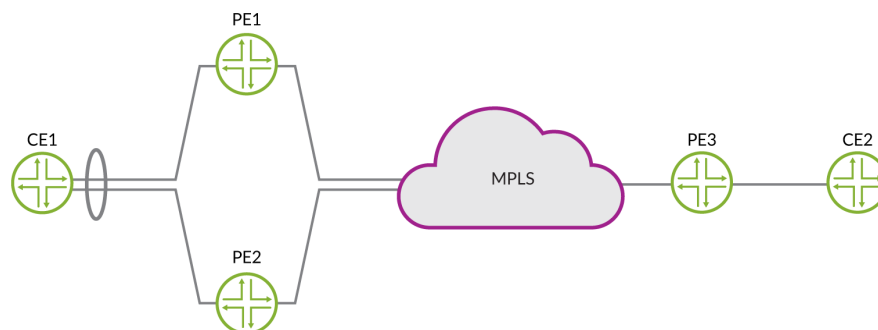
- [Convergence in a Multihomed EVPN-MPLS Network | 1444](#)

## Convergence in a Multihomed EVPN-MPLS Network

Multihoming a CE device to two or more PE devices on an EVPN MPLS network provides redundant connectivity and allows the network to continue providing services when one of the PE devices fail or if there is a PE-CE link failure. When the multihomed connection from a CE device to a PE device fails, the devices in the networks updates the EVPN routes, including any MAC routes and next hop entries in their forwarding table

[Figure 138 on page 1444](#) shows a simple EVPN topology with CE1 multihomed to PE1 and PE2 with an Ethernet segment. If the connection from CE1 to PE1 fails and similarly when the connection from CE1 to PE1 is restored, PE1 will withdraw MAC routes from PE2 and PE3, while PE2 and PE3 will remove the MAC routes that they learned from PE1. EVPN egress link protection adds backup next hops on the multihomed PE devices and is used to support fast reroute (FRR) in the network. When you enable EVPN egress link protection, you can improve the convergence time in the EVPN MPLS network and reduce traffic loss when the link to a PE device fails.

**Figure 138: Simple EVPN MPLS Topology**



To enable EVPN egress link protection, include `evpn-egress-link-protection` at the `[edit routing-options forwarding-table]` hierarchy level on all multihomed PE devices in the EVPN network.



**NOTE:** In addition, you must also include `dynamic-list-next-hop` at the `[edit routing-options forwarding-table]` hierarchy level on the single-homed PE device (PE3).

The following example shows a sample configuration for evpn egress link protection.

```
routing-instances {
 blue {
 instance-type evpn;
 vlan-id 100;
 interface ae0.0;
 route-distinguisher 10.255.255.1:100;
 vrf-target target:100:100;
 protocols evpn;
 }
}
routing-options {
 forwarding-table {
 export lb;
 evpn-egress-link-protection;
 }
}
policy-options {
 policy-statement lb {
 term 1 {
 then {
 load-balance per-packet;
 }
 }
 }
}
```

## RELATED DOCUMENTATION

[Configuring Dynamic List Next Hop | 510](#)

*forwarding-table*

# Pseudowire Termination at an EVPN

## IN THIS CHAPTER

- [Overview of Pseudowire Termination at an EVPN | 1446](#)
- [Configuring Pseudowire Termination | 1447](#)
- [Support for Redundant Logical Tunnel | 1451](#)

## Overview of Pseudowire Termination at an EVPN

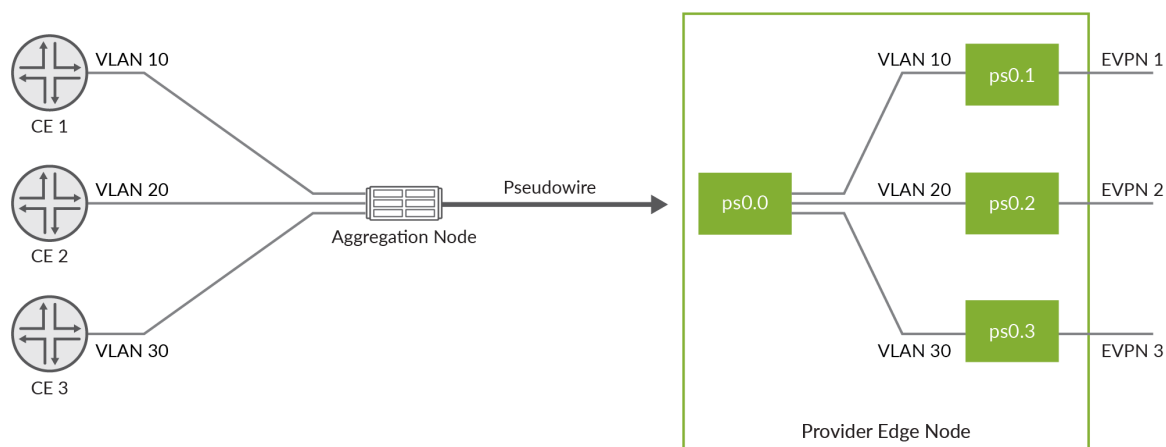
### IN THIS SECTION

- [Benefits of Pseudowire Termination at an EVPN | 1447](#)
- [Support for Junos Node Slicing | 1447](#)

Pseudowires provide point-to-point service over an MPLS or Layer 2 circuit. They enable providers to extend their Layer 2 networks. A pseudowire tunnel terminates on a logical interface on the PE router with the transport logical interface configured for interoperability with the access or aggregation device on the other end of the pseudowire tunnel. It is identified as psn.0 and supports both port-based and VLAN-based encapsulation over pseudowire. The corresponding service logical interfaces are identified as psn.1 to psn.n and are configured to support EVPN in the EVPN routing instance.

[Figure 139 on page 1447](#) shows a port-based pseudowire tunnel that originates on an aggregation node carrying VLAN traffic from several CE devices. The pseudowire terminates in a transport logical interface (ps0.0) on a single-homed PE router, where the VLAN traffic is demultiplexed into different service logical interfaces (ps0.1, ps0.2, and ps0.3) along with their corresponding EVPN routing instances.

**Figure 139: Pseudowire Terminating into Single-Homed PE Device**



## Benefits of Pseudowire Termination at an EVPN

- **Resiliency**—When a redundant logical tunnel is configured on a separate FPC, connectivity for the pseudowire automatically switches to another FPC.
- **Reduced cost**—Rather than using additional devices to terminate a pseudowire tunnels, you can configure pseudowire termination on the same device that also supports EVPN.

## Support for Junos Node Slicing

Pseudowire termination is supported on multiple partitions in a single MX Series. Using Junos Node Slicing, you can create multiple partitions on a single MX router. These partitions are referred to as a guest network functions (GNFs). For more information on Junos Node slicing, see [Junos Node Slicing User Guide](#).

## Configuring Pseudowire Termination

To configure a pseudowire logical interface on the PE device, perform these tasks:

- Configure a logical interface.
- Configure the transport logical interface
- Configure the service logical interface.
- Configure the EVPN routing instance to support the incoming VLAN services.



To configure the logical interface:

1. Specify that you want to configure the pseudowire logical interface device.

```
user@host# edit interfaces ps0
```

2. Specify the logical tunnel interface that is the anchor point for the pseudowire logical interface device. In this case, the logical tunnel interface and anchor point is in the format *lt-fpc/pic/port*.



**NOTE:** Tunnel services must be enabled in order to create the **lt** interface that is the anchor point or a member link in a redundant logical tunnel. You use the command, `set chassis fpc slot-number pic pic-number tunnel-services bandwidth bandwidth` to enable tunnel services.

```
[edit interfaces ps0]
user@host# set anchor-point lt-1/0/10
```

3. (Optional) Specify the MAC address for the pseudowire logical interface device.



**NOTE:** You should ensure that you change the MAC address prior to passing traffic or binding subscribers on the pseudowire port. Changing the MAC address when the pseudowire port is active (for example, while an upper layer protocol is negotiating) can negatively impact network performance until adjacencies learn of the new MAC address.

```
[edit interfaces ps0]
user@host# set mac 00:00:5E:00:53:55
```

4. (Optional) Specify the VLAN tagging method used for the pseudowire logical interface device. You can specify single tagging, dual (stacked) tagging, mixed (flexible) tagging, or no tagging.

```
[edit interfaces ps0]
user@host# set flexible-vlan-tagging
```

See [Enabling VLAN Tagging](#) for additional information about VLAN tagging.

To configure the logical interface:

1. Specify that you want to configure the pseudowire logical interface.

```
user@host# edit interfaces ps0
```

2. Specify that you want to configure unit 0 , which represents the transport logical interface.

```
[edit interfaces ps0]
user@host# edit unit 0
```

3. Specify either the encapsulation method for the transport logical interface. You can specify either ethernet-ccc (port-based) or vlan-ccc (vlan-based) encapsulation.

```
[edit interfaces ps0 unit 0]
user@host# set encapsulation ethernet-ccc
```

To configure the service logical interface:

1. Configure the unit for the service logical interface. Use a non-zero unit number.

```
user@host# edit interfaces ps0 unit1
```

2. Configure the encapsulation on the service interface.

```
[edit interfaces ps0 unit 1]
user@host# set encapsulation vlan-bridge;
```

3. (Optional) Configure the VLAN IDs.

```
[edit interfaces ps0 unit 1]
user@host# set vlan-id vlan-id;
```

4. (Optional) Configure the VLAN tag IDs to support dual-tagged VLANs.

```
[edit interfaces ps0 unit 1]
set vlan-tags inner vlan-id outer vlan-id;
```

The following is a sample configuration for a logical interface with services supporting single-tag and dual-tag VLANs and the L2 circuit associated with the interface:

```
ps0 {
 anchor-point {
 lt-0/0/0;
 }
 flexible-vlan-tagging;
 mac 00:00:5E:00:53:00;
 unit 0 {
 encapsulation ethernet-ccc;
 }
 unit 1 {
 encapsulation vlan-bridge;
 vlan-id 600;
 }
 unit 2 {
 encapsulation vlan-bridge;
 vlan-tag outer 200 inner 101;
 }
}
protocols {
 l2circuit {
 neighbor 192.168.100.1 {
 interface ps0.0 {
 virtual-circuit-id 700;
 encapsulation-type ethernet-vlan;
 ignore-mtu-mismatch;
 pseudowire-status-tlv;
 }
 }
 }
}
```

The following is a sample configuration for an EVPN routing instance that corresponds to a service logical interface:

```
routing-instances evpn-1 {
 instance-type evpn;
 vlan-id 600;
 interface ps0.1;
```

```

route-distinguisher 3;3;
vrf-target target:1:1;
protocols {
 evpn;
}
}

```

For more information on configuring logical interface termination, see [Pseudowire Subscriber Logical Interfaces Overview](#).

For more information on configuring EVPN routing instances, see "[Configuring EVPN Routing Instances](#)" on page 31.

## Support for Redundant Logical Tunnel

You can configure two logical tunnels on two different line cards to create a redundant logical tunnel (RLT). This provides redundancy when there is a failure in the FPC. Pseudowire over RLT supports two members in active-standby mode. Only one member link is active and carrying traffic at any given time.

To create the RLT, configure a pair of `redundant-logical-tunnel` interface at the `[edit chassis redundancy-group interface-type]` hierarchy and include the logical tunnel interface in the redundancy group by configuring `member-interface interface-name` at the `[edit interfaces interface-name redundancy-group]` hierarchy level.

The following is a sample configuration for pseudowire RLT.

```

chassis {
 pseudowire-service {
 device-count 500;
 }
 redundancy-group {
 interface-type {
 redundant-logical-tunnel {
 device-count 10;
 }
 }
 }
}
interfaces {
 rlt0 {
 redundancy-group {
 member-interface lt-0/1/0;

```

```
 member-interface lt-0/1/1;
 }
}
ps0 {
 anchor-point {
 rlt0;
 }
}
}
```

For more information on redundant logical tunnels, see [Redundant Logical Tunnels Overview](#)

# Configuring the Distribution of Routes

## IN THIS CHAPTER

- [Configuring an IGP on the PE and P Routers on EX9200 Switches | 1453](#)
- [Configuring IBGP Sessions Between PE Routers in VPNs on EX9200 Switches | 1454](#)
- [Configuring a Signaling Protocol and LSPs for VPNs on EX9200 Switches | 1455](#)
- [Configuring Entropy Labels | 1458](#)
- [Configuring Control Word for EVPN-MPLS | 1459](#)
- [Understanding P2MPs LSP for the EVPN Inclusive Provider Tunnel | 1461](#)
- [Configuring Bud Node Support | 1463](#)

## Configuring an IGP on the PE and P Routers on EX9200 Switches

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.

## RELATED DOCUMENTATION

---

[Understanding IS-IS Configuration](#)

---

[Example: Configuring IS-IS](#)

---

[OSPF User Guide](#)

## Configuring IBGP Sessions Between PE Routers in VPNs on EX9200 Switches

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.

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.

## RELATED DOCUMENTATION

[Configuring a Signaling Protocol and LSPs for VPNs on EX9200 Switches | 1455](#)

## Configuring a Signaling Protocol and LSPs for VPNs on EX9200 Switches

### IN THIS SECTION

- [Using LDP for VPN Signaling | 1456](#)



For VPNs to function, you must enable the LDP signaling protocol on the provider edge (PE) routers and on the provider (P) routers.

To enable the LDP signaling protocol, perform the steps in the following section:

## 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 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 1456](#)) 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;
 }
}
```

For more information about configuring OSPF and IS-IS, see the [OSPF User Guide](#) and [IS-IS User Guide](#).

## RELATED DOCUMENTATION

| [EVPN Overview for Switches](#) | 1432

## Configuring Entropy Labels

An entropy label is a special label that enhances a network device's ability to load-balance traffic across equal-cost multipath (ECMP) paths or link aggregation groups (LAGs). When you configure an entropy label at the ingress device, the transit device use the label to determine a path in place of performing a deep packet inspection (DPI). If the transit device does not support LDP Entropy Label Capability (ELC), the transit device ignores the entropy label and continues to propagate the packet using traditional DPI for load balancing. The entropy label is popped at the penultimate hop device.

An entropy label can be any label value between 16 to 1048575 (regular 20-bit label range). Since this range overlaps with the existing regular label range, a special label called entropy label indicator (ELI) is also inserted in the label stack before the entropy label. IANA assigned ELI label a value of 7. This distinguish entropy labels from other service labels.

To configure entropy labels, include the **entropy label** at the **[edit protocols ldp]** hierarchy on the ingress device and include the configured policy in the routing policy.

```
protocols {
 ldp {
 entropy-label {
 ingress-policy policy-name;
 }
 }
}
```

```
policy-options {
 policy-statement policy-name {
 term Add-Entropy-label {
 from {
 route-filter route exact;
 }
 }
 }
}
```

```

 then accept;
 }
}

```

To enable a penultimate hop device to pop the entropy label, include the **load-balance-label-capability** statement at the **[edit forwarding-option]** hierarchy.

```

forwarding-options {
 load-balance-label-capability;
}

```

## RELATED DOCUMENTATION

[Configuring the Entropy Label for LSPs](#)

[entropy-label](#)

## Configuring Control Word for EVPN-MPLS

Transit devices in an EVPN-MPLS network are not aware of the type of payload it carries. While parsing an MPLS encapsulated packet for hashing, a transit device 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, causing out-of-order packet delivery. To identify the payload as an Ethernet payload, we can insert a control word with a value of 0 in the first four bits between the label stack and the L2 header of the packet. This ensures that the packet is not identified as an IPv4 or IPv6 packet.



**NOTE:** You do not need to enable control word on the devices in your transit network when the transit devices consist of Juniper Networks EX 9200 switches, MX series routers, or PTX Series routers. These Juniper devices correctly identify the Ethernet payload as an IPv4/IPv6 payloads, even when the Ethernet destination MAC address starts with 0x4 or 0x6 nibble. The Juniper devices perform hashing based on the IP header fields inside the Ethernet frame and will not send out-of-order packets. In this case, we recommend not using control word as there are no benefits..

To enable control word, set control-word for the evpn protocol for a specified routing instance. The following output shows a sample multihomed routing instance with control word configured

```
user@router1# show routing-instances
routing-instances EVPN-green
vlan-id 200;
interface ae0.1;
route-distinguisher 10.255.255.1:200;
vrf-target target:100:200;
protocols {
 evpn {
 control-word;
 }
}
```

To view routes where control word is supported, use the show route table mpls.0 protocol evpn operational command. Egress routes display an offset of 252.

```
show route table mpls.0 protocol evpn
303744 *[EVPN/7] 00:00:13, remote-pe 10.255.255.2, routing-instance blue, route-type
Egress-MAC
312 > to 5.0.0.1 via ge-0/0/2.0, Push 299984 Offset: 252
313 303760 *[EVPN/7] 00:00:13, remote-pe 10.255.255.2, routing-instance blue,
route-type Egress-MAC
314 > to 5.0.0.1 via ge-0/0/2.0, Push 299888 Offset: 252
315 303776 *[EVPN/7] 00:00:13, remote-pe 10.255.255.2, routing-instance blue,
route-type Egress-MAC
316 > to 5.0.0.1 via ge-0/0/2.0, Push 300032 Offset: 252
317 303792 *[EVPN/7] 00:00:13, remote-pe 10.255.255.2, routing-instance blue,
route-type Egress-IM, vlan-id 4
318 > to 5.0.0.1 via ge-0/0/2.0, Push 302000 Offset: 252
319 303808 *[EVPN/7] 00:00:13, remote-pe 10.255.255.2, routing-instance blue,
route-type Egress-IM, vlan-id 5
320 > to 5.0.0.1 via ge-0/0/2.0, Push 302016 Offset: 252
321 303824 *[EVPN/7] 00:00:13, remote-pe 10.255.255.2, routing-instance blue,
route-type Egress-IM, vlan-id 6
322 > to 5.0.0.1 via ge-0/0/2.0, Push 302032 Offset: 252
```

## Understanding P2MPs LSP for the EVPN Inclusive Provider Tunnel

### IN THIS SECTION

- [NSR and Unified ISSU Support on EVPN with P2MP | 1462](#)
- [Benefits of EVPN P2MPs LSP for the EVPN Inclusive Provider Tunnel | 1462](#)

An EVPN instance comprises Customer Edge devices (CEs) that are connected to Provider Edge devices (PEs) to form the edge of the MPLS infrastructure. A CE may be a host, a router, or a switch. The PEs provide virtual Layer 2 bridged connectivity between the CEs. There may be multiple EVPN instances in the provider's network. The PEs may be connected by an MPLS Label Switched Path (LSP) infrastructure, which provides the benefits of MPLS technology, such as fast reroute, resiliency, etc.

The PEs may also be connected by an IP infrastructure, where IP/GRE (Generic Routing Encapsulation) tunneling or other IP tunneling can be used between the PEs. Here we understand the MPLS LSPs as the tunneling technology designed to be extensible to IP tunneling as the Packet Switched Network (PSN) tunneling technology.

Starting in Junos OS Release 18.2R1 onwards, Junos OS provides the ability to configure and signal a P2MP LSP for the EVPN Inclusive Provider Tunnel for BUM traffic. P2MP LSPs can provide efficient core bandwidth utilization by using the multicast replication only at the required nodes instead of ingress replication at the ingress PE.

You can configure and signal a P2MP LSP for the EVPN Inclusive Provider Tunnel in the PMSI Attributes of the Inclusive Multicast Ethernet Tag Route. The P2MP tunnel is used for forwarding Broadcast, unknown Unicast, and Multicast (BUM) packets at the ingress PE. When representing the PMSI attributes in the Inclusive Multicast Ethernet Tag Route, a transport label is not represented for P2MP Provider Tunnels, except in the case where Aggregation is used. The transport label is used to identify the EVI at the egress PE.

When the P2MP Provider Tunnels are used, the ESI labels for Split Horizon are assigned upstream instead of downstream. The label that the egress PE receives as a Split Horizon label is allocated by the ingress PE. Since each upstream PE cannot allocate the same label, the label does not identify the ESI and must be referred in the context label space of the ingress PE. The transport label uniquely identifies the ingress PE and the split horizon label is identified by transport-label or SH-label tuple.

At the ingress PE, an upstream assigned ESI label is allocated and signaled for Split Horizon function. This label is added to the BUM traffic that originates from the given Ethernet Segment. By default, an egress may receive traffic with an ESI label that is not configured on this PE because of P2MP tunnels and upstream assigned labels.



**NOTE:** For IR tunnels, ingress includes the ESI label to only those PEs with the ES. In such case, the egress PE pops the ESI label and floods the packet normally.

For E-tree, the leaf label is assigned upstream and its function is similar to the SH label. The ingress PE allocates the leaf label and represents it in the E-tree extended community for the Ethernet A-D per ES route. When the Leaf PE acts as an ingress, it adds the leaf label for BUM traffic. An egress PE acting as a leaf discards the traffic which contains the leaf label. Egress PEs acting as a root pops the leaf label and forwards the traffic. Since leaf labels are upstream assigned, they may not be unique because multiple Leaf PEs may have allocated the same leaf label. Therefore, the leaf label must be identified by the (transport-label, Leaf-label) tuple.



**NOTE:** The EVPN P2MP functions without a `lsi` or `vt` interface. Instead, the EVPN allocates a label for use as the mLDP/RSVP transport label at the egress PE.

## NSR and Unified ISSU Support on EVPN with P2MP

Nonstop active routing (NSR) and graceful Routing Engine switchover (GRES) minimize traffic loss when there is a Routing Engine switchover. When a Routing Engine fails, NSR and GRES enable a routing platform with redundant Routing Engines to switch over from a primary Routing Engine to a backup Routing Engine and continue forwarding packets.

Unified in-service software upgrade (ISSU) allows you to upgrade your Junos OS software on your MX Series router with no disruption on the control plane and with minimal disruption of traffic. Both GRES and NSR must be enabled to use unified ISSU. Nonstop active routing (NSR) and graceful Routing Engine switchover (GRES) minimize traffic loss when there is a Routing Engine switchover.

When a Routing Engine fails, NSR and GRES enable a routing platform with redundant Routing Engines to switch over from a primary Routing Engine to a backup Routing Engine and continue forwarding packets. Unified in-service software upgrade (ISSU) allows you to upgrade your Junos OS software on your MX Series router with no disruption on the control plane and with minimal disruption of traffic. Both GRES and NSR must be enabled to use unified ISSU.

Junos OS mirrors essential data when NSR is enabled. For EVPN with P2MP mLDP replication, the LDP/RSVP transport label will be mirrored on the standby Routing Engine. For information on other mirrored data and NSR data flow, see ["NSR and Unified ISSU Support for EVPN" on page 565](#).

## Benefits of EVPN P2MPs LSP for the EVPN Inclusive Provider Tunnel

- Provides efficient core bandwidth utilization by using multicast replication only at the required nodes.

- Manages the ingress replication at the ingress PE to avoid an over-load.
- Supports P2MP LSPs for EVPN Inclusive P-Multicast Trees for EVPN MPLS for both mLDP and RSVP-TE P2MP-E-tree.

## RELATED DOCUMENTATION

[Example: Configuring Point-to-Multipoint LDP LSPs as the Data Plane for Intra-AS MBGP MVPNs](#)

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

*Flooding Unknown Traffic Using Point-to-Multipoint LSPs in VPLS*

*Configuring Point-to-Multipoint LSPs for an MBGP MVPN*

*Configuring Dynamic Point-to-Multipoint Flooding LSPs*

[Configuring RSVP Automatic Mesh](#)

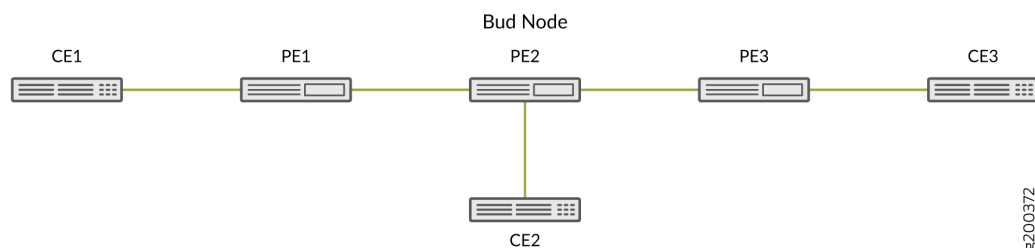
[container-label-switched-path](#)

## Configuring Bud Node Support

Along a LSP, PE devices function as an ingress device, transit device or egress device. A bud node is a PE device that functions as both an egress and transit device. In a P2MP LSP, you can have devices that will function in the role of a both as a transit device and an egress device.

[Figure 140 on page 1463](#) illustrates a simple EVPN network with a bud node at PE2. When CE1 sends a multicast packet out, PE2 operates as a transit device and forwards the packet to PE3. It also functions as a egress device and pops the MPLS label and replicates multicast packets destined for CE2.

**Figure 140: Bud Node in an EVPN Network**



To enable a PE device to function as a bud-node, include `p2mp-bud-support` statement at the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy. When `p2mp-bud-support` is enabled or disabled, you



may observed dropped packets on the device. This occurs because changing bud node support affects the forwarding state in the routing instance, which results in the forwarding table being rebuilt.



**NOTE:** We recommend that all PE devices in the LSP that may potentially function as both a egress and transit device be enabled as a bud node.

# Configuring VLAN Services and Virtual Switch Support

## IN THIS CHAPTER

- [Overview of VLAN Services for EVPN | 1465](#)
- [VLAN-Based Service for EVPN | 1467](#)
- [VLAN Bundle Service for EVPN | 1471](#)
- [Configuring EVPN VLAN Bundle Services | 1473](#)
- [Virtual Switch Support for EVPN Overview | 1476](#)
- [Configuring EVPN with Support for Virtual Switch | 1478](#)
- [Example: Configuring EVPN with Support for Virtual Switch | 1482](#)

## Overview of VLAN Services for EVPN

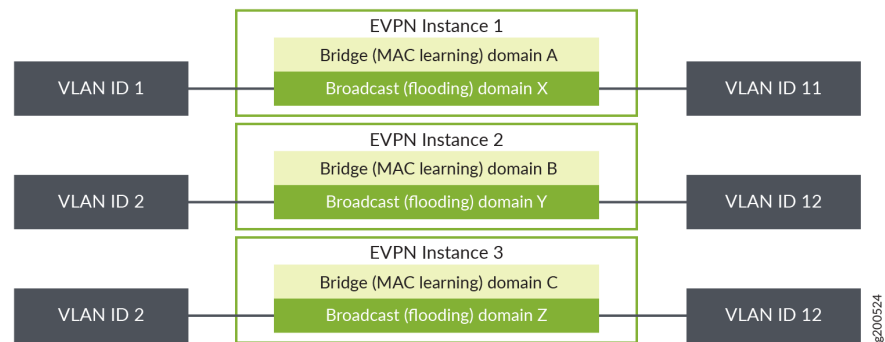
A Data Center Service Provider (DCSP) hosts the data center for its multiple customers on a common physical network. To each customer (also called a tenant), the service looks like a full-fledged data center that can expand to 4094 VLANs and all private subnets. For disaster recovery, high availability, and optimization of resource utilization, it is common for the DCSP to span across the data center to more than one site. When deploying the data center services, a DCSP faces the following main challenges:

- Extending Layer 2 domains across more than one data center site. This requires optimal intra-subnet traffic forwarding.
- Supporting optimal inter-subnet traffic forwarding and optimal routing in the event of virtual machine (VM).
- Supporting multiple tenants with independent VLAN and subnet space.

The DCSP might require Ethernet VLAN services to be extended over a WAN with a single EVPN instance (EVI). Junos OS supports VLAN-based service, VLAN bundle service, and VLAN-aware bundle service, while maintaining data and control plane separation. The following figures illustrate the relationship between VLANs and EVIs.

- [Figure 141 on page 1466](#) illustrates a VLAN-based service where you have a one-to-one mapping between a VLAN ID and a EVI.
- [Figure 142 on page 1466](#) illustrates a VLAN bundles service where you can have one EVI mapped to many VLAN IDs in a single bridge domain. The bridge table is shared among the VLANs.
- [Figure 143 on page 1467](#) illustrates the relationship between VLANs and EVIS in a VLAN-aware bundle service where you can have one EVI mapped to many VLAN IDs. Each VLAN has a different bridge table.

**Figure 141: VLAN-Based Service**



**Figure 142: VLAN Bundle Service**

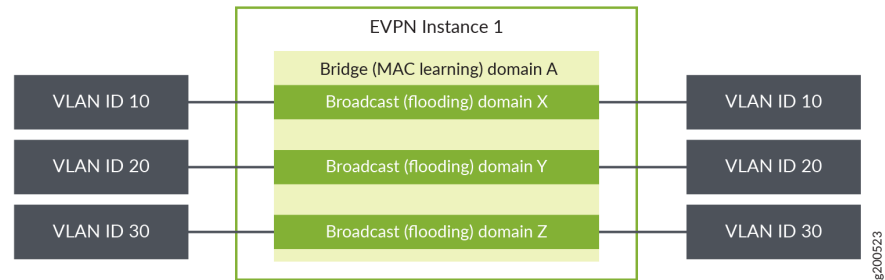
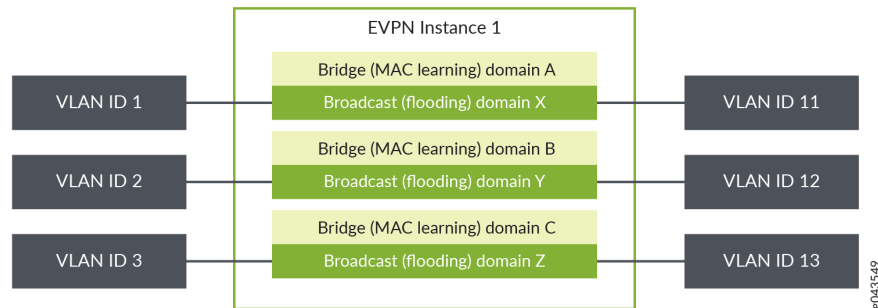


Figure 143: VLAN-Aware Bundle Service



## RELATED DOCUMENTATION

[VLAN-Based Service for EVPN | 1467](#)

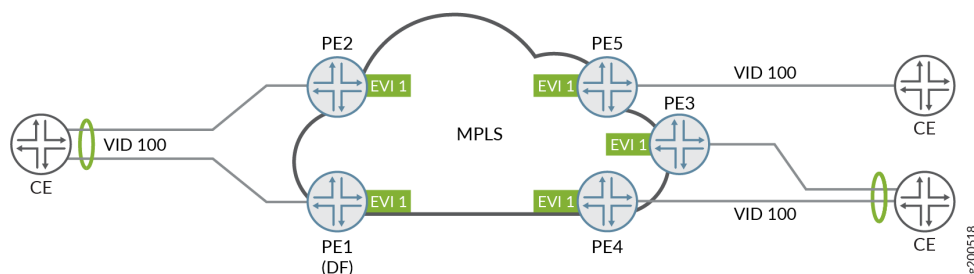
[VLAN Bundle Service for EVPN | 1471](#)

[Virtual Switch Support for EVPN Overview | 718](#)

## VLAN-Based Service for EVPN

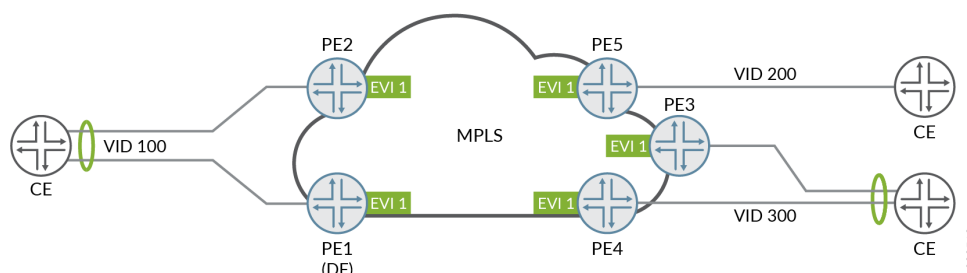
VLAN-based service allows a one-to-one mapping of a single broadcast domain to a single bridge domain. Each VLAN is mapped to a single EVPN instance (EVI), resulting in a separate bridge table for each VLAN. VLAN translation is not supported in older Junos releases. Without VLAN translation, the customer edge VLAN must use the same VLAN ID (VID). You can still send an MPLS encapsulated frames with the originating VID. [Figure 144 on page 1467](#) illustrates a topology where all the CE devices use the same CE VID for a single VLAN-based EVI. VID translation is not needed.

Figure 144: Single VID with no VLAN Translation



VLAN-based service with VID translation as described in RFC 7432 is supported for devices running Junos OS software. This means that Junos supports VID translation and the customer can have a different VID for each VLAN. As described in the RFC, the VID translation must be performed at the egress PE device while the MPLS encapsulated frames should also retain the originating VID. [Figure 145 on page 1468](#) illustrates a topology where CE devices use different CE-VIDs for single VLAN-based EVI.

**Figure 145: Multiple VIDs with VLAN Translation**



For more information on configuring VLAN-based service, see ["Configuring EVPN with VLAN-Based Service" on page 711](#).

The following is a sample configuration for a single VLAN-based EVI. In this example, the `VLAN-id=none` statement is included to remove the originating VID and to set the Ethernet tag ID to zero in the MPLS frame. This ensures that the same VID is used on all the PE devices and VLAN translation is not required.

```
interfaces {
 xe-0/0/1 {
 unit 100 {
 encapsulation vlan-bridge;
 vlan-id 100;
 }
 }
}

routing-instances evpn-vlan-based-no-vid {
 instance-type evpn;
 vlan-id none;
 interface xe-0/0/1.100;
 route-distinguisher 10.0.0.1:100;
 vrf-target target:65303:101100;
 protocols evpn;
}
```

The following is a sample configuration for a single VLAN-based EVI. The same VID is used on all the PE devices, so VLAN translation is not required. In this example, the CE-VID is used and sent as part of the MPLS frame.

```
interfaces {
 xe-0/0/1 {
 unit 100 {
 encapsulation vlan-bridge;
 vlan-id 100;
 }
 }
}
routing-instances evpn-vlan-based-with-vid {
 instance-type evpn;
 interface xe-0/0/1.100;
 route-distinguisher 10.0.0.1:100;
 vrf-target target:65303:101100;
 protocols evpn;
}
```

Junos supports VLAN-based service with translation as described in RFC 7432. The following is a sample VLAN-based service configuration that adheres to a strict compliance of RFC 7432. Strict compliance to RFC 7432 requires that translation occurs at the egress PE device, the originating VID be carried in the MPLS frame, and the Ethernet tag ID be set to zero for all EVPN routes. Therefore, the `VLAN-id=none` and the `no-normalization` statements are included. This will set the Ethernet tag ID to zero, while ensuring that different VIDs can still be used.

```
interfaces {
 xe-0/0/1 {
 unit 100 {
 encapsulation vlan-bridge;
 vlan-id 100;
 output-vlan-map {
 swap;
 }
 }
 }
}
routing-instances evpn-vlan-based-normalization-strict-RFC-compliance {
 instance-type evpn;
 vlan-id none;
}
```

```

no-normalization;
interface xe-0/0/1.100;
route-distinguisher 10.0.0.1:100;
vrf-target target:65303:101100;
protocols evpn;
}

```

The following is a sample VLAN-based service configuration that adheres to RFC 7432 except for the condition that the originating VID be carried in the Ethernet frame. The originating VID is removed and the Ethernet tag ID is set to zero.

```

interfaces {
 xe-0/0/1 {
 unit 100 {
 encapsulation vlan-bridge;
 vlan-id 100;
 }
 }
}
routing-instances evpn-vlan-based-normalization-loose-RFC-compliance {
 instance-type evpn;
 vlan-id none;
 interface xe-0/0/1.100;
 route-distinguisher 10.0.0.1:100;
 vrf-target target:65303:101100;
 protocols evpn;
}

```

Starting with Junos OS Release 24.2R1, you can use the *advertise-zero-ethernet-tag* configuration statement when you need a vlan-based service with a valid `vlan-id` that provides Layer 3 gateway functionality. This also provides RFC 7432 compliance for Layer 2 gateway functionality with control plane EVPN routes advertised with an Ethernet Tag ID value of 0. You use this statement with `instance-type evpn` routing instances and must have a valid `vlan-id` and `no-normalization` configured.

Please refer to Feature Explorer [VLAN-based EVPN with IRB interfaces](#) for a complete list of the products that support this feature.

The following is a sample configuration using the `advertise-zero-ethernet-tag` statement and a valid `vlan-id`:

```

[edit routing-instances evpna]
instance-type evpn;
protocols {

```

```

 evpn {
 advertise-zero-ethernet-tag;
 }
}
vlan-id 600;
routing-interface irb.600;
no-normalization;
interface ge-0/0/1.600;
route-distinguisher 1:1;
vrf-target target:1:1;

```

## RELATED DOCUMENTATION

[Stacking and Rewriting Gigabit Ethernet VLAN Tags Overview](#)

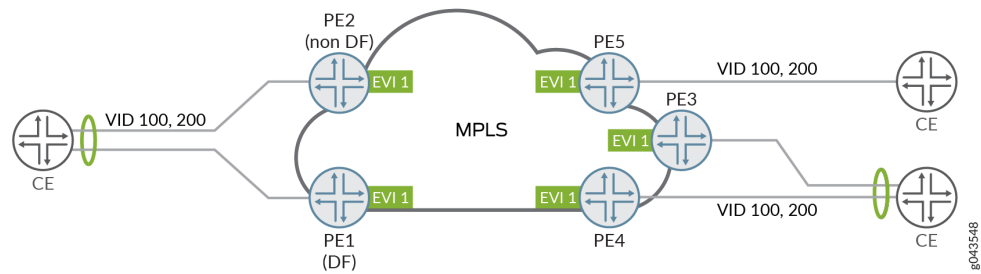
## VLAN Bundle Service for EVPN

VLAN bundle service allows multiple broadcast domains to map to a single bridge domain. Multiple VLANs are mapped to a single EVPN instance (EVI) and share the same bridge table in the MAC-VRF table, thus reducing the number of routes and labels stored in the table. This lowers the control plane overhead on the router. Having a single bridge domain requires all CE devices in the VLAN network to have unique MAC addresses. VLAN ID translation is not permitted as the MPLS encapsulated frames must retain the originating VLAN ID. As such the Ethernet Tag ID in all EVPN routes is set to zero. A VLAN range cannot be specified. The entire VLAN from 1 to 4095 must be bundled on the interface.




[Figure 146 on page 1472](#) illustrates a topology where VID 100 and VID 200 are bundled and assigned to EVI 1. The service provider creates a single broadcast domain for the customer and assigns a preconfigured number of CE-VIDs on the ingress PE routers (PE1 through PE5) to EVI 1. All CE devices use the same CE-VIDs for the EVI.



Figure 146: VLAN Bundle Network Topology



Using the VLAN bundle service reduces the number of routes and labels, which in turn, reduces the control plane overhead. The trade-off for the ease of provisioning in a customer network is that the service provider has no control over the customer broadcast domain since there is a single inclusive multicast tree and no CE-VID translation. .

- **NOTE:** Junos OS also supports port-based VLAN bundle service where all of the VLANs on a port are part of the same service and are mapped to the same bundle.
- **NOTE:** Integrated routing and bridging (IRB) is not supported for VLAN bundle service.
- **NOTE:** Arp suppression is automatically disabled when you configure EVPN bundle services. As a result, you may observe ARP packets being sent by the device.

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                                                                             |
|---------|-----------------------------------------------------------------------------------------|
| 17.1    | VLAN bundle service allows multiple broadcast domains to map to a single bridge domain. |

RELATED DOCUMENTATION

| [Configuring EVPN VLAN Bundle Services](#) | 1473

## Configuring EVPN VLAN Bundle Services

To configure EVPN VLAN bundle services, complete the following configuration on all PE routers within the EVPN service provider's network:

1. Configure the EVPN routing instance name using the `routing-instances` statement at the [edit] hierarchy level:

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

2. Configure the `evpn` option for the `instance-type` statement at the [edit `routing-instances routing-instance-name`] hierarchy level:

```
instance-type evpn;
```

3. Configure a route distinguisher on a PE router by including the `route-distinguisher` statement:

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

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 a list of the hierarchy levels at which you can include the `route-distinguisher` statement, see *route-distinguisher*.

The route distinguisher is a 6-byte value that you can specify in either 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 ISP's own or the customer's own AS number.



**NOTE:** The automatic derivation of the BGP route target (auto-RT) for advertised prefixes is supported on 2-byte AS numbers only.

- *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.

4. Configure either import and export policies for the EVPN routing table, or configure the default policies using the `vrf-target` statement at the `[edit routing-instances routing-instance-name]` hierarchy level.

See *Configuring Policies for the VRF Table on PE Routers in VPNs*.

5. Configure each EVPN interface for the EVPN routing instance:



**NOTE:** Adding a trunk port with dual tags to an EVPN and MPLS routing instance to an EVPN and VXLAN routing instance causes the CLI commit check configuration to fail and generate an error. To avoid configuration check errors, use sub-interface style interface configuration with dual tags for the routing instances.

- Configure each interface using the `interface` statement at the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy level.
- Configure interface encapsulation for the CE-facing interfaces at the `[edit interfaces interface-name encapsulation]` hierarchy level. Supported encapsulations are `ethernet-bridge`, `vlan-bridge`, and `extended-vlan-bridge`.



**NOTE:** If you are configuring the port-based VLAN bundle service, you will need to also configure the interface encapsulation `ethernet-bridge` unit to 0.

- (Optional) Include the `ignore-encapsulation-mismatch` statement at the `[edit routing-instances routing-instance-name protocols evpn interface interface-name]` hierarchy level to allow the EVPN to establish a connection to the CE device even if the CE device interface encapsulation and the EVPN interface encapsulations do not match.
  - Specify a static MAC address for a logical interface in a bridge domain using the `static-mac` statement at the `[edit routing-instances routing-instance-name protocols evpn interface interface-name]` hierarchy level.
6. Specify the maximum number of media access control (MAC) addresses that can be learned by the EVPN routing instance by including the `interface-mac-limit` statement.

You can configure the same limit for all interfaces configured for a routing instance by including the `interface-mac-limit` statement at the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy level. You can also configure a limit for a specific interface by including this statement at the `[edit routing-instances routing-instance-name protocols evpn interface interface-name]` hierarchy level.

By default, packets with new source MAC addresses are forwarded after the MAC address limit is reached. You can alter this behavior by including the `packet-action drop` statement at either the `[edit`

`routing-instances routing-instance-name protocols evpn interface-mac-limit]` or the `[edit routing-instances routing-instance-name protocols evpn interface interface-name]` hierarchy level. If you configure this statement, packets from new source MAC addresses are dropped after the configured MAC address limit is reached.

7. Specify the MPLS label allocation setting for the EVPN by including the *label-allocation* statement with the per-instance option at the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy level.

If you configure this statement, one MPLS label is allocated for the specified EVPN routing instance.

8. Enable MAC accounting for the EVPN by including the *mac-statistics* statement at the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy level.
9. Disable MAC learning by including the *no-mac-learning* statement at either the `[edit routing-instances routing-instance-name protocols evpn]` hierarchy level to apply this behavior to all of the devices configured for an EVPN routing instance or at the `[edit routing-instances routing-instance-name protocols evpn interface interface-name]` hierarchy level to apply this behavior to only one of the CE devices.



**NOTE:** Arp suppression is automatically disabled when you configure EVPN bundle services. As a result, you may observe ARP packets being sent by the device.

The following output shows a sample EVPN VLAN bundle services configuration.

```
user@router1# show routing-instances evpn_1
instance-type evpn;
 interface xe-2/3/3.300;
 route-distinguisher 7.7.7.7:3;
 vrf-target target:65221:3;
 protocols evpn;
 label-allocation per-instance;
}
```

## RELATED DOCUMENTATION

VLAN Bundle Service for EVPN | 1471

## Virtual Switch Support for EVPN Overview

Starting with Junos OS Release 14.1, VLAN-aware bundle service is introduced on MX Series routers. Starting with Junos OS Release 14.2, VLAN-aware bundle service is introduced on EX9200 switches. Starting with Junos OS Release 17.3, VLAN-aware bundle service is introduced on QFX Series switches. This feature allows Ethernet VLANs over a WAN to share a single EVPN instance while maintaining data-plane separation between the different VLANs.

Junos OS has a highly flexible and scalable virtual switch interface. With a virtual switch, a single router or switch can be divided into multiple logical switches. Layer 2 domains (also called *bridge-domains* or *vlan*s) can be defined independently in each virtual switch. To configure VLAN-aware bundle service, an EVPN must run in a virtual-switch routing instance.

On the EX Series and MX Series, a single EVPN instance can stretch up to 4094 bridge domains or VLANs defined in a virtual switch to remote sites. A virtual switch can have more than 4094 bridge domains or VLANs with a combination of none, single, and dual VLANs. However, because EVPN signaling deals only with single VLAN tags, a maximum of 4094 bridge domains or VLANs can be stretched. The EVPN virtual switch also provides support for trunk and access interfaces.

Starting with Junos OS Release 17.3R1, on the QFX10000 line of switches, you can configure both EVPN and virtual-switch routing instances. The EVPN routing instance supports VLAN-based service. With VLAN-based service, the EVPN instance includes only a single broadcast domain, and there is a one-to-one mapping between a VNI and MAC-VRF. Up to 100 EVPN routing instances are supported. The virtual-switch routing instance supports VLAN-aware service, and up to 10 virtual-switch routing instances with 2000 VLANs are supported.

If you create VLANs that are not part of a routing instance, they become part of the default switch routing instance.



### NOTE:

- The `none` VLAN option is supported with bridge domains or VLANs under the virtual switch instance type for EVPNs.
- Access interfaces configured with single or dual VLAN tags are supported in EVPN. By default, only Ethernet frames with single or no VLAN tags are transported across the EVPN core. As a result, dual-tagged Ethernet frames received on the access interfaces must be normalized to Ethernet frames with single or no VLAN tags for proper transmission over the EVPN core.

You can enable transporting of dual-tagged frames across the EVPN core network by including both the `vlan-id none` and `no-normalization` configuration statements together.

- Dual tagged trunk interfaces are not supported in an EVPN virtual switch instance type for EVPN-MPLS.

There are two types of VLAN-aware bundle service:

- VLAN-aware bundle without translation

The service interface provides bundling of customer VLANs into a single Layer 2 VPN service instance with a guarantee end-to-end customer VLAN transparency. The data-plane separation between the customer VLANs is maintained by creating a dedicated bridge-domain for each VLAN.

- VLAN-aware bundle with translation

The service interface provides bundling of customer VLANs into a single Layer 2 VPN service instance. The data-plane separation between the customer VLANs is maintained by creating a dedicated bridge-domain for each VLAN. The service interface supports customer VLAN translation to handle the scenario where different VLAN Identifiers (VIDs) are used on different interfaces to designate the same customer VLAN.

EVPN with virtual switch provides support for VLAN-aware bundle with translation only.

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                                                                                                                                |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 17.3R1  | Starting with Junos OS Release 17.3R1, on the QFX10000 line of switches, you can configure both EVPN and virtual-switch routing instances. |
| 17.3    | Starting with Junos OS Release 17.3, VLAN-aware bundle service is introduced on QFX Series switches.                                       |
| 14.2    | Starting with Junos OS Release 14.2, VLAN-aware bundle service is introduced on EX9200 switches.                                           |
| 14.1    | Starting with Junos OS Release 14.1, VLAN-aware bundle service is introduced on MX Series routers.                                         |

RELATED DOCUMENTATION

| [Example: Configuring EVPN with Support for Virtual Switch](#) | 1482

## Configuring EVPN with Support for Virtual Switch

You can configure an Ethernet VPN (EVPN) with virtual switch support to enable multiple tenants with independent VLAN and subnet space within an EVPN instance. Virtual switch provides the ability to extend Ethernet VLANs over a WAN using a single EVPN instance while maintaining data-plane separation between the various VLANs associated with that instance. A single EVPN instance can stretch up to 4094 bridge domains defined in a virtual switch to remote sites.

When configuring virtual switch for EVPN, be aware of the following considerations:

- Due to default ARP policing, some of the ARP packets not destined for the device can be missed. This can lead to delayed ARP learning and synchronization.
- Clearing ARP for an EVPN can lead to inconsistency between the ARP table and the EVPN ARP table. To avoid this situation, clear both ARP and EVPN ARP tables.
- The `vlan-tag` can be configured for local switching. However, vlan-tagged VLANs should not be extended over the EVPN cloud.

Before you begin:

1. Configure the router interfaces.
2. Configure the router ID and autonomous system number for the device.
3. Enable chained composite next hop for EVPN.
4. Configure OSPF or any other IGP protocol.
5. Configure a BGP internal group.
6. Include the EVPN signaling network layer reachability information (NLRI) to the internal BGP group.
7. Configure RSVP or LDP.
8. Configure MPLS.
9. Create a label-switched path between the provider edge (PE) devices.

To configure the PE device:

1. Configure the virtual switch routing instance.

```
[edit routing-instances]
```

```
user@PE1# set evpn-instance instance-type virtual-switch
```

2. Configure the interface names for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance interface interface-name
```

3. Configure the route distinguisher for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance route-distinguisher route-distinguisher-value
```

4. Configure the VPN routing and forwarding (VRF) target community for the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vrf-target vrf-target
```

5. List the VLAN identifiers that are to be EVPN extended.

```
[edit routing-instances]
user@PE1# set evpn-instance protocols evpn extended-vlan-list [vlan-id-range]
```

6. Configure the bridge domain for the first virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance bridge-domains first-bridge-domain-name domain-type bridge
```

7. Assign the VLAN ID for the first bridge domain.

```
[edit routing-instances]
user@PE1# set evpn-instance bridge-domains first-bridge-domain-name vlan-id 10
```

8. Configure the IRB interface as the routing interface for the first bridge domain.

```
[edit routing-instances]
user@PE1# set evpn-instance bridge-domains first-bridge-domain-name routing-interface irb.0
```



9. Configure the interface name for the first bridge domain.

```
[edit routing-instances]
user@PE1# set evpn-instance bridge-domains first-bridge-domain-name bridge-options
interface CE-facing-interface
```

10. Configure the bridge domain for the second virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance bridge-domains second-bridge-domain-name domain-type bridge
```

11. Assign the VLAN ID for the second bridge domain.

```
[edit routing-instances]
user@PE1# set evpn-instance bridge-domains second-bridge-domain-name vlan-id VLAN-ID
```

12. Configure the IRB interface as the routing interface for the second bridge domain.

```
[edit routing-instances]
user@PE1# set evpn-instance bridge-domains second-bridge-domain-name routing-interface irb.1
```

13. Configure the interface name for the second bridge domain.

```
[edit routing-instances]
user@PE1# set evpn-instance bridge-domains second-bridge-domain-name bridge-options
interface CE-facing-interface
```

14. Configure the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance instance-type vrf
```

15. Configure the IRB interface as the routing interface for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance interface irb.0
user@PE1# set vrf-instance interface irb.1
```

16. Configure the route distinguisher for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance route-distinguisher route-distinguisher-value
```

17. Configure the VRF target community for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance vrf-target vrf-target
```

18. Configure the VRF label for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance vrf-table-label
```

19. Verify and commit the configuration.

For example:

```
[edit routing-instances]
user@PE1# set evpna instance-type virtual-switch
user@PE1# set evpna interface ge-0/1/4.0
user@PE1# set evpna interface ge-0/1/4.1
user@PE1# set evpna route-distinguisher 10.255.169.37:1
user@PE1# set evpna vrf-target target:100:1
user@PE1# set evpna protocols evpn extended-vlan-list [10 20]
user@PE1# set evpna bridge-domains bda domain-type bridge
user@PE1# set evpna bridge-domains bda vlan-id 10
user@PE1# set evpna bridge-domains bda routing-interface irb.0
user@PE1# set evpna bridge-domains bda bridge-options interface ge-0/1/4.0
user@PE1# set evpna bridge-domains bdb domain-type bridge
user@PE1# set evpna bridge-domains bdb vlan-id 20
user@PE1# set evpna bridge-domains bdb routing-interface irb.1
user@PE1# set evpna bridge-domains bdb bridge-options interface ge-0/1/4.1
user@PE1# set vrf instance-type vrf
user@PE1# set vrf interface irb.0
user@PE1# set vrf interface irb.1
user@PE1# set vrf route-distinguisher 192.0.2.1:2
```

```
user@PE1# set vrf vrf-target target:100:2
user@PE1# set vrf vrf-table-label
```

```
[edit]
user@PE1# commit
commit complete
```

## RELATED DOCUMENTATION

[Example: Configuring EVPN with Support for Virtual Switch | 1482](#)

## Example: Configuring EVPN with Support for Virtual Switch

### IN THIS SECTION

- [Example: Configuring EVPN with Support for Virtual Switch | 1482](#)

## Example: Configuring EVPN with Support for Virtual Switch

### IN THIS SECTION

- [Requirements | 1483](#)
- [Overview | 1483](#)
- [Configuration | 1484](#)
- [Verification | 1495](#)

This example shows how to configure a virtual switch in an Ethernet VPN (EVPN) deployment.

## Requirements

This example uses the following hardware and software components:

- Two MX Series 5G Universal Routing Platforms containing MPC FPCs.
- Two customer edge (CE) routers.
- Junos OS Release 14.1 or later.

Before you begin:

1. Configure the router interfaces.
2. Configure OSPF or any other IGP protocol.
3. Configure BGP.
4. Configure RSVP or LDP.
5. Configure MPLS.

## Overview

### IN THIS SECTION

- [Topology | 1484](#)

Starting with Junos OS Release 14.1, the Ethernet VPN (EVPN) solution on MX Series routers with MPC interfaces is extended to provide virtual switch support that enables multiple tenants with independent VLAN and subnet space within an EVPN instance. Virtual switch provides the ability to extend Ethernet VLANs over a WAN using a single EVPN instance while maintaining data-plane separation between the various VLANs associated with that instance. A single EVPN instance can stretch up to 4094 bridge domains defined in a virtual switch to remote sites.

When configuring virtual switch for EVPN, be aware of the following considerations:

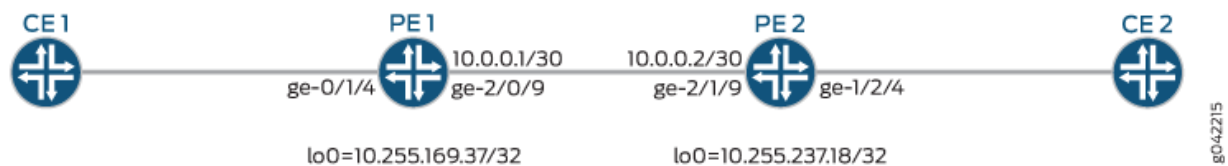
- Due to default ARP policing, some of the ARP packets not destined for the device can be missed. This can lead to delayed ARP learning and synchronization.
- Clearing ARP for an EVPN can lead to inconsistency between the ARP table and the EVPN ARP table. To avoid this situation, clear both ARP and EVPN ARP tables.

- The `vlan-tag` can be configured for local switching. However, vlan-tagged VLANs should not be extended over the EVPN cloud.

### Topology

Figure 147 on page 1484 illustrates a simple EVPN topology with virtual switch support. Routers PE1 and PE2 are the provider edge (PE) routers that connect to one customer edge (CE) router each – CE1 and CE2.

Figure 147: EVPN with Virtual Switch Support



### Configuration

#### IN THIS SECTION

- Procedure | 1484
- Results | 1492

### 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.

#### PE1

```

set interfaces ge-2/0/9 unit 0 family inet address 10.0.0.1/30
set interfaces ge-2/0/9 unit 0 family mpls
set interfaces ge-0/1/4 flexible-vlan-tagging

```

```

set interfaces ge-0/1/4 encapsulation flexible-ethernet-services
set interfaces ge-0/1/4 unit 0 family bridge interface-mode trunk
set interfaces ge-0/1/4 unit 0 vlan-id-list 10
set interfaces ge-0/1/4 unit 1 family bridge interface-mode trunk
set interfaces ge-0/1/4 unit 1 vlan-id-list 20
set interfaces irb unit 0 family inet address 192.168.1.1/16
set interfaces irb unit 1 family inet address 192.168.2.1/16
set interfaces lo0 unit 0 family inet address 10.255.169.37/32
set routing-options router-id 10.255.169.37
set routing-options autonomous-system 100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path PE1-to-PE2 from 10.255.169.37
set protocols mpls label-switched-path PE1-to-PE2 to 10.255.237.18
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.169.37
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 10.255.237.18
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances evpna instance-type virtual-switch
set routing-instances evpna interface ge-0/1/4.0
set routing-instances evpna interface ge-0/1/4.1
set routing-instances evpna route-distinguisher 10.255.169.37:1
set routing-instances evpna vrf-target target:100:1
set routing-instances evpna protocols evpn extended-vlan-list [10 20]
set routing-instances evpna bridge-domains bda domain-type bridge
set routing-instances evpna bridge-domains bda vlan-id 10
set routing-instances evpna bridge-domains bda routing-interface irb.0
set routing-instances evpna bridge-domains bda bridge-options interface ge-0/1/4.0
set routing-instances evpna bridge-domains bdb domain-type bridge
set routing-instances evpna bridge-domains bdb vlan-id 20
set routing-instances evpna bridge-domains bdb routing-interface irb.1
set routing-instances evpna bridge-domains bdb bridge-options interface ge-0/1/4.1
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf interface irb.1
set routing-instances vrf route-distinguisher 198.51.100.1:2

```

```
set routing-instances vrf vrf-target target:100:2
set routing-instances vrf vrf-table-label
```

## PE2

```
set interfaces ge-2/1/9 unit 0 family inet address 10.0.0.2/30
set interfaces ge-2/1/9 unit 0 family mpls
set interfaces ge-1/2/4 flexible-vlan-tagging
set interfaces ge-1/2/4 encapsulation flexible-ethernet-services
set interfaces ge-1/2/4 unit 0 family bridge interface-mode trunk
set interfaces ge-1/2/4 unit 0 vlan-id-list 10
set interfaces ge-1/2/4 unit 1 family bridge interface-mode trunk
set interfaces ge-1/2/4 unit 1 vlan-id-list 20
set interfaces irb unit 0 family inet address 192.168.2.2/16
set interfaces irb unit 1 family inet address 192.168.2.3/16
set interfaces lo0 unit 0 family inet address 10.255.237.18/32
set routing-options router-id 10.255.237.18
set routing-options autonomous-system 100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path PE2-to-PE1 from 10.255.237.18
set protocols mpls label-switched-path PE2-to-PE1 to 10.255.169.37
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.237.18
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 10.255.169.37
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances evpna instance-type virtual-switch
set routing-instances evpna interface ge-1/2/4.0
set routing-instances evpna interface ge-1/2/4.1
set routing-instances evpna route-distinguisher 10.255.237.18:1
set routing-instances evpna vrf-target target:100:1
set routing-instances evpna protocols evpn extended-vlan-list [10 20]
set routing-instances evpna bridge-domains bda domain-type bridge
set routing-instances evpna bridge-domains bda vlan-id 10
set routing-instances evpna bridge-domains bda routing-interface irb.0
set routing-instances evpna bridge-domains bda bridge-options interface ge-1/2/4.0
set routing-instances evpna bridge-domains bdb domain-type bridge
```

```

set routing-instances evpna bridge-domains bdb vlan-id 20
set routing-instances evpna bridge-domains bdb routing-interface irb.1
set routing-instances evpna bridge-domains bdb bridge-options interface ge-1/2/4.1
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf interface irb.1
set routing-instances vrf route-distinguisher 198.51.100.2:2
set routing-instances vrf vrf-target target:100:2
set routing-instances vrf vrf-table-label

```

## 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 Router PE1:



**NOTE:** Repeat this procedure for Router PE2, after modifying the appropriate interface names, addresses, and other parameters.

### 1. Configure the PE1 interfaces.

```

[edit interfaces]
user@PE1# set ge-2/0/9 unit 0 family inet address 10.0.0.1/30
user@PE1# set ge-2/0/9 unit 0 family mpls
user@PE1# set ge-0/1/4 flexible-vlan-tagging
user@PE1# set ge-0/1/4 encapsulation flexible-ethernet-services
user@PE1# set ge-0/1/4 unit 0 family bridge interface-mode trunk
user@PE1# set ge-0/1/4 unit 0 vlan-id-list 10
user@PE1# set ge-0/1/4 unit 1 family bridge interface-mode trunk
user@PE1# set ge-0/1/4 unit 1 vlan-id-list 20
user@PE1# set irb unit 0 family inet address 192.168.1.1/16
user@PE1# set irb unit 1 family inet address 192.168.2.1/16
user@PE1# set lo0 unit 0 family inet address 10.255.169.37/32

```



2. Set the router ID and autonomous system number for Router PE1.

```
[edit routing-options]
user@PE1# set router-id 10.255.169.37
user@PE1# set autonomous-system 100
```

3. Configure the chained composite next hop for EVPN.

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

4. Enable RSVP on all the interfaces of Router PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set rsvp interface all
user@PE1# set rsvp interface fxp0.0 disable
```

5. Create label-switched paths for PE1 to reach PE2.

```
[edit protocols]
user@PE1# set mpls label-switched-path PE1-to-PE2 from 10.255.169.37
user@PE1# set mpls label-switched-path PE1-to-PE2 to 10.255.237.18
```

6. Enable MPLS on all the interfaces of Router PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set mpls interface all
user@PE1# set mpls interface fxp0.0 disable
```

7. Configure the BGP group for Router PE1.

```
[edit protocols]
user@PE1# set bgp group ibgp type internal
```

8. Assign local and neighbor addresses to the ibgp BGP group for Router PE1 to peer with Router PE2.

```
[edit protocols]
user@PE1# set bgp group ibgp local-address 10.255.169.37
user@PE1# set bgp group ibgp neighbor 10.255.237.18
```

9. Include the EVPN signaling Network Layer Reachability Information (NLRI) to the ibgp BGP group.

```
[edit protocols]
user@PE1# set bgp group ibgp family evpn signaling
```

10. Configure OSPF on all the interfaces of Router PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set ospf area 0.0.0.0 interface all
user@PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
```

11. Configure the virtual switch routing instance.

```
[edit routing-instances]
user@PE1# set evpna instance-type virtual-switch
```

12. Configure the interface name for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna interface ge-0/1/4.0
user@PE1# set evpna interface ge-0/1/4.1
```

13. Configure the route distinguisher for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna route-distinguisher 10.255.169.37:1
```

14. Configure the VPN routing and forwarding (VRF) target community for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna vrf-target target:100:1
```

15. List the VLAN identifiers that are to be EVPN extended.

```
[edit routing-instances]
user@PE1# set evpna protocols evpn extended-vlan-list [10 20]
```

16. Configure the bridge domains for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna bridge-domains bda domain-type bridge
```

17. Assign the VLAN ID for the bda bridge domain.

```
[edit routing-instances]
user@PE1# set evpna bridge-domains bda vlan-id 10
```

18. Configure the IRB interface as the routing interface for the bda bridge domain.

```
[edit routing-instances]
user@PE1# set evpna bridge-domains bda routing-interface irb.0
```

19. Configure the interface name for the bda bridge domain.

```
[edit routing-instances]
user@PE1# set evpna bridge-domains bda bridge-options interface ge-0/1/4.0
```

20. Configure the bridge domains for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna bridge-domains bdb domain-type bridge
```

21. Assign the VLAN ID for the bdb bridge domain.

```
[edit routing-instances]
user@PE1# set evpna bridge-domains bdb vlan-id 20
```

22. Configure the IRB interface as the routing interface for the bda bridge domain.

```
[edit routing-instances]
user@PE1# set evpna bridge-domains bdb routing-interface irb.1
```

23. Configure the interface name for bdb bridge domain.

```
[edit routing-instances]
user@PE1# set evpna bridge-domains bdb bridge-options interface ge-0/1/4.1
```

24. Configure the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf instance-type vrf
```

25. Configure the IRB interface as the routing interface for the vrf routing instance.

```
[edit routing-instances]
user@PE1# set vrf interface irb.0
user@PE1# set vrf interface irb.1
```

26. Configure the route distinguisher for the vrf routing instance.

```
[edit routing-instances]
user@PE1# set vrf route-distinguisher 198.51.100.1:2
```

27. Configure the VRF target community for the vrf routing instance.

```
[edit routing-instances]
user@PE1# set vrf vrf-target target:100:2
```

## 28. Configure VRF label for the vrf routing instance.

```
[edit routing-instances]
user@PE1# set vrf vrf-table-label
```

### Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show routing-options`, `show protocols`, 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-2/0/9 {
 unit 0 {
 family inet {
 address 10.0.0.1/30;
 }
 family mpls;
 }
}
ge-0/1/4 {
 flexible-vlan-tagging;
 encapsulation flexible-ethernet-services;
 unit 0 {
 family bridge {
 interface-mode trunk;
 vlan-id-list 10;
 }
 }
 unit 1 {
 family bridge {
 interface-mode trunk;
 vlan-id-list 20;
 }
 }
}
irb {
 unit 0 {
 family inet {
 address 192.168.1.1/16;
 }
 }
}
```

```

 }
 unit 1 {
 family inet {
 address 192.168.2.1/16;
 }
 }
}
lo0 {
 unit 0 {
 family inet {
 address 10.255.169.37/32;
 }
 }
}
}

```

```

user@PE1# show routing-options
router-id 10.255.169.37;
autonomous-system 100;
forwarding-table {
 chained-composite-next-hop {
 ingress {
 evpn;
 }
 }
}

```

```

user@PE1# show protocols
rsvp {
 interface all;
 interface fxp0.0 {
 disable;
 }
}
mpls {
 label-switched-path PE1-to-PE2 {
 from 10.255.169.37;
 to 10.255.237.18;
 }
 interface all;
 interface fxp0.0 {

```

```

 disable;
 }
}
bgp {
 group ibgp {
 type internal;
 local-address 10.255.169.37;
 family evpn {
 signaling;
 }
 neighbor 10.255.237.18;
 }
}
ospf {
 area 0.0.0.0 {
 interface all;
 interface fxp0.0 {
 disable;
 }
 }
}
}

```

```

user@PE1# show routing-instances
evpna {
 instance-type virtual-switch;
 interface ge-0/1/4.0;
 interface ge-0/1/4.1;
 route-distinguisher 10.255.169.37:1;
 vrf-target target:100:1;
 protocols {
 evpn {
 extended-vlan-list [10 20];
 }
 }
}
bridge-domains {
 bda {
 domain-type bridge;
 vlan-id 10;
 routing-interface irb.0;
 bridge-options {
 interface ge-0/1/4.0;

```

```
 }
 }
 bdb {
 domain-type bridge;
 vlan-id 20;
 routing-interface irb.1;
 bridge-options {
 interface ge-0/1/4.1;
 }
 }
}
vrf {
 instance-type vrf;
 interface irb.0;
 interface irb.1;
 route-distinguisher 10.255.169.37:2;
 vrf-target target:100:2;
 vrf-table-label;
}
```

## Verification

### IN THIS SECTION

- [Verifying the Bridge Domain Configuration | 1495](#)
- [Verifying MAC Table Routes | 1496](#)
- [Verifying the Bridge EVPN Peer Gateway MAC | 1497](#)

Confirm that the configuration is working properly.

### *Verifying the Bridge Domain Configuration*

## Purpose

Verify the bridge domain configuration for the evpna routing instance.



## Action

From operational mode, run the `show bridge domain extensive` command.

```
user@PE1> show bridge domain extensive
Routing instance: evpna
Bridge domain: bda State: Active
Bridge VLAN ID: 10 EVPN extended: Yes
Interfaces:
 ge-0/1/4.0
 pip-10.000010000000
 pip-10.fe0f0f000000
Total MAC count: 2

Bridge domain: bdb State: Active
Bridge VLAN ID: 20 EVPN extended: Yes
Interfaces:
 ge-0/1/4.1
 pip-11.010010000000
 pip-11.ffff0f000000
Total MAC count: 2
```

## Meaning

The configured bridge domains `bda` and `bdb` and their associated VLAN IDs and interfaces are displayed. The bridge domains are also extended with EVPN.

### *Verifying MAC Table Routes*

## Purpose

Verify the MACs learned in the data plane and control plane.

## Action

From operational mode, run the `show bridge mac-table` command.

```
user@PE1> show bridge 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 : evpna
Bridging domain : bda, VLAN : 10
 MAC MAC Logical NH RTR
 address flags interface Index ID
 00:00:00:aa:01:01 S ge-0/1/4.0
 00:00:00:bb:01:01 DC
 00:00:00:cc:01:01 DC 1048574 1048574
 1048576 1048576

Bridging domain : bdb, VLAN : 20
 MAC MAC Logical NH RTR
 address flags interface Index ID
 00:00:00:aa:02:01 S ge-0/1/4.1
 00:00:00:bb:02:01 DC 1048575 1048575
 00:00:00:cc:02:01 DC 1048577 1048577

```

## Meaning

The configured static MACs for the bridge domains are displayed.

### *Verifying the Bridge EVPN Peer Gateway MAC*

## Purpose

Verify the bridge EVPN peer gateway MAC for the evpna routing instance.

## Action

From operational mode, run the `show bridge evpn peer-gateway-macs` command.

```

user@PE1> show bridge evpn peer-gateway-macs
Routing instance : evpna
Bridging domain : bda, VLAN : 10
 Installed GW MAC addresses:
 00:23:9c:96:af:f0
 a8:d0:e5:5b:02:08

Bridging domain : bdb, VLAN : 20
 Installed GW MAC addresses:

```

```
00:23:9c:96:af:f0
a8:d0:e5:5b:02:08
```

### Meaning

The gateway MACs of the EVPN peers for the evpna routing instance are displayed.

### SEE ALSO

| [Virtual Switch Support for EVPN Overview](#) | 718

# Configuring Integrated Bridging and Routing

## IN THIS CHAPTER

- [EVPN with IRB Solution Overview | 1499](#)
- [An EVPN with IRB Solution on EX9200 Switches Overview | 1505](#)
- [Anycast Gateways | 1511](#)
- [Configuring EVPN with IRB Solution | 1515](#)
- [Configuring an EVPN with IRB Solution on EX9200 Switches | 1519](#)
- [Example: Configuring EVPN with IRB Solution | 1522](#)
- [Example: Configuring an EVPN with IRB Solution on EX9200 Switches | 1547](#)

## EVPN with IRB Solution Overview

### IN THIS SECTION

- [Need for an EVPN IRB Solution | 1500](#)
- [Implementing the EVPN IRB Solution | 1501](#)
- [Benefits of Implementing the EVPN IRB Solution | 1503](#)

A Data Center Service Provider (DCSP) hosts the data center for its multiple customers on a common physical network. To each customer (also called a tenant), the service looks like a full-fledged data center that can expand to 4094 VLANs and all private subnets. For disaster recovery, high availability, and optimization of resource utilization, it is common for the DCSP to span across the data center to more than one site. To deploy the data center services, a DCSP faces the following main challenges:

- Extending Layer 2 domains across more than one data center site. This requires optimal intra-subnet traffic forwarding.

- Supporting optimal inter-subnet traffic forwarding and optimal routing in the event of virtual machine (VM).
- Supporting multiple tenants with independent VLAN and subnet space.

Ethernet VPN (EVPN) is targeted to handle all of the above mentioned challenges, wherein:

- The basic EVPN functionality enables optimal intra-subnet traffic forwarding
- Implementing the integrated routing and bridging (IRB) solution in an EVPN deployment enables optimal inter-subnet traffic forwarding
- Configuring EVPN with virtual switch support enables multiple tenants with independent VLAN and subnet space

The following sections describe the IRB solution for EVPNs:

## Need for an EVPN IRB Solution

EVPN is a technology used to provide Layer 2 extension and interconnection across an IP/MPLS core network to different physical sites belonging to a single Layer 2 domain. In a data center environment with EVPN, there is a need for both Layer 2 (intra-subnet traffic) and Layer 3 (inter-subnet traffic) forwarding and potentially interoperation with tenant Layer 3 VPNs.

With only a Layer 2 solution, there is no optimum forwarding of inter-subnet traffic, even when the traffic is local, for instance, when both the subnets are on the same server.

With only a Layer 3 solution, the following issues for intra-subnet traffic can arise:

- MAC address aliasing issue where duplicate MAC addresses are not detected.
- TTL issue for applications that use TTL 1 to confine traffic within a subnet.
- IPv6 link-local addressing and duplicate address detection that relies on Layer 2 connectivity.
- Layer 3 forwarding does not support the forwarding semantics of a subnet broadcast.
- Support of non-IP applications that require Layer 2 forwarding.

Because of the above mentioned shortcomings of a pure Layer 2 and Layer 3 solution, there is a need for a solution incorporating optimal forwarding of both Layer 2 and Layer 3 traffic in the data center environment when faced with operational considerations such as Layer 3 VPN interoperability and virtual machine (VM) mobility.

An EVPN-based integrated routing and bridging (IRB) solution provides optimum unicast and multicast forwarding for both intra-subnets and inter-subnets within and across data centers.

The EVPN IRB feature is useful for service providers operating in an IP/MPLS network that provides both Layer 2 VPN or VPLS services and Layer 3 VPN services who want to extend their service to provide cloud computation and storage services to their existing customers.

## Implementing the EVPN IRB Solution

An EVPN IRB solution provides the following:

- Optimal forwarding for intra-subnet (Layer 2) traffic.
- Optimal forwarding for inter-subnet (Layer 3) traffic.
- Support for ingress replication for multicast traffic.
- Support for network-based as well as host-based overlay models.
- Support for consistent policy-based forwarding for both Layer 2 and Layer 3 traffic.
- Support for the following routing protocols on the IRB interface:
  - BFD
  - BGP
  - IS-IS
  - OSPF and OSPF version 3
- Support for single-active and all-active multihoming

Junos OS supports several models of EVPN configuration to satisfy the individual needs of EVPN and data center cloud services customers. To provide flexibility and scalability, multiple bridge domains can be defined within a particular EVPN instance. Likewise, one or more EVPN instances can be associated with a single Layer 3 VPN virtual routing and forwarding (VRF). In general, each data center tenant is assigned a unique Layer 3 VPN VRF, while a tenant could comprise one or more EVPN instances and one or more bridge domains per EVPN instance. To support this model, each configured bridge domain (including the default bridge domain for an EVPN instance) requires an IRB interface to perform the Layer 2 and Layer 3 functions. Each bridge domain or IRB interface maps to a unique IP subnet in the VRF.



**NOTE:** You can associate an IRB interface with the primary instance inet.0 table instead of a VRF in an EVPN IRB solution.

There are two major functions that are supported for IRB in EVPN.

- Host MAC-IP synchronization

This includes:

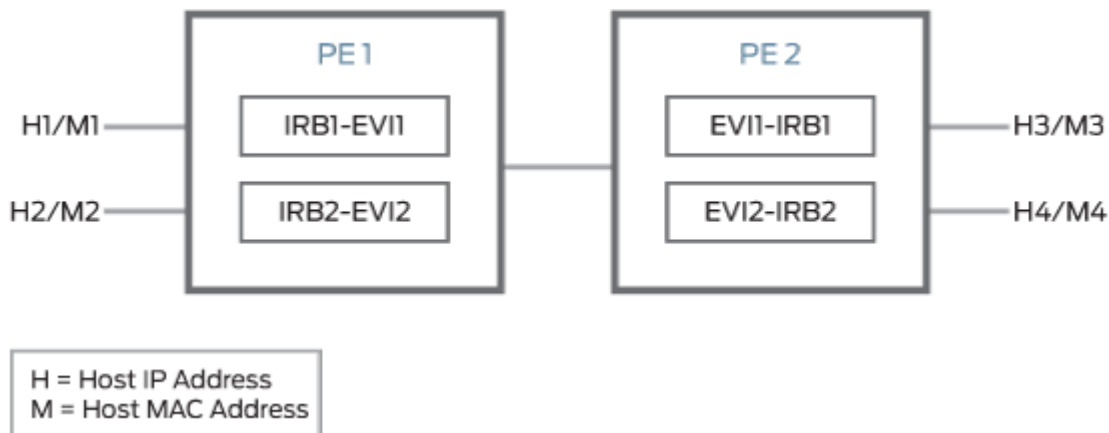
- Advertising the IP address along with the MAC advertisement route in EVPN. This is done by using the IP field in the EVPN MAC advertisement route.
- The receiving PE router installs MAC into the EVPN instance (EVI) table and installs IP into the associated VRF.
- Gateway MAC-IP synchronization

This includes:

- Advertising all local IRB MAC and IP addresses in an EVPN. This is achieved by including the default gateway extended community in the EVPN MAC advertisement route.
- The receiving PE creates a forwarding state to route packets destined for the gateway MAC, and a proxy ARP is done for the gateway IP with the MAC advertised in the route.

Figure 148 on page 1502 illustrates the inter-subnet traffic forwarding between two provider edge (PE) devices – PE1 and PE2. The IRB1 and IRB2 interfaces on each PE device belong to a different subnet, but they share a common VRF.

**Figure 148: Inter-Subnet Traffic Forwarding**



The inter-subnet traffic forwarding is performed as follows:

1. PE2 advertises H3-M3 and H4-M4 binding to PE1. Similarly PE1 advertises H1-M1 and H2-M2 binding to PE2.
2. PE1 and PE2 install the MAC address in the corresponding EVI MAC table, whereas the IP routes are installed in the shared VRF.
3. The advertising PE device is set as the next hop for the IP routes.

4. If H1 sends packets to H4, the packets are sent to IRB1 on PE1.
5. IP lookup for H4 happens in the shared VRF on PE1. Because the next hop for the H4 IP is PE2 (the advertising PE), an IP unicast packet is sent to PE2.
6. PE1 rewrites the MAC header based on the information in the VRF route, and PE2 performs a MAC lookup to forward the packet to H4.

## Benefits of Implementing the EVPN IRB Solution

The main goal of the EVPN IRB solution is to provide optimal Layer 2 and Layer 3 forwarding. The solution is required to efficiently handle inter-subnet forwarding as well as virtual machine (VM) mobility. VM mobility refers to the ability of a VM to migrate from one server to another within the same or a different data center while retaining its existing MAC and IP address. Providing optimal forwarding for inter-subnet traffic and effective VM mobility involves solving two problems – the default gateway problem and the triangular routing problem.

IPv6 addresses are supported on IRB interfaces with EVPN using the Neighbor Discovery Protocol (NDP). The following capabilities are introduced for IPv6 support with EVPN:

- IPv6 addresses on IRB interfaces in primary routing instances
- Learning IPv6 neighborhood from solicited NA message
- NS and NA packets on the IRB interfaces are disabled from network core
- Virtual gateway addresses are used as Layer 3 addresses
- Host MAC-IP synchronization for IPv6

You can configure the IPv6 addresses in the IRB interface at the `[edit interfaces irb]` hierarchy level.

## Gateway MAC and IP Synchronization

In an EVPN IRB deployment, the IP default gateway for a VM is the IP address configured on the IRB interface of the provider edge (PE) router corresponding to the bridge domain or VLAN of which the VM is a member. The default gateway problem arises because a VM does not flush its ARP table when relocating from one server to another and continues sending packets with the destination MAC address set to that of the original gateway. If the old and new servers are not part of the same Layer 2 domain (the new Layer 2 domain could be within the current data center or a new data center), the gateway previously identified is no longer the optimal or local gateway. The new gateway needs to identify packets containing the MAC addresses of other gateways on remote PE routers and forward the traffic as if the packets were destined to the local gateway itself. At the minimum, this functionality requires each PE router to advertise its gateway or IRB MAC and IP addresses to all other PE routers in the network. The gateway address exchange can be accomplished using the standard MAC route



advertisement message (including the IP address parameter) and tagging that route with the default gateway extended community so that the remote PE routers can distinguish the gateway MAC advertisement routes from normal MAC advertisement routes.



**NOTE:** We require that you configure the `no-gateway-community` option at the `[edit routing-instances EVPN-instance-name protocols evpn default-gateway]` hierarchy level in each EVPN routing instance in which you configure an IRB interface with a virtual gateway address. On platforms that support configuring the `no-gateway-community` option at the global level, you can alternatively configure `no-gateway-community` at the `[edit protocols evpn default-gateway]` hierarchy level.

See *default-gateway* for details on using the `no-gateway-community` option.

## Layer 3 VPN Interworking

The inter-data center aspect of the EVPN IRB solution involves routing between VMs that are present in different data centers or routing between a host site completely outside of the data center environment and a VM within a data center. This solution relies on the ability of EVPN MAC route advertisements to carry both MAC address and IP address information. The local MAC learning functionality of the PE router is extended to also capture IP address information associated with MAC addresses learned locally. That IP-MAC address mapping information is then distributed to each PE router through normal EVPN procedures. When a PE router receives such MAC and IP information, it installs the MAC route in the EVPN instance as well as a host route for the associated IP address in the Layer 3 VPN VRF corresponding to that EVPN instance. When a VM moves from one data center to another, normal EVPN procedures result in the MAC and IP address being advertised from the new PE router which the VM resides behind. The host route installed in the VRF associated with an EVPN solicits Layer 3 traffic destined to that VM to the new PE router and avoids triangular routing between the source, the former PE router the VM resided behind, and the new PE router.

BGP scalability is a potential concern with the inter-data center triangular routing avoidance solution because of the potential for injection of many host routes into Layer 3 VPN. With the method previously described, in the worst case there is an IP host route for each MAC address learned through the local EVPN MAC learning procedures or through a MAC advertisement message received from a remote PE router. BGP route target filtering can be used to limit distribution of such routes.

The following functional elements are required to implement the inter-data center triangular routing avoidance using Layer 3 inter-subnet forwarding procedures:

1. The source host sends an IP packet using its own source MAC and IP address with the destination MAC of the IRB interface of the local PE router and the IP address of the destination host.
2. When the IRB interface receives the frame with its MAC as the destination, it performs a Layer 3 lookup in the VRF associated with the EVPN instance to determine where to route the packet.

3. In the VRF, the PE router finds the Layer 3 route derived from a MAC plus an IP EVPN route received from the remote PE router earlier. The destination MAC address is then changed to the destination MAC address corresponding to the destination IP.
4. The packet is then forwarded to the remote PE router serving the destination host using MPLS, using the label corresponding to the EVPN instance of which the destination host is a member.
5. The egress PE router receiving the packet performs a Layer 2 lookup for the destination host's MAC and sends the packet to the destination host on the attached subnet via the egress PE router's IRB interface.
6. Because the ingress PE router is performing Layer 3 routing, the IP TTL is decremented.

## RELATED DOCUMENTATION

[EVPN Multihoming Overview | 162](#)

[Understanding VXLANs | 606](#)

[Example: Configuring EVPN-MPLS with IRB Solution | 1528](#)

## An EVPN with IRB Solution on EX9200 Switches Overview

### IN THIS SECTION

- [Need for an EVPN IRB Solution | 1506](#)
- [Implementing the EVPN IRB Solution | 1507](#)
- [Benefits of Implementing the EVPN IRB Solution | 1508](#)
- [IPv6 Support for IRB Interfaces with EVPN Using Neighborhood Discovery Protocol \(NDP\) | 1510](#)

A data center service provider (DCSP) hosts the data center for its multiple customers on a common physical network. To each customer (also called a tenant), the service looks like a full-fledged data center that can expand to 4094 VLANs and all private subnets. For disaster recovery, high availability, and optimization of resource utilization, it is common for the DCSP to span the data center over multiple sites. To deploy data center services, a DCSP faces the following main challenges:

- Extending Layer 2 domains across more than one data center site. This requires optimal intra-subnet traffic forwarding.

- Supporting optimal inter-subnet traffic forwarding and optimal routing in the event of virtual machine (VM) motion.
- Supporting multiple tenants with independent VLAN and subnet space.

Ethernet VPN (EVPN) is targeted to handle all the preceding challenges, wherein:

- The basic EVPN functionality enables optimal intra-subnet traffic forwarding
- Implementing the integrated routing and bridging (IRB) solution in an EVPN deployment enables optimal inter-subnet traffic forwarding
- Configuring EVPN with virtual switch support enables multiple tenants with independent VLAN and subnet space

The following sections describe the integrated routing and bridging (IRB) solution for EVPNs:

## Need for an EVPN IRB Solution

EVPN is a technology used to provide Layer 2 extension and interconnection across an IP/MPLS core network to different physical sites belonging to a single Layer 2 domain. In a data center environment with EVPN, there is a need for both Layer 2 (intra-subnet traffic) and Layer 3 (inter-subnet traffic) forwarding and potentially interoperation with tenant Layer 3 VPNs.

With only a Layer 2 solution, there is no optimum forwarding of inter-subnet traffic, even when the traffic is local, for instance, when both the subnets are on the same server.

With only a Layer 3 solution, the following issues for intra-subnet traffic can arise:

- MAC address aliasing issue where duplicate MAC addresses are not detected.
- TTL issue for applications that use TTL 1 to confine traffic within a subnet.
- IPv6 link-local addressing and duplicate address detection that relies on Layer 2 connectivity.
- Layer 3 forwarding does not support the forwarding semantics of a subnet broadcast.
- Support of non-IP applications that require Layer 2 forwarding.

Because of the above mentioned shortcomings of a pure Layer 2 and Layer 3 solution, there is a need for a solution incorporating optimal forwarding of both Layer 2 and Layer 3 traffic in the data center environment when faced with operational considerations such as Layer 3 VPN interoperability and virtual machine (VM) mobility.

An EVPN-based integrated routing and bridging (IRB) solution provides optimum unicast and multicast forwarding for both intra-subnets and inter-subnets within and across data centers.

The EVPN IRB feature is useful for service providers operating in an IP/MPLS network that provides both Layer 2 VPN or VPLS services and Layer 3 VPN services who want to extend their service to provide cloud computation and storage services to their existing customers.

## Implementing the EVPN IRB Solution

An EVPN IRB solution provides the following:

- Optimal forwarding for intra-subnet (Layer 2) traffic.
- Optimal forwarding for inter-subnet (Layer 3) traffic.
- Support for ingress replication for multicast traffic.
- Support for network-based as well as host-based overlay models.
- Support for consistent policy-based forwarding for both Layer 2 and Layer 3 traffic.

Junos OS supports several models of EVPN configuration to satisfy the individual needs of EVPN and data center cloud services customers. To provide flexibility and scalability, multiple VLANs can be defined within a particular EVPN instance. Likewise, one or more EVPN instances can be associated with a single Layer 3 VPN virtual routing and forwarding (VRF). In general, each data center tenant is assigned a unique Layer 3 VPN VRF, while a tenant could comprise one or more EVPN instances and one or more VLANs per EVPN instance. To support this model, each configured VLAN (including the default VLAN for an EVPN instance) requires an IRB interface to perform the Layer 2 and Layer 3 functions. Each VLAN or IRB interface maps to a unique IP subnet in the VRF.

There are two major functions that are supported for IRB in EVPN.

- Host MAC-IP synchronization

This includes:

- Advertising the IP address along with the MAC advertisement route in EVPN. This is done by using the IP field in the EVPN MAC advertisement route.
- The receiving PE router installs MAC into the EVPN instance (EVI) table and installs IP into the associated VRF.

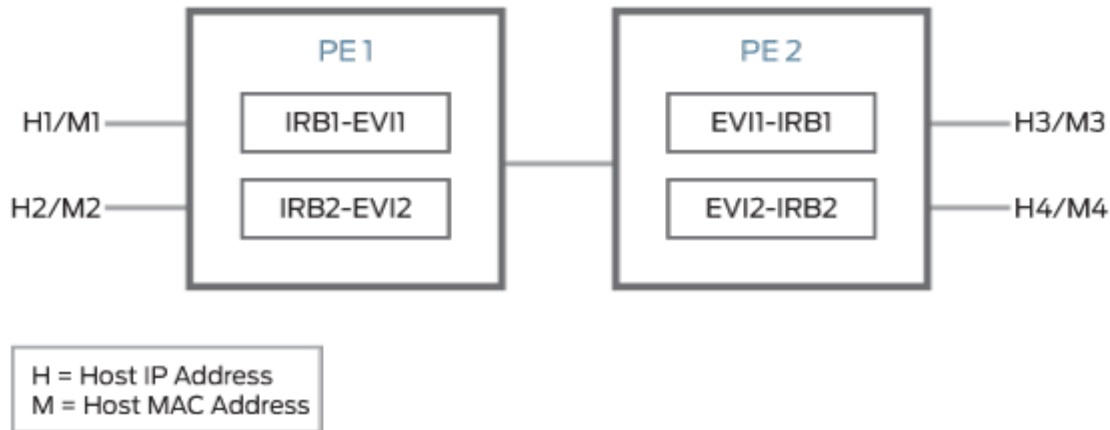
- Gateway MAC-IP synchronization

This includes:

- Advertising all local IRB MAC and IP addresses in an EVPN. This is achieved by including the default gateway extended community in the EVPN MAC advertisement route.
- The receiving PE creates a forwarding state to route packets destined for the gateway MAC, and a proxy ARP is done for the gateway IP with the MAC advertised in the route.

Figure 149 on page 1508 illustrates the inter-subnet traffic forwarding between two provider edge (PE) devices—PE1 and PE2. The IRB1 and IRB2 interfaces on each PE device belong to a different subnet, but they share a common VRF.

**Figure 149: Inter-Subnet Traffic Forwarding**



The inter-subnet traffic forwarding is performed as follows:

1. PE2 advertises the H3-M3 and H4-M4 binding to PE1. Similarly, PE1 advertises the H1-M1 and H2-M2 binding to PE2.
2. PE1 and PE2 install the MAC address in the corresponding EVI MAC table, whereas the IP routes are installed in the shared VRF.
3. The advertising PE device is set as the next hop for the IP routes.
4. If H1 sends packets to H4, the packets are sent to IRB1 on PE1.
5. IP lookup for H4 happens in the shared VRF on PE1. Because the next hop for the H4 IP is PE2 (the advertising PE), an IP unicast packet is sent to PE2.
6. PE1 rewrites the MAC header based on the information in the VRF route, and PE2 performs a MAC lookup to forward the packet to H4.

### Benefits of Implementing the EVPN IRB Solution

The main goal of the EVPN IRB solution is to provide optimal Layer 2 and Layer 3 forwarding. The solution is required to efficiently handle inter-subnet forwarding as well as virtual machine (VM) mobility. VM mobility refers to the ability of a VM to migrate from one server to another within the same or a different data center while retaining its existing MAC and IP address. Providing optimal forwarding for inter-subnet traffic and effective VM mobility involves solving two problems – the default gateway problem and the triangular routing problem.

## Gateway MAC and IP Synchronization

In an EVPN IRB deployment, the IP default gateway for a VM is the IP address configured on the IRB interface of the provider edge (PE) router corresponding to the VLAN of which the VM is a member. The default gateway problem arises because a VM does not flush its ARP table when relocating from one server to another and continues sending packets with the destination MAC address set to that of the original gateway. If the old and new servers are not part of the same Layer 2 domain (the new Layer 2 domain could be within the current data center or a new data center), the gateway previously identified is no longer the optimal or local gateway. The new gateway needs to identify packets containing the MAC addresses of other gateways on remote PE routers and forward the traffic as if the packets were destined to the local gateway itself. At minimum, this functionality requires each PE router to advertise its gateway or IRB MAC and IP addresses to all other PE routers in the network. The gateway address exchange can be accomplished using the standard MAC route advertisement message (including the IP address parameter) and tagging that route with the default gateway extended community so that the remote PE routers can distinguish the gateway MAC advertisement routes from normal MAC advertisement routes.

## Layer 3 VPN Interworking

The inter-data center aspect of the EVPN IRB solution involves routing between VMs that are present in different data centers or routing between a host site completely outside of the data center environment and a VM within a data center. This solution relies on the ability of EVPN MAC route advertisements to carry both MAC address and IP address information. The local MAC learning functionality of the PE router is extended to also capture IP address information associated with MAC addresses learned locally. That IP-MAC address mapping information is then distributed to each PE router through normal EVPN procedures. When a PE router receives such MAC and IP information, it installs the MAC route in the EVPN instance as well as a host route for the associated IP address in the Layer 3 VPN VRF corresponding to that EVPN instance. When a VM moves from one data center to another, normal EVPN procedures result in the MAC and IP address being advertised from the new PE router which the VM resides behind. The host route installed in the VRF associated with an EVPN solicits Layer 3 traffic destined to that VM to the new PE router and avoids triangular routing between the source, the former PE router the VM resided behind, and the new PE router.

BGP scalability is a potential concern with the inter-data center triangular routing avoidance solution because of the potential for injection of many host routes into Layer 3 VPN. With the method previously described, in the worst case there is an IP host route for each MAC address learned through the local EVPN MAC learning procedures or through a MAC advertisement message received from a remote PE router. BGP route target filtering can be used to limit distribution of such routes.

The following functional elements are required to implement the inter-data center triangular routing avoidance using Layer 3 inter-subnet forwarding procedures:

1. The source host sends an IP packet using its own source MAC and IP address with the destination MAC of the IRB interface of the local PE router and the IP address of the destination host.

2. When the IRB interface receives the frame with its MAC as the destination, it performs a Layer 3 lookup in the VRF associated with the EVPN instance to determine where to route the packet.
3. In the VRF, the PE router finds the Layer 3 route derived from a MAC plus an IP EVPN route received from the remote PE router earlier. The destination MAC address is then changed to the destination MAC address corresponding to the destination IP.
4. The packet is then forwarded to the remote PE router serving the destination host using MPLS, using the label corresponding to the EVPN instance of which the destination host is a member.
5. The egress PE router receiving the packet performs a Layer 2 lookup for the destination host's MAC and sends the packet to the destination host on the attached subnet via the egress PE router's IRB interface.
6. Because the ingress PE router is performing Layer 3 routing, the IP TTL is decremented.

### IPv6 Support for IRB Interfaces with EVPN Using Neighborhood Discovery Protocol (NDP)

IPv6 addresses are supported on IRB interfaces with EVPN using NDP. The following capabilities are introduced for IPv6 support with EVPN:

- IPv6 addresses on IRB interfaces in primary routing instances
- Learning IPv6 neighbors from solicited neighbor advertisement (NA) messages
- Neighbor solicitation (NS) and neighbor advertisement (NA) packets on the IRB interfaces are disabled from network core
- Virtual gateway addresses are used as Layer 3 addresses
- Host MAC-IP synchronization for IPv6

You can configure the IPv6 addresses in the IRB interface at the `[edit interfaces irb]` hierarchy level.

#### 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                                                                                    |
|---------|------------------------------------------------------------------------------------------------|
| 17.3R1  | Starting in Junos OS Release 17.3R1, IPv6 addresses are supported on IRB interfaces with EVPN. |

## RELATED DOCUMENTATION

[Example: Configuring an EVPN with IRB Solution on EX9200 Switches | 1547](#)

[EVPN Overview for Switches | 1432](#)

## Anycast Gateways

### IN THIS SECTION

- [Benefits of Anycast Gateways | 1512](#)
- [Anycast Gateway Configuration Guidelines | 1512](#)
- [Anycast Gateway Configuration Limitations | 1514](#)

In an EVPN-MPLS environment with two Juniper Networks devices multihomed in all-active mode, you can configure IRB interfaces on the devices. With the IRB interfaces in place, the multihomed devices function as gateways that handle intersubnet routing. To set up an IRB interface on a Juniper Networks device, you can configure the following:

- An IRB interface with:
  - An IPv4 or an IPv6 address

```
set interface irb unit logical-unit-number family inet ipv4-address
set interface irb unit logical-unit-number family inet6 ipv6-address
```

- A media access control (MAC) address

```
set interface irb unit logical-unit-number mac mac-address
```



**NOTE:** In addition to explicitly configuring a MAC address using the above command syntax, you can use the MAC address that the Juniper Networks device automatically generates (chassis MAC).

- A virtual gateway address (VGA) with:



- An IPv4 or an IPv6 address

```
set interfaces irb unit logical-unit-number family inet address primary-ipv4-address/8
virtual-gateway-address gateway-ipv4-address
set interfaces irb unit logical-unit-number family inet6 address primary-ipv6-address/104
virtual-gateway-address gateway-ipv6-address
```

- A MAC address

```
set interface irb unit logical-unit-number virtual-gateway-v4-mac mac-address
set interface irb unit logical-unit-number virtual-gateway-v6-mac mac-address
```



**NOTE:** In addition to explicitly configuring a MAC address using the above command syntax, you can use the MAC address that the Juniper Networks device automatically generates (chassis MAC).

When specifying an IP or MAC address for an IRB interface or VGA on the multihomed devices, you can now use an anycast address. This support of anycast addresses enables you to configure the same addresses for the IRB interface or VGA on each of the multihomed devices, thereby establishing the devices as anycast gateways.

Your IP address subnet scheme will determine whether you use the IRB interface command syntax or the VGA command syntax to set up your anycast gateway.

In an Ethernet VPN–Multiprotocol Label Switching (EVPN–MPLS) environment, you can configure two Juniper Networks devices multihomed in all-active mode as anycast gateways.

The following sections provide more information about anycast gateways.

## Benefits of Anycast Gateways

- With the two multihomed Juniper Networks devices acting as anycast gateways in an EVPN–MPLS network, a host in the same network that generates Layer 3 packets with destinations in other networks can now send the packets to the local anycast gateway. Upon receipt of these Layer 3 packets, the anycast gateway routes the packets in the core network based on destination IP lookup.

## Anycast Gateway Configuration Guidelines

- In general, when configuring addresses for an anycast gateway:
  - For IPv4 or IPv6 addresses, you can specify any subnet.

- For MAC addresses, you can use the MAC address that the Juniper Networks device automatically generates (chassis MAC), or you can explicitly configure a MAC address using the CLI.
- Your IP address subnet scheme will determine whether you use the IRB interface command syntax or the VGA command syntax to set up your anycast gateway.

To set up your multihomed devices as anycast gateways, we provide the following configuration guidelines:

- Guideline 1—If the IP address for the anycast gateways is in the /30 or /31 (for IPv4) or /126 or /127 (for IPv6) subnet:
- You must configure the same IP address for the IRB interface on each of the multihomed devices using one of the following commands.

```
set interface irb unit logical-unit-number family inet ipv4-address
set interface irb unit logical-unit-number family inet6 ipv6-address
```

- You must explicitly configure the MAC address using the following command:

```
set interface irb unit logical-unit-number mac mac-address
```

- You must not configure a VGA (IP and MAC addresses).
- Guideline 2—If the IP address for the anycast gateways is a subnet other than /30, /31, /126, or /127 then a VGA can be configured:
- You must configure the same IP address for the VGA on each of the multihomed devices using one of the following commands.

```
set interfaces irb unit logical-unit-number family inet address primary-ipv4-address/8
virtual-gateway-address gateway-ipv4-address
set interfaces irb unit logical-unit-number family inet6 address primary-ipv6-address/104
virtual-gateway-address gateway-ipv6-address
```

- You must explicitly configure the MAC address using one of the following commands:

```
set interface irb unit logical-unit-number virtual-gateway-v4-mac
set interface irb unit logical-unit-number virtual-gateway-v6-mac mac-address
```

- When specifying a MAC address for the VEA, we do not recommend using the same MAC address used for VRRP.



**NOTE:** You can also see ["Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Fabric with an Anycast Gateway"](#) on page 845 (["Overview and Topology"](#) on page 847 section) for similar guidelines to configure a leaf device as an anycast gateway in an EVPN-VXLAN edge-routed bridging (ERB) overlay fabric.

## Anycast Gateway Configuration Limitations

When configuring the anycast gateway using guidelines described earlier in this topic, keep the following in mind:

- In general, we do not recommend reusing a VRRP MAC address as a MAC address for an IRB interface. However, if you must do so, as is the general practice when configuring VRRP on Juniper Networks devices, you must use a VRRP IPv4 MAC address for the IPv4 family and a VRRP IPv6 MAC address for the IPv6 family.

Given these parameters, the only configuration guideline with which this limitation will work is configuration guideline 2.

- When configuring anycast gateway addresses using guideline 1 in an EVPN-MPLS environment, you must also specify the `default-gateway do-not-advertise` configuration statements within a routing instance. For example:

```
set routing-instance routing-instance-name protocols evpn default-gateway do-not-advertise
```

- In an EVPN-MPLS environment, if your anycast gateway IP addresses are in different subnets and you specify the addresses within multiple routing instances:
- If you configured an anycast gateway IP address using configuration guideline 1 in one routing instance, and another anycast gateway IP address using configuration guideline 2 in a different routing instance, you must also specify the `default-gateway no-gateway-community` configuration statements within the routing instance:

```
set routing-instance routing-instance-name protocols evpn default-gateway no-gateway-community
```

This additional configuration applies only to the routing instance that includes anycast gateway IP addresses configuring using guideline 1.

- For each routing instance in which you specified the anycast gateway IP address using configuration guideline 1, we recommend specifying a single non-VRRP MAC address.
- Automatic ESI generation is enabled by default on devices with EVPN multihoming for virtual gateway redundancy. We recommend that you disable the automatic ESI generation for EVPN-VXLAN networks with edge-routed bridging (ERB) overlays. In that case, you can include the `no-auto-virtual-gateway-esi` statement at the `[edit interfaces irb unit logical-unit-number]` hierarchy level.

Starting in Junos OS Release 22.1R1, MX960, MX2020, and MX10008 routers also enable automatic ESI generation by default for EVPN Layer 3 gateway IRB interface ESIs. However, the `no-auto-virtual-gateway-esi` statement is not supported with EVPN-MPLS networks. As a result, you will always see auto-generated ESIs for IRB interfaces in this case.

- In an EVPN-VXLAN environment with multihoming, you might use multiple EVPN routing instances on peer provider edge (PE) devices that share an Ethernet segment (ES). When you configure anycast gateways with the `default-gateway` statement, we don't support mixing the default behavior (advertise option) with the `no-gateway-community` option on the links that participate in the same ES.

As a result, if you configure the `default-gateway` statement with the `no-gateway-community` option in any EVPN routing instances on any peer PE device that share an ES, you must configure this statement:

- In all the routing instances that share the ES on a PE device,
- On all the peer PE devices that share the ES
- Only with either the `no-gateway-community` option or the `do-not-advertise`.

You can't omit setting the `default-gateway` statement or include the statement with the `advertise` option in any routing instance on any peer PE device.

- We support setting an anycast gateway IP address on IRB interfaces on ACX5448 devices. However, for IRB interfaces with /30 or /31 IP addresses on connections between PE and customer edge (CE) device interfaces, the CE device doesn't have enough pool space for the BGP session IP address allocation. As a result, we don't support BGP with IRB interface /30 and /31 anycast IP addresses.

## Configuring EVPN with IRB Solution

You can configure an Ethernet VPN (EVPN) with IRB solution to enable Layer 2 switching and Layer 3 routing operations within a single node, thus avoiding extra hops for inter-subnet traffic. The EVPN IRB solution eliminates the default gateway problem using the gateway MAC and IP synchronization, and avoids the triangular routing problem with Layer 3 interworking by creating IP host routes for virtual machines (VMs) in the tenant virtual routing and forwarding (VRF) routing instances.

Before you begin:

1. Configure the router interfaces.
2. Configure the router ID and autonomous system number for the device.
3. Enable chained composite next hop for EVPN.
4. Configure OSPF or any other IGP protocol.
5. Configure a BGP internal group.
6. Include the EVPN signaling network layer reachability information (NLRI) to the internal BGP group.
7. Configure RSVP or LDP.
8. Configure MPLS.
9. Create a label-switched path between the provider edge (PE) devices.



**NOTE:** We require that you configure the `no-gateway-community` option at the `[edit routing-instances EVPN-instance-name protocols evpn default-gateway]` hierarchy level in each EVPN routing instance in which you configure an IRB interface with a virtual gateway address. See *default-gateway* for details on using the `no-gateway-community` option. Note that these simple configuration steps don't include an IRB interface configuration with a virtual gateway address in the EVPN instance.

To configure the PE device:

1. Configure the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance instance-type evpn
```

2. Set the VLAN identifier for the bridging domain in the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vlan-id VLAN-ID
```

3. Configure the interface name for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance interface CE-facing-interface
```

4. Configure the IRB interface as the routing interface for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance routing-interface irb.0
```

5. Configure the route distinguisher for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance route-distinguisher route-distinguisher-value
```

6. Configure the VPN routing and forwarding (VRF) target community for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vrf-target vrf-target-value
```

7. Assign the interface name that connects the PE device site to the VPN.

```
[edit routing-instances]
user@PE1# set evpn-instance protocols evpn interface CE-facing-interface
```

8. Configure the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance instance-type vrf
```

9. Configure the IRB interface as the routing interface for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance interface irb.0
```

10. Configure the route distinguisher for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance route-distinguisher route-distinguisher-value
```

11. Configure the VRF label for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance vrf-table-label
```

12. Verify and commit the configuration.

For example:

```
[edit routing-instances]
user@PE1# set evpna instance-type evpn
user@PE1# set evpna vlan-id 10
user@PE1# set evpna interface ge-1/1/8.0
user@PE1# set evpna routing-interface irb.0
user@PE1# set evpna route-distinguisher 192.0.2.1:100
user@PE1# set evpna vrf-target target:100:100
user@PE1# set evpna protocols evpn interface ge-1/1/8.0
user@PE1# set vrf instance-type vrf
user@PE1# set vrf interface irb.0
user@PE1# set vrf route-distinguisher 192.0.2.1:300
user@PE1# set vrf vrf-target target:100:300
user@PE1# set vrf vrf-table-label
```

```
[edit]
user@PE1# commit
commit complete
```

To configure an **ACX** series router running Junos OS Evolved as the PE device in an EVPN-ELAN vlan-based routing-instance with IRB, verify and commit the following configuration:

```
[edit routing-instances]
user@PE1# set evpna instance-type mac-vrf
user@PE1# set evpna protocols evpn normalization
user@PE1# set evpna service-type vlan-based
user@PE1# set evpna route-distinguisher 192.0.2.1:100
user@PE1# set evpna vrf-target target:100:100
user@PE1# set evpna vlans v-500 vlan-id 10
```

```
user@PE1# set evpna vlans v-500 interface ae0.0
user@PE1# set evpna vlans v-500 l3-interface irb.0
```

```
[edit]
user@PE1# commit
commit complete
```

## RELATED DOCUMENTATION

[Example: Configuring EVPN with IRB Solution](#) | 1522

## Configuring an EVPN with IRB Solution on EX9200 Switches

You can configure an Ethernet VPN (EVPN) with IRB solution to enable Layer 2 switching and Layer 3 routing operations within a single node, thus avoiding extra hops for inter-subnet traffic. The EVPN IRB solution eliminates the default gateway problem using the gateway MAC and IP synchronization, and avoids the triangular routing problem with Layer 3 interworking by creating IP host routes for virtual machines (VMs) in the tenant virtual routing and forwarding (VRF) routing instances.

Before you begin:

1. Configure the switch interfaces.
2. Configure the router ID and autonomous system number for the device.
3. Enable the chained composite next hop for EVPN.
4. Configure OSPF or any other IGP protocol.
5. Configure a BGP internal group.
6. Include the EVPN signaling network layer reachability information (NLRI) to the internal BGP group.
7. Configure LDP.
8. Configure MPLS.

To configure the PE device:



1. Configure the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance instance-type evpn
```

2. Set the VLAN identifier for the bridging domain in the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vlan-id VLAN-ID
```

3. Configure the interface name for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance interface CE-facing-interface
```

4. Configure the IRB interface as the routing interface for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance l3-interface irb.0
```

5. Configure the route distinguisher for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance route-distinguisher route-distinguisher-value
```

6. Configure the VPN routing and forwarding (VRF) target community for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn-instance vrf-target vrf-target-value
```

7. Assign the interface name that connects the PE device site to the VPN.

```
[edit routing-instances]
user@PE1# set evpn-instance protocols evpn interface CE-facing-interface
```

8. Configure the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance instance-type vrf
```

9. Configure the IRB interface as the routing interface for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance interface irb.0
```

10. Configure the route distinguisher for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance route-distinguisher route-distinguisher-value
```

11. Configure the VRF label for the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf-instance vrf-table-label
```

12. Verify and commit the configuration.

For example:

```
[edit routing-instances]
user@PE1# set evpna instance-type evpn
user@PE1# set evpna vlan-id 10
user@PE1# set evpna interface ge-1/1/8.0
user@PE1# set evpna l3-interface irb.0
user@PE1# set evpna route-distinguisher 100.255.0.1:100
user@PE1# set evpna vrf-target target:100:100
user@PE1# set evpna protocols evpn interface ge-1/1/8.0
user@PE1# set vrf instance-type vrf
user@PE1# set vrf interface irb.0
user@PE1# set vrf route-distinguisher 100.255.0.1:300
```

```
user@PE1# set vrf vrf-target target:100:300
user@PE1# set vrf vrf-table-label
```

```
[edit]
user@PE1# commit
commit complete
```

## RELATED DOCUMENTATION

[Example: Configuring an EVPN with IRB Solution on EX9200 Switches | 1547](#)

## Example: Configuring EVPN with IRB Solution

### IN THIS SECTION

- [EVPN with IRB Solution Overview | 1522](#)
- [Example: Configuring EVPN-MPLS with IRB Solution | 1528](#)

## EVPN with IRB Solution Overview

### IN THIS SECTION

- [Need for an EVPN IRB Solution | 1523](#)
- [Implementing the EVPN IRB Solution | 1524](#)
- [Benefits of Implementing the EVPN IRB Solution | 1526](#)

A Data Center Service Provider (DCSP) hosts the data center for its multiple customers on a common physical network. To each customer (also called a tenant), the service looks like a full-fledged data center that can expand to 4094 VLANs and all private subnets. For disaster recovery, high availability, and

optimization of resource utilization, it is common for the DCSP to span across the data center to more than one site. To deploy the data center services, a DCSP faces the following main challenges:

- Extending Layer 2 domains across more than one data center site. This requires optimal intra-subnet traffic forwarding.
- Supporting optimal inter-subnet traffic forwarding and optimal routing in the event of virtual machine (VM).
- Supporting multiple tenants with independent VLAN and subnet space.

Ethernet VPN (EVPN) is targeted to handle all of the above mentioned challenges, wherein:

- The basic EVPN functionality enables optimal intra-subnet traffic forwarding
- Implementing the integrated routing and bridging (IRB) solution in an EVPN deployment enables optimal inter-subnet traffic forwarding
- Configuring EVPN with virtual switch support enables multiple tenants with independent VLAN and subnet space

The following sections describe the IRB solution for EVPNs:

### **Need for an EVPN IRB Solution**

EVPN is a technology used to provide Layer 2 extension and interconnection across an IP/MPLS core network to different physical sites belonging to a single Layer 2 domain. In a data center environment with EVPN, there is a need for both Layer 2 (intra-subnet traffic) and Layer 3 (inter-subnet traffic) forwarding and potentially interoperation with tenant Layer 3 VPNs.

With only a Layer 2 solution, there is no optimum forwarding of inter-subnet traffic, even when the traffic is local, for instance, when both the subnets are on the same server.

With only a Layer 3 solution, the following issues for intra-subnet traffic can arise:

- MAC address aliasing issue where duplicate MAC addresses are not detected.
- TTL issue for applications that use TTL 1 to confine traffic within a subnet.
- IPv6 link-local addressing and duplicate address detection that relies on Layer 2 connectivity.
- Layer 3 forwarding does not support the forwarding semantics of a subnet broadcast.
- Support of non-IP applications that require Layer 2 forwarding.

Because of the above mentioned shortcomings of a pure Layer 2 and Layer 3 solution, there is a need for a solution incorporating optimal forwarding of both Layer 2 and Layer 3 traffic in the data center

environment when faced with operational considerations such as Layer 3 VPN interoperability and virtual machine (VM) mobility.

An EVPN-based integrated routing and bridging (IRB) solution provides optimum unicast and multicast forwarding for both intra-subnets and inter-subnets within and across data centers.

The EVPN IRB feature is useful for service providers operating in an IP/MPLS network that provides both Layer 2 VPN or VPLS services and Layer 3 VPN services who want to extend their service to provide cloud computation and storage services to their existing customers.

### **Implementing the EVPN IRB Solution**

An EVPN IRB solution provides the following:

- Optimal forwarding for intra-subnet (Layer 2) traffic.
- Optimal forwarding for inter-subnet (Layer 3) traffic.
- Support for ingress replication for multicast traffic.
- Support for network-based as well as host-based overlay models.
- Support for consistent policy-based forwarding for both Layer 2 and Layer 3 traffic.
- Support for the following routing protocols on the IRB interface:
  - BFD
  - BGP
  - IS-IS
  - OSPF and OSPF version 3
- Support for single-active and all-active multihoming

Junos OS supports several models of EVPN configuration to satisfy the individual needs of EVPN and data center cloud services customers. To provide flexibility and scalability, multiple bridge domains can be defined within a particular EVPN instance. Likewise, one or more EVPN instances can be associated with a single Layer 3 VPN virtual routing and forwarding (VRF). In general, each data center tenant is assigned a unique Layer 3 VPN VRF, while a tenant could comprise one or more EVPN instances and one or more bridge domains per EVPN instance. To support this model, each configured bridge domain (including the default bridge domain for an EVPN instance) requires an IRB interface to perform the Layer 2 and Layer 3 functions. Each bridge domain or IRB interface maps to a unique IP subnet in the VRF.



**NOTE:** You can associate an IRB interface with the primary instance inet.0 table instead of a VRF in an EVPN IRB solution.

There are two major functions that are supported for IRB in EVPN.

- Host MAC-IP synchronization

This includes:

- Advertising the IP address along with the MAC advertisement route in EVPN. This is done by using the IP field in the EVPN MAC advertisement route.
- The receiving PE router installs MAC into the EVPN instance (EVI) table and installs IP into the associated VRF.

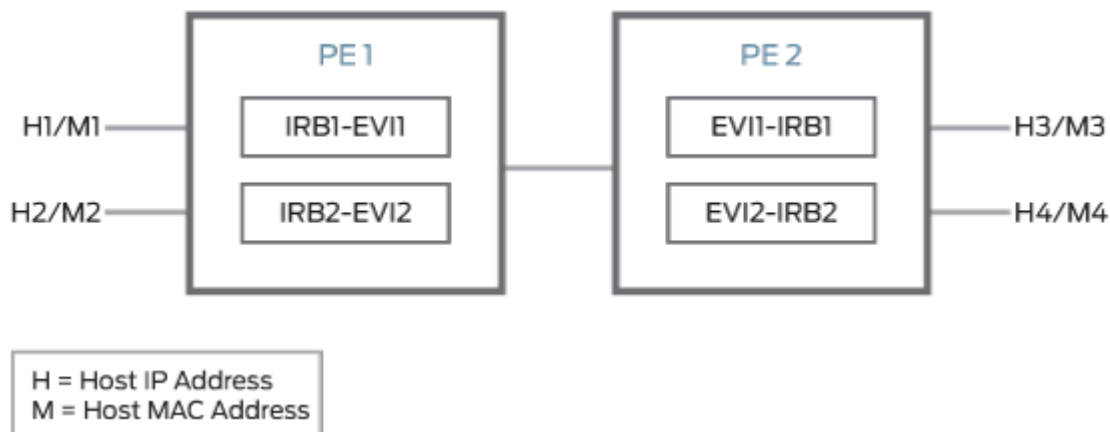
- Gateway MAC-IP synchronization

This includes:

- Advertising all local IRB MAC and IP addresses in an EVPN. This is achieved by including the default gateway extended community in the EVPN MAC advertisement route.
- The receiving PE creates a forwarding state to route packets destined for the gateway MAC, and a proxy ARP is done for the gateway IP with the MAC advertised in the route.

Figure 150 on page 1525 illustrates the inter-subnet traffic forwarding between two provider edge (PE) devices – PE1 and PE2. The IRB1 and IRB2 interfaces on each PE device belong to a different subnet, but they share a common VRF.

**Figure 150: Inter-Subnet Traffic Forwarding**



The inter-subnet traffic forwarding is performed as follows:

1. PE2 advertises H3-M3 and H4-M4 binding to PE1. Similarly PE1 advertises H1-M1 and H2-M2 binding to PE2.
2. PE1 and PE2 install the MAC address in the corresponding EVI MAC table, whereas the IP routes are installed in the shared VRF.
3. The advertising PE device is set as the next hop for the IP routes.
4. If H1 sends packets to H4, the packets are sent to IRB1 on PE1.
5. IP lookup for H4 happens in the shared VRF on PE1. Because the next hop for the H4 IP is PE2 (the advertising PE), an IP unicast packet is sent to PE2.
6. PE1 rewrites the MAC header based on the information in the VRF route, and PE2 performs a MAC lookup to forward the packet to H4.

### Benefits of Implementing the EVPN IRB Solution

The main goal of the EVPN IRB solution is to provide optimal Layer 2 and Layer 3 forwarding. The solution is required to efficiently handle inter-subnet forwarding as well as virtual machine (VM) mobility. VM mobility refers to the ability of a VM to migrate from one server to another within the same or a different data center while retaining its existing MAC and IP address. Providing optimal forwarding for inter-subnet traffic and effective VM mobility involves solving two problems – the default gateway problem and the triangular routing problem.

IPv6 addresses are supported on IRB interfaces with EVPN using the Neighbor Discovery Protocol (NDP). The following capabilities are introduced for IPv6 support with EVPN:

- IPv6 addresses on IRB interfaces in primary routing instances
- Learning IPv6 neighborhood from solicited NA message
- NS and NA packets on the IRB interfaces are disabled from network core
- Virtual gateway addresses are used as Layer 3 addresses
- Host MAC-IP synchronization for IPv6

You can configure the IPv6 addresses in the IRB interface at the `[edit interfaces irb]` hierarchy level.

### Gateway MAC and IP Synchronization

In an EVPN IRB deployment, the IP default gateway for a VM is the IP address configured on the IRB interface of the provider edge (PE) router corresponding to the bridge domain or VLAN of which the VM is a member. The default gateway problem arises because a VM does not flush its ARP table when relocating from one server to another and continues sending packets with the destination MAC address set to that of the original gateway. If the old and new servers are not part of the same Layer 2 domain

(the new Layer 2 domain could be within the current data center or a new data center), the gateway previously identified is no longer the optimal or local gateway. The new gateway needs to identify packets containing the MAC addresses of other gateways on remote PE routers and forward the traffic as if the packets were destined to the local gateway itself. At the minimum, this functionality requires each PE router to advertise its gateway or IRB MAC and IP addresses to all other PE routers in the network. The gateway address exchange can be accomplished using the standard MAC route advertisement message (including the IP address parameter) and tagging that route with the default gateway extended community so that the remote PE routers can distinguish the gateway MAC advertisement routes from normal MAC advertisement routes.



**NOTE:** We require that you configure the `no-gateway-community` option at the `[edit routing-instances EVPN-instance-name protocols evpn default-gateway]` hierarchy level in each EVPN routing instance in which you configure an IRB interface with a virtual gateway address. On platforms that support configuring the `no-gateway-community` option at the global level, you can alternatively configure `no-gateway-community` at the `[edit protocols evpn default-gateway]` hierarchy level.

See *default-gateway* for details on using the `no-gateway-community` option.

### Layer 3 VPN Interworking

The inter-data center aspect of the EVPN IRB solution involves routing between VMs that are present in different data centers or routing between a host site completely outside of the data center environment and a VM within a data center. This solution relies on the ability of EVPN MAC route advertisements to carry both MAC address and IP address information. The local MAC learning functionality of the PE router is extended to also capture IP address information associated with MAC addresses learned locally. That IP-MAC address mapping information is then distributed to each PE router through normal EVPN procedures. When a PE router receives such MAC and IP information, it installs the MAC route in the EVPN instance as well as a host route for the associated IP address in the Layer 3 VPN VRF corresponding to that EVPN instance. When a VM moves from one data center to another, normal EVPN procedures result in the MAC and IP address being advertised from the new PE router which the VM resides behind. The host route installed in the VRF associated with an EVPN solicits Layer 3 traffic destined to that VM to the new PE router and avoids triangular routing between the source, the former PE router the VM resided behind, and the new PE router.

BGP scalability is a potential concern with the inter-data center triangular routing avoidance solution because of the potential for injection of many host routes into Layer 3 VPN. With the method previously described, in the worst case there is an IP host route for each MAC address learned through the local EVPN MAC learning procedures or through a MAC advertisement message received from a remote PE router. BGP route target filtering can be used to limit distribution of such routes.



The following functional elements are required to implement the inter-data center triangular routing avoidance using Layer 3 inter-subnet forwarding procedures:

1. The source host sends an IP packet using its own source MAC and IP address with the destination MAC of the IRB interface of the local PE router and the IP address of the destination host.
2. When the IRB interface receives the frame with its MAC as the destination, it performs a Layer 3 lookup in the VRF associated with the EVPN instance to determine where to route the packet.
3. In the VRF, the PE router finds the Layer 3 route derived from a MAC plus an IP EVPN route received from the remote PE router earlier. The destination MAC address is then changed to the destination MAC address corresponding to the destination IP.
4. The packet is then forwarded to the remote PE router serving the destination host using MPLS, using the label corresponding to the EVPN instance of which the destination host is a member.
5. The egress PE router receiving the packet performs a Layer 2 lookup for the destination host's MAC and sends the packet to the destination host on the attached subnet via the egress PE router's IRB interface.
6. Because the ingress PE router is performing Layer 3 routing, the IP TTL is decremented.

## SEE ALSO

[EVPN Multihoming Overview | 162](#)

[Understanding VXLANs | 606](#)

[Example: Configuring EVPN-MPLS with IRB Solution | 1528](#)

## Example: Configuring EVPN-MPLS with IRB Solution

### IN THIS SECTION

- [Requirements | 1529](#)
- [Overview | 1529](#)
- [Configuration | 1530](#)
- [Verification | 1538](#)

This example shows how to configure an integrated routing and bridging (IRB) solution in an Ethernet VPN (EVPN) deployment.

## Requirements

This example uses the following hardware and software components:

- Two MX Series Routing Platforms as PE routers.
- Two customer edge (CE) routers, each connected to the PE routers.
- Junos OS Release 14.1 or later running on all the PE routers.
  - Updated and re-validated using Junos OS Release 22.1R1.

Before you begin:

1. Configure the router interfaces.
2. Configure OSPF or any other IGP protocol.
3. Configure BGP.
4. Configure RSVP or LDP.
5. Configure MPLS.

## Overview

### IN THIS SECTION

- [Topology | 1530](#)

In an EVPN solution, multiple bridge domains can be defined within a particular EVPN instance, and one or more EVPN instances can be associated with a single Layer 3 VPN VRF. In general, each data center tenant is assigned a unique Layer 3 VPN virtual route forwarding (VRF), although the tenant can be comprised of one or more EVPN instances or bridge domains per EVPN instance.

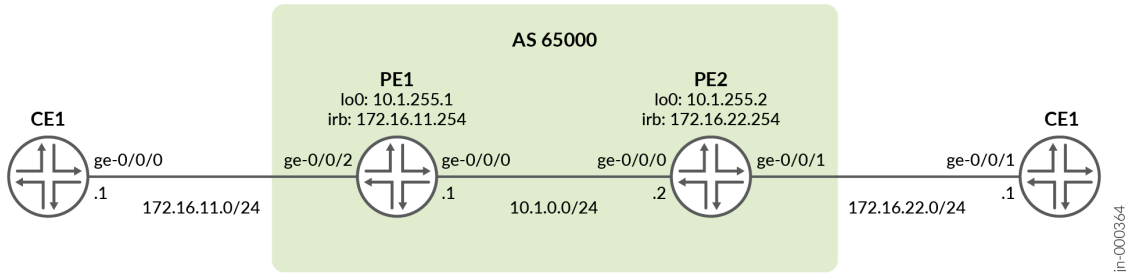
To support this flexibility and scalability factor, the EVPN solution provides support for the IRB interfaces on MX Series routers containing MPC FPCs to facilitate optimal Layer 2 and Layer 3 forwarding along with virtual machine mobility. The IRB interfaces are configured on each configured bridge domain including the default bridge domain for an EVPN instance.

IRB is the ability to do Layer 2 switching and Layer 3 routing within a single node, thus avoiding extra hops for inter-subnet traffic. The EVPN IRB solution eliminates the default gateway problem using the gateway MAC and IP synchronization, and avoids the triangular routing problem with Layer 3 interworking by creating IP host routes for virtual machines (VMs) in the tenant VRFs.

Topology

Figure 151 on page 1530 illustrates a simple EVPN topology with IRB solution. Routers PE1 and PE2 are the provider edge routers that connect to two customer edge (CE) routers each – CE1 and CE2.

Figure 151: EVPN with IRB Solution



Configuration

IN THIS SECTION

- Procedure | 1530
- Results | 1535

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.

CE1

```
set interfaces ge-0/0/0 vlan-tagging
set interfaces ge-0/0/0 unit 0 vlan-id 10
```

```

set interfaces ge-0/0/0 unit 0 family inet address 172.16.11.1/24
set routing-options static route 172.16.22.0/24 next-hop 172.16.11.254

```

## PE1

```

set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.1/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/2 flexible-vlan-tagging
set interfaces ge-0/0/2 encapsulation flexible-ethernet-services
set interfaces ge-0/0/2 unit 0 encapsulation vlan-bridge
set interfaces ge-0/0/2 unit 0 vlan-id 10
set interfaces irb unit 0 family inet address 172.16.11.254/24
set interfaces lo0 unit 0 family inet address 10.1.255.1/32
set routing-instances evpna instance-type evpn
set routing-instances evpna protocols evpn interface ge-0/0/2.0
set routing-instances evpna vlan-id 10
set routing-instances evpna routing-interface irb.0
set routing-instances evpna interface ge-0/0/2.0
set routing-instances evpna route-distinguisher 10.1.255.1:1
set routing-instances evpna vrf-target target:65000:1
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf route-distinguisher 10.1.255.1:10
set routing-instances vrf vrf-target target:65000:10
set routing-instances vrf vrf-table-label
set routing-options router-id 10.1.255.1
set routing-options autonomous-system 65000
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.1.255.1
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 10.1.255.2
set protocols mpls label-switched-path PE1-to-PE2 from 10.1.255.1
set protocols mpls label-switched-path PE1-to-PE2 to 10.1.255.2
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable

```

```
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
```

## PE2

```
set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.2/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 flexible-vlan-tagging
set interfaces ge-0/0/1 encapsulation flexible-ethernet-services
set interfaces ge-0/0/1 unit 0 encapsulation vlan-bridge
set interfaces ge-0/0/1 unit 0 vlan-id 20
set interfaces irb unit 0 family inet address 172.16.22.254/24
set interfaces lo0 unit 0 family inet address 10.1.255.2/32
set routing-instances evpna instance-type evpn
set routing-instances evpna protocols evpn interface ge-0/0/1.0
set routing-instances evpna vlan-id 20
set routing-instances evpna routing-interface irb.0
set routing-instances evpna interface ge-0/0/1.0
set routing-instances evpna route-distinguisher 10.1.255.2:1
set routing-instances evpna vrf-target target:65000:1
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf route-distinguisher 10.1.255.2:10
set routing-instances vrf vrf-target target:65000:10
set routing-instances vrf vrf-table-label
set routing-options router-id 10.1.255.2
set routing-options autonomous-system 65000
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.1.255.2
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 10.1.255.1
set protocols mpls label-switched-path PE2-to-PE1 from 10.1.255.2
set protocols mpls label-switched-path PE2-to-PE1 to 10.1.255.1
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

```
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
```

## CE2

```
set interfaces ge-0/0/1 vlan-tagging
set interfaces ge-0/0/1 unit 0 vlan-id 20
set interfaces ge-0/0/1 unit 0 family inet address 172.16.22.1/24
set routing-options static route 172.16.11.0/24 next-hop 172.16.22.254
```

## Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see [Use the CLI Editor in Configuration Mode](#).

To configure PE1:



**NOTE:** Repeat this procedure for PE2, after modifying the appropriate interface names, addresses, and other parameters.

### 1. Configure the interfaces on PE1.

```
user@PE1# set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.1/24
user@PE1# set interfaces ge-0/0/0 unit 0 family mpls
user@PE1# set interfaces ge-0/0/2 flexible-vlan-tagging
user@PE1# set interfaces ge-0/0/2 encapsulation flexible-ethernet-services
user@PE1# set interfaces ge-0/0/2 unit 0 encapsulation vlan-bridge
user@PE1# set interfaces ge-0/0/2 unit 0 vlan-id 10
user@PE1# set interfaces irb unit 0 family inet address 172.16.11.254/24
user@PE1# set interfaces lo0 unit 0 family inet address 10.1.255.1/32
```

### 2. Set the router ID and autonomous system number for PE1.

```
user@PE1# set routing-options router-id 10.1.255.1
user@PE1# set routing-options autonomous-system 65000
```

3. Configure the chained composite next hop for EVPN.

```
user@PE1# set routing-options forwarding-table chained-composite-next-hop ingress evpn
```

4. Enable RSVP on all the interfaces of PE1, excluding the management interface.

```
user@PE1# set protocols rsvp interface all
user@PE1# set protocols rsvp interface fxp0.0 disable
```

5. Enable MPLS on all the interfaces of PE1, excluding the management interface. Create a label-switched path from PE1 to PE2.

```
user@PE1# set protocols mpls label-switched-path PE1-to-PE2 from 10.1.255.1
user@PE1# set protocols mpls label-switched-path PE1-to-PE2 to 10.1.255.2
user@PE1# set protocols mpls interface all
user@PE1# set protocols mpls interface fxp0.0 disable
```

6. Configure the BGP group for IBGP on PE1. Assign local and neighbor addresses for PE1 to peer with PE2 using the loopback address. Include the family inet-vpn unicast and evpn signaling for Network Layer Reachability Information (NLRI).

```
user@PE1# set protocols bgp group ibgp type internal
user@PE1# set protocols bgp group ibgp local-address 10.1.255.1
user@PE1# set protocols bgp group ibgp family inet-vpn unicast
user@PE1# set protocols bgp group ibgp family evpn signaling
user@PE1# set protocols bgp group ibgp neighbor 10.1.255.2
```

7. Configure OSPF on all the interfaces of PE1, excluding the management interface. Enable traffic-engineering for OSPF. For RSVP-signaled LSPs with OSPF as the IGP, traffic-engineering must be enabled for the LSPs to come up.

```
user@PE1# set protocols ospf traffic-engineering
user@PE1# set protocols ospf area 0.0.0.0 interface all
user@PE1# set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

8. Configure the EVPN routing instance. Configure the VLAN identifier, the interface connected to CE1, the IRB interface as a routing interface, the route distinguisher, and the VRF target for the evpna routing instance.

```

user@PE1#set routing-instances evpna instance-type evpn
user@PE1#set routing-instances evpna protocols evpn interface ge-0/0/2.0
user@PE1#set routing-instances evpna vlan-id 10
user@PE1#set routing-instances evpna routing-interface irb.0
user@PE1#set routing-instances evpna interface ge-0/0/2.0
user@PE1#set routing-instances evpna route-distinguisher 10.1.255.1:1
user@PE1#set routing-instances evpna vrf-target target:65000:1

```

9. Configure the VRF routing instance. Configure the IRB interface, the route distinguisher, the VRF target, and the VRF table label for the vrf routing instance.

```

user@PE1# set routing-instances vrf instance-type vrf
user@PE1# set routing-instances vrf interface irb.0
user@PE1# set routing-instances vrf route-distinguisher 10.1.255.1:10
user@PE1# set routing-instances vrf vrf-target target:65000:10
user@PE1# set routing-instances vrf vrf-table-label

```

## Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show routing-options`, `show protocols`, 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-0/0/0 {
 unit 0 {
 family inet {
 address 10.1.0.1/24;
 }
 family mpls;
 }
}
ge-0/0/2 {
 flexible-vlan-tagging;
 encapsulation flexible-ethernet-services;
 unit 0 {

```



```

 encapsulation vlan-bridge;
 vlan-id 10;
 }
}
irb {
 unit 0 {
 family inet {
 address 172.16.11.254/24;
 }
 }
}
lo0 {
 unit 0 {
 family inet {
 address 10.1.255.1/32;
 }
 }
}
}

```

```

user@PE1# show routing-options
router-id 10.1.255.1;
autonomous-system 65000;
forwarding-table {
 chained-composite-next-hop {
 ingress {
 evpn;
 }
 }
}
}

```

```

user@PE1# show protocols
bgp {
 group ibgp {
 type internal;
 local-address 10.1.255.1;
 family inet-vpn {
 unicast;
 }
 family evpn {
 signaling;
 }
 }
}

```

```

 }
 neighbor 10.1.255.2;
 }
}
mpls {
 label-switched-path PE1-to-PE2 {
 from 10.1.255.1;
 to 10.1.255.2;
 }
 interface all;
 interface fxp0.0 {
 disable;
 }
}
ospf {
 traffic-engineering;
 area 0.0.0.0 {
 interface all;
 interface fxp0.0 {
 disable;
 }
 }
}
}
rsvp {
 interface all;
 interface fxp0.0 {
 disable;
 }
}
}

```

```

user@PE1# show routing-instances
evpna {
 instance-type evpn;
 protocols {
 evpn {
 interface ge-0/0/2.0;
 }
 }
 vlan-id 10;
 routing-interface irb.0;
 interface ge-0/0/2.0;
}

```

```
route-distinguisher 10.1.255.1:1;
vrf-target target:65000:1;
}
vrf {
 instance-type vrf;
 interface irb.0;
 route-distinguisher 10.1.255.1:10;
 vrf-target target:65000:10;
 vrf-table-label;
}
```

## Verification

### IN THIS SECTION

- [Verifying Local IRB MACs | 1538](#)
- [Verifying Remote IRB MACs | 1539](#)
- [Verifying Local IRB IPs | 1541](#)
- [Verifying Remote IRB IPs | 1542](#)
- [Verifying CE-CE Reachability | 1543](#)
- [Verifying CE-PE Reachability | 1544](#)
- [Verifying PE-PE Reachability | 1545](#)

Confirm that the configuration is working properly.

### *Verifying Local IRB MACs*

#### Purpose

Verify that the local IRB MACs are learned from L2ALD.

#### Action

On PE1, determine the MAC address of the local IRB interface.

From operational mode, run the `show interfaces irb extensive | match "Current address"` command.

```
user@PE1> show interfaces irb extensive | match "Current address"
Current address: 2c:6b:f5:1b:46:f0, Hardware address: 2c:6b:f5:1b:46:f0
```

From operational mode, run the `show route table evpn.evpn.0 extensive | find 2c:6b:f5:1b:46:f0` command.

```
user@PE1> show route table evpn.evpn.0 extensive | find 2c:6b:f5:1b:46:f0
2:10.1.255.1:1::10::2c:6b:f5:1b:46:f0/304 MAC/IP (1 entry, 1 announced)
 *EVPN Preference: 170
 Next hop type: Indirect, Next hop index: 0
 Address: 0x7a18b78
 Next-hop reference count: 12, key opaque handle: 0x0
 Protocol next hop: 10.1.255.1
 Indirect next hop: 0x0 - INH Session ID: 0
 State: <Active Int Ext>
 Age: 1:34:57
 Validation State: unverified
 Task: evpn-evpn
 Announcement bits (1): 2-rt-export
 AS path: I
 Communities: evpn-default-gateway
 Route Label: 299936
 ESI: 00:00:00:00:00:00:00:00:00:00
 Thread: junos-main
```

## Meaning

The route for the local IRB interface appears in the EVPN instance route table on PE1 and is learned from EVPN and tagged with the default gateway extended community.

### *Verifying Remote IRB MACs*

## Purpose

Verify that the remote IRB MACs are learned from BGP.

## Action

On PE2, verify that the remote IRB MAC from PE1 is learned.

From operational mode, run the same show route table evpna.evpn.0 extensive | find 2c:6b:f5:1b:46:f0 command that was run on PE1.

```

user@PE2> show route table evpna.evpn.0 extensive | find 2c:6b:f5:1b:46:f0
2:10.1.255.1:1::10::2c:6b:f5:1b:46:f0/304 MAC/IP (1 entry, 1 announced)
 *BGP Preference: 170/-101
 Route Distinguisher: 10.1.255.1:1
 Next hop type: Indirect, Next hop index: 0
 Address: 0x7a19160
 Next-hop reference count: 10, key opaque handle: 0x0
 Source: 10.1.255.1
 Protocol next hop: 10.1.255.1
 Indirect next hop: 0x2 no-forward INH Session ID: 0
 State: <Secondary Active Int Ext>
 Local AS: 65000 Peer AS: 65000
 Age: 1:41:11 Metric2: 1
 Validation State: unverified
 Task: BGP_65000.10.1.255.1
 Announcement bits (1): 0-evpna-evpn
 AS path: I
 Communities: target:65000:1 evpn-default-gateway
 Import Accepted
 Route Label: 299936
 ESI: 00:00:00:00:00:00:00:00:00:00
 Localpref: 100
 Router ID: 10.1.255.1
 Primary Routing Table: bgp.evpn.0
 Thread: junos-main
 Indirect next hops: 1
 Protocol next hop: 10.1.255.1 Metric: 1
 Indirect next hop: 0x2 no-forward INH Session ID: 0
 Indirect path forwarding next hops: 1
 Next hop type: Router
 Next hop: 10.1.0.1 via ge-0/0/0.0
 Session Id: 0
 10.1.255.1/32 Originating RIB: inet.3
 Metric: 1 Node path count: 1
 Forwarding nexthops: 1
 Next hop type: Router
 Next hop: 10.1.0.1 via ge-0/0/0.0
 Session Id: 0

```

## Meaning

The route for the remote IRB interface appears in the EVPN instance route table on PE2. The route is learned from BGP and tagged with the default gateway extended community.

### *Verifying Local IRB IPs*

## Purpose

Verify that the local IRB IPs are learned locally by RPD.

## Action

On PE1, determine the MAC and IP addresses of the local IRB interface.

From operational mode, run the `show interfaces irb extensive | match "Current address"` command.

```
user@PE1> show interfaces irb extensive | match "Current address"
Current address: 2c:6b:f5:1b:46:f0, Hardware address: 2c:6b:f5:1b:46:f0
```

From operational mode, run the `show interfaces irb.0 terse | match inet` command.

```
user@PE1> show interfaces irb.0 terse | match inet
irb.0 up up inet 172.16.11.254/24
```

From operational mode, run the `show route table evpna.evpn.0 extensive | find "a8:d0:e5:54:0d:10::10.0.0.251"` command.

```
user@PE1> show route table evpna.evpn.0 extensive | find 2c:6b:f5:1b:46:f0::172.16.11.254
2:10.1.255.1:1::10::2c:6b:f5:1b:46:f0::172.16.11.254/304 MAC/IP (1 entry, 1 announced)
 *EVPN Preference: 170
 Next hop type: Indirect, Next hop index: 0
 Address: 0x7a18b78
 Next-hop reference count: 12, key opaque handle: 0x0
 Protocol next hop: 10.1.255.1
 Indirect next hop: 0x0 - INH Session ID: 0
 State: <Active Int Ext>
 Age: 2:12:19
```

```

Validation State: unverified
Task: evpna-evpn
Announcement bits (1): 2-rt-export
AS path: I
Communities: evpn-default-gateway
Route Label: 299936
ESI: 00:00:00:00:00:00:00:00:00
Thread: junos-main

```

## Meaning

The MAC plus IP route for the local IRB interface appears in the EVPN instance route table on PE1 and is learned from EVPN and tagged with the default gateway extended community.

### *Verifying Remote IRB IPs*

## Purpose

Verify that the remote IRB IP is learned from BGP.

## Action

On Router PE2, verify that the remote IRB MAC from PE1 is learned.

From operational mode, run the same `show route table evpna.evpn.0 extensive | find 2c:6b:f5:1b:46:f0::172.16.11.254` command that was run on PE1.

```

user@PE2> show route table evpna.evpn.0 extensive | find 2c:6b:f5:1b:46:f0::172.16.11.254
2:10.1.255.1:1::10::2c:6b:f5:1b:46:f0::172.16.11.254/304 MAC/IP (1 entry, 1 announced)
 *BGP Preference: 170/-101
 Route Distinguisher: 10.1.255.1:1
 Next hop type: Indirect, Next hop index: 0
 Address: 0x7a19160
 Next-hop reference count: 10, key opaque handle: 0x0
 Source: 10.1.255.1
 Protocol next hop: 10.1.255.1
 Indirect next hop: 0x2 no-forward INH Session ID: 0
 State: <Secondary Active Int Ext>
 Local AS: 65000 Peer AS: 65000
 Age: 2:13:11 Metric2: 1
 Validation State: unverified

```

```

Task: BGP_65000.10.1.255.1
Announcement bits (1): 0-evpna-evpn
AS path: I
Communities: target:65000:1 evpn-default-gateway
Import Accepted
Route Label: 299936
ESI: 00:00:00:00:00:00:00:00:00
Localpref: 100
Router ID: 10.1.255.1
Primary Routing Table: bgp.evpn.0
Thread: junos-main
Indirect next hops: 1
 Protocol next hop: 10.1.255.1 Metric: 1
 Indirect next hop: 0x2 no-forward INH Session ID: 0
 Indirect path forwarding next hops: 1
 Next hop type: Router
 Next hop: 10.1.0.1 via ge-0/0/0.0
 Session Id: 0
 10.1.255.1/32 Originating RIB: inet.3
 Metric: 1 Node path count: 1
 Forwarding nexthops: 1
 Next hop type: Router
 Next hop: 10.1.0.1 via ge-0/0/0.0
 Session Id: 0

```

## Meaning

The MAC plus IP route for the remote IRB interface appears in the EVPN instance route table on PE2 and is tagged with the default gateway extended community.

## *Verifying CE-CE Reachability*

## Purpose

Verify CE1 can ping CE2.



## Action

From operational mode, run the `show route 172.16.22.1` command on CE1 to ping CE2.

```
user@CE1> show route 172.16.22.1

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

172.16.22.0/24 *[Static/5] 02:28:23
 > to 172.16.11.254 via ge-0/0/0.0
```

From operational mode, run the `ping` command on CE1 to ping CE2.

```
user@CE1> ping 172.16.22.1 count 2
PING 172.16.22.1 (172.16.22.1): 56 data bytes
64 bytes from 172.16.22.1: icmp_seq=0 ttl=62 time=4.890 ms
64 bytes from 172.16.22.1: icmp_seq=1 ttl=62 time=4.658 ms

--- 172.16.22.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 4.658/4.774/4.890/0.116 ms
```

## Meaning

Ping from CE1 to CE2 is successful.

### *Verifying CE-PE Reachability*

## Purpose

Verify CE1 can ping PE2.

## Action

From operational mode, run the `show route table vrf.inet.0` command on PE2.

```
user@PE2> show route table vrf.inet.0
```

```

vrf.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.11.0/24 *[BGP/170] 02:27:44, localpref 100, from 10.1.255.1
 AS path: I, validation-state: unverified
 > to 10.1.0.1 via ge-0/0/0.0, label-switched-path PE2-to-PE1
172.16.11.1/32 *[BGP/170] 02:27:44, localpref 100, from 10.1.255.1
 AS path: I, validation-state: unverified
 > to 10.1.0.1 via ge-0/0/0.0, label-switched-path PE2-to-PE1
172.16.22.0/24 *[Direct/0] 02:28:14
 > via irb.0
172.16.22.1/32 *[EVPN/7] 02:24:44
 > via irb.0
172.16.22.254/32 *[Local/0] 02:28:14
 Local via irb.0

```

From operational mode, run the ping command on CE1 to ping the IRB interface on PE2.

```

user@CE1> ping 172.16.22.254 count 2
PING 172.16.22.254 (172.16.22.254): 56 data bytes
64 bytes from 172.16.22.254: icmp_seq=0 ttl=63 time=3.662 ms
64 bytes from 172.16.22.254: icmp_seq=1 ttl=63 time=2.766 ms

--- 172.16.22.254 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.766/3.214/3.662/0.448 ms

```

## Meaning

Ping from CE1 to PE2 is successful.

## Verifying PE-PE Reachability

## Purpose

Verify PE1 can ping PE2.

## Action

From operational mode, run the `show route table vrf.inet.0` command on PE1.

```
user@PE1> show route table vrf.inet.0

vrf.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.11.0/24 *[Direct/0] 02:40:42
 > via irb.0
172.16.11.1/32 *[EVPN/7] 02:37:42
 > via irb.0
172.16.11.254/32 *[Local/0] 02:40:42
 Local via irb.0
172.16.22.0/24 *[BGP/170] 02:35:48, localpref 100, from 10.1.255.2
 AS path: I, validation-state: unverified
 > to 10.1.0.2 via ge-0/0/0.0, label-switched-path PE1-to-PE2
172.16.22.1/32 *[BGP/170] 02:32:46, localpref 100, from 10.1.255.2
 AS path: I, validation-state: unverified
 > to 10.1.0.2 via ge-0/0/0.0, label-switched-path PE1-to-PE2
```

From operational mode, run the `ping` command from PE1 to ping the IRB interface on PE2.

```
user@PE1> ping 172.16.22.254 routing-instance vrf count 2
PING 172.16.22.254 (172.16.22.254): 56 data bytes
64 bytes from 172.16.22.254: icmp_seq=0 ttl=64 time=1.946 ms
64 bytes from 172.16.22.254: icmp_seq=1 ttl=64 time=2.151 ms

--- 172.16.22.254 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.946/2.048/2.151/0.102 ms
```

## Meaning

Ping from PE1 to PE2 is successful.

## SEE ALSO

[EVPN with IRB Solution Overview | 1499](#)

## Example: Configuring an EVPN with IRB Solution on EX9200 Switches

### IN THIS SECTION

- [Requirements | 1547](#)
- [Overview | 1547](#)
- [Configuration | 1548](#)
- [Verification | 1557](#)

This example shows how to configure an integrated routing and bridging (IRB) solution in an Ethernet VPN (EVPN) deployment.

### Requirements

This example uses the following hardware and software components:

- Two EX9200 switches configured as PE routers
- Junos OS Release 14.2 or later running on all the PE routers

Before you begin:

1. Configure the switch interfaces.
2. Configure OSPF or any other IGP protocol.
3. Configure BGP.
4. Configure LDP.
5. Configure MPLS.

### Overview

In an EVPN solution, multiple VLANs can be defined within a particular EVPN instance, and one or more EVPN instances can be associated with a single Layer 3 VPN VRF. In general, each data center tenant is

assigned a unique Layer 3 VPN virtual route forwarding (VRF), although the tenant can comprise one or more EVPN instances or VLANs per EVPN instance.

To support this flexibility and scalability factor, the EVPN solution provides support for the IRB interfaces on EX9200 switches to facilitate optimal Layer 2 and Layer 3 forwarding along with virtual machine mobility. The IRB interfaces are configured on each configured VLAN including the default VLAN for an EVPN instance.

IRB is the ability to do Layer 2 switching and Layer 3 routing within a single node, thus avoiding extra hops for inter-subnet traffic. The EVPN IRB solution eliminates the default gateway problem using the gateway MAC and IP synchronization, and avoids the triangular routing problem with Layer 3 interworking by creating IP host routes for virtual machines (VMs) in the tenant VRFs.

## Configuration

### IN THIS SECTION

- Procedure | [1548](#)
- Results | [1554](#)

## 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.

#### CE1

```
set interfaces ge-1/1/7 vlan-tagging
set interfaces ge-1/1/7 unit 0 vlan-id 10
set interfaces ge-1/1/7 unit 0 family inet address 10.0.0.1/24
set routing-options static route 198.51.100.0/24 next-hop 10.0.0.251
```

#### PE1

```
set interfaces ge-1/0/8 unit 0 family inet address 192.0.2.1/24
set interfaces ge-1/0/8 unit 0 family mpls
```

```

set interfaces ge-1/1/8 flexible-vlan-tagging
set interfaces ge-1/1/8 encapsulation flexible-ethernet-services
set interfaces ge-1/1/8 unit 0 encapsulation vlan-bridge
set interfaces ge-1/1/8 unit 0 vlan-id 10
set interfaces irb unit 0 family inet address 10.0.0.251/24
set interfaces lo0 unit 0 family inet address 203.0.113.1/32
set routing-options router-id 203.0.113.1
set routing-options autonomous-system 100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 203.0.113.1
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 203.0.113.2
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances evpna instance-type evpn
set routing-instances evpna vlan-id 10
set routing-instances evpna interface ge-1/1/8.0
set routing-instances evpna l3-interface irb.0
set routing-instances evpna route-distinguisher 203.0.113.1:100
set routing-instances evpna vrf-target target:100:100
set routing-instances evpna protocols evpn interface ge-1/1/8.0
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf route-distinguisher 203.0.113.1:300
set routing-instances vrf vrf-target target:100:300
set routing-instances vrf vrf-table-label

```

## PE2

```

set interfaces ge-2/0/8 unit 0 family inet address 192.0.2.2/24
set interfaces ge-2/0/8 unit 0 family mpls
set interfaces ge-2/1/8 flexible-vlan-tagging
set interfaces ge-2/1/8 encapsulation flexible-ethernet-services
set interfaces ge-2/1/8 unit 0 encapsulation vlan-bridge
set interfaces ge-2/1/8 unit 0 vlan-id 20
set interfaces irb unit 0 family inet address 198.51.100.251/24
set interfaces lo0 unit 0 family inet address 203.0.113.2/32

```

```

set routing-options router-id 203.0.113.2
set routing-options autonomous-system 100
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 203.0.113.2
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 203.0.113.1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances evpna instance-type evpn
set routing-instances evpna vlan-id 20
set routing-instances evpna interface ge-2/1/8.0
set routing-instances evpna l3-interface irb.0
set routing-instances evpna route-distinguisher 203.0.113.2:100
set routing-instances evpna vrf-target target:200:100
set routing-instances evpna protocols evpn interface ge-2/1/8.0
set routing-instances vrf instance-type vrf
set routing-instances vrf interface irb.0
set routing-instances vrf route-distinguisher 203.0.113.2:300
set routing-instances vrf vrf-target target:200:300
set routing-instances vrf vrf-table-label

```

## CE2

```

set interfaces ge-2/1/7 unit 0 vlan-id 20
set interfaces ge-2/1/7 unit 0 family inet address 198.51.100.2/24
set routing-options static route 10.0.0.0/24 next-hop 198.51.100.251

```

## 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 Router PE1:



**NOTE:** Repeat this procedure for Router PE2, after modifying the appropriate interface names, addresses, and other parameters.

1. Configure Router PE1 interfaces.

```
[edit interfaces]
user@PE1# set ge-1/0/8 unit 0 family inet address 192.0.2.1/24
user@PE1# set ge-1/0/8 unit 0 family mpls
user@PE1# set ge-1/1/8 flexible-vlan-tagging
user@PE1# set ge-1/1/8 encapsulation flexible-ethernet-services
user@PE1# set ge-1/1/8 unit 0 encapsulation vlan-bridge
user@PE1# set ge-1/1/8 unit 0 vlan-id 10
user@PE1# set irb unit 0 family inet address 10.0.0.251/24
user@PE1# set lo0 unit 0 family inet address 203.0.113.1/32
```

2. Set the router ID and autonomous system number for Router PE1.

```
[edit routing-options]
user@PE1# set router-id 203.0.113.1
user@PE1# set autonomous-system 100
```

3. Configure the chained composite next hop for EVPN.

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

4. Enable LDP on all interfaces of Router PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set ldp interface all
user@PE1# set ldp interface fxp0.0 disable
```



5. Enable MPLS on all the interfaces of Router PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set mpls interface all
user@PE1# set mpls mpls interface fxp0.0 disable
```

6. Configure the BGP group for Router PE1.

```
[edit protocols]
user@PE1# set bgp group ibgp type internal
```

7. Assign local and neighbor addresses to the ibgp BGP group for Router PE1 to peer with Router PE2.

```
[edit protocols]
user@PE1# set bgp group ibgp local-address 203.0.113.1
user@PE1# set bgp group ibgp neighbor 203.0.113.2
```

8. Include the EVPN signaling Network Layer Reachability Information (NLRI) to the ibgp BGP group.

```
[edit protocols]
user@PE1# set bgp group ibgp family evpn signaling
```

9. Configure OSPF on all the interfaces of Router PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set ospf area 0.0.0.0 interface all
user@PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
```

10. Configure the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn instance-type evpn
```

11. Set the VLAN identifier for the bridging domain in the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna vlan-id 10
```

12. Configure the interface name for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna interface ge-1/1/8.0
```

13. Configure the IRB interface as the routing interface for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna l3-interface irb.0
```

14. Configure the route distinguisher for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna route-distinguisher 203.0.113.1:100
```

15. Configure the VPN routing and forwarding (VRF) target community for the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna vrf-target target:100:100
```

16. Assign the interface name that connects the PE1 site to the VPN.

```
[edit routing-instances]
user@PE1# set evpna protocols evpn interface ge-1/1/8.0
```

17. Configure the VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf instance-type vrf
```

18. Configure the IRB interface as the routing interface for the vrf routing instance.

```
[edit routing-instances]
user@PE1# set vrf interface irb.0
```

19. Configure the route distinguisher for the vrf routing instance.

```
[edit routing-instances]
user@PE1# set vrf route-distinguisher 203.0.113.1:300
```

20. Configure the VRF label for the vrf routing instance.

```
[edit routing-instances]
user@PE1# set vrf vrf-table-label
```

## Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show routing-options`, `show protocols`, 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-1/0/8 {
 unit 0 {
 family inet {
 address 192.0.2.1/24;
 }
 family mpls;
 }
}
ge-1/1/8 {
 flexible-vlan-tagging;
 encapsulation flexible-ethernet-services;
 unit 0 {
 encapsulation vlan-bridge;
 vlan-id 10;
 }
}
irb {
```

```

 unit 0 {
 family inet {
 address 10.0.0.251/24;
 }
 }
}
lo0 {
 unit 0 {
 family inet {
 address 203.0.113.1/32 {
 }
 }
 }
}
}

```

```

user@PE1# show routing-options
router-id 203.0.113.1;
autonomous-system 100;
forwarding-table {
 chained-composite-next-hop {
 ingress {
 evpn;
 }
 }
}

```

```

user@PE1# show protocols
ldp {
 interface all;
 interface fxp0.0 {
 disable;
 }
}
mpls {
 interface all;
 interface fxp0.0 {
 disable;
 }
}
bgp {

```

```

group ibgp {
 type internal;
 local-address 203.0.113.1;
 family evpn {
 signaling;
 }
 neighbor 203.0.113.2;
}
}
ospf {
 area 0.0.0.0 {
 interface all;
 interface fxp0.0 {
 disable;
 }
 }
}
}

```

```

user@PE1# show routing-instances
evpna {
 instance-type evpn;
 vlan-id 10;
 interface ge-1/1/8.0;
 l3-interface irb.0;
 route-distinguisher 203.0.113.1:100;
 vrf-target target:100:100;
 protocols {
 evpn {
 interface ge-1/1/8.0;
 }
 }
}
vrf {
 instance-type vrf;
 interface irb.0;
 route-distinguisher 203.0.113.1:300;
 vrf-target target:100:300;
 vrf-table-label;
}

```

## Verification

### IN THIS SECTION

- [Verifying Local IRB MACs | 1557](#)
- [Verifying Remote IRB MACs | 1558](#)
- [Verifying Local IRB IPs | 1560](#)
- [Verifying Remote IRB IPs | 1561](#)
- [Verifying CE-CE Inter-Subnet Forwarding | 1563](#)

Confirm that the configuration is working properly.

### Verifying Local IRB MACs

#### Purpose

Verify that the local IRB MACs are learned from L2ALD.

#### Action

On Router PE1, determine the MAC address of the local IRB interface.

From operational mode, run the `show interfaces irb extensive | match "Current address"` command.

```
user@PE1> show interfaces irb extensive | match
"Current address"
Current address: a8:d0:e5:54:0d:10, Hardware address: a8:d0:e5:54:0d:10
```

From operational mode, run the `show route table evpn.evpn.0 extensive | find "a8:d0:e5:54:0d:10"` command.

```
user@PE1> show route table evpn.evpn.0 extensive
| find "a8:d0:e5:54:0d:10"
2:10.255.0.1:100::0::100::a8:d0:e5:54:0d:10/384 (1 entry, 1 announced)
TSI:
Page 0 idx 0, (group PE type Internal) Type 1 val 0x2736568 (adv_entry)
Advertised metrics:
Flags: Nexthop Change
```

```

Nexthop: Self
Localpref: 100
AS path: [100] I
Communities: target:100:100 evpn-default-gateway
Path 2:10.255.0.1:100::0::100::a8:d0:e5:54:0d:10 Vector len 4. Val: 0
 *EVPN Preference: 170
 Next hop type: Indirect
 Address: 0x26f8354
 Next-hop reference count: 6
 Protocol next hop: 10.255.0.1
 Indirect next hop: 0x0 - INH Session ID: 0x0
 State: <Active Int Ext>
 Age: 23:29:08
 Validation State: unverified
 Task: evpna-evpn
 Announcement bits (1): 1-BGP_RT_Background
 AS path: I
 Communities: evpn-default-gateway
 Route Label: 299776

```

## Meaning

The MAC-only route for the local IRB interface appears in the EVPN instance route table on Router PE1 and is learned from EVPN and tagged with the default gateway extended community.

## Verifying Remote IRB MACs

### Purpose

Verify that the remote IRB MACs are learned from BGP.

### Action

On Router PE1, determine the MAC address of the local IRB interface.

From operational mode, run the `show interfaces irb extensive | match "Current address"` command.

```

user@PE1> show interfaces irb extensive | match
"Current address"
Current address: a8:d0:e5:54:0d:10, Hardware address: a8:d0:e5:54:0d:10

```

On Router PE2, verify that the remote IRB MACs are learned.

From operational mode, run the `show route table evpna.evpn.0 extensive | find "a8:d0:e5:54:0d:10"` command.

```

user@PE2> show route table evpna.evpn.0 extensive
| find "a8:d0:e5:54:0d:10"
2:10.255.0.1:100::0::100::a8:d0:e5:54:0d:10/384 (1 entry, 1 announced)
 *BGP Preference: 170/-101
 Route Distinguisher: 2.91.223.24:100
 Next hop type: Indirect
 Address: 0x26f8d6c
 Next-hop reference count: 10
 Source: 10.255.0.1
 Protocol next hop: 10.255.0.1
 Indirect next hop: 0x2 no-forward INH Session ID: 0x0
 State: <Secondary Active Int Ext>
 Local AS: 100 Peer AS: 100
 Age: 23:22:17 Metric2: 1
 Validation State: unverified
 Task: BGP_100.10.255.0.1
 Announcement bits (1): 0-evpna-evpn
 AS path: I
 Communities: target:100:100 evpn-default-gateway
 Import Accepted
 Route Label: 299776
 Localpref: 100
 Router ID: 10.255.0.1
 Primary Routing Table bgp.evpn.0
 Indirect next hops: 1
 Protocol next hop: 10.255.0.1 Metric: 1
 Indirect next hop: 0x2 no-forward INH Session ID: 0x0
 Indirect path forwarding next hops: 1
 Next hop type: Router
 Next hop: 1.0.0.1 via ge-1/0/8.0
 Session Id: 0x1
 10.255.0.1/32 Originating RIB: inet.3
 Metric: 1 Node path count: 1
 Forwarding nexthops: 1
 Nexthop: 1.0.0.1 via ge-1/0/8.0

```



## Meaning

The MAC-only route for the remote IRB interface appears in the EVPN instance route table on Router PE2 and is learned from BGP and tagged with the default gateway extended community.

## Verifying Local IRB IPs

### Purpose

Verify that the local IRB IPs are learned locally by RPD.

### Action

On Router PE1, determine the MAC and IP addresses of the local IRB interface.

From operational mode, run the `show interfaces irb extensive | match "Current address"` command.

```
user@PE1> show interfaces irb extensive | match
"Current address"
Current address: a8:d0:e5:54:0d:10, Hardware address: a8:d0:e5:54:0d:10
```

From operational mode, run the `show interfaces irb.0 terse | match inet` command.

```
user@PE1> show interfaces irb.0 terse | match
inet
irb.0 up up inet 10.0.0.251/24
```

From operational mode, run the `show route table evpna.evpn.0 extensive | find "a8:d0:e5:54:0d:10::10.0.0.251"` command.

```
user@PE2> show route table evpna.evpn.0 extensive
| find "a8:d0:e5:54:0d:10::10.0.0.251"
2:10.255.0.1:100::0::100::a8:d0:e5:54:0d:10::10.0.0.251/384 (1 entry, 1 announced)
TSI:
Page 0 idx 0, (group PE type Internal) Type 1 val 0x27365a0 (adv_entry)
 Advertised metrics:
 Flags: Nexthop Change
 Nexthop: Self
 Localpref: 100
 AS path: [100] I
```

```

Communities: target:100:100 evpn-default-gateway
Path 2:10.255.0.1:100::0::100::a8:d0:e5:54:0d:10::10.0.0.251 Vector len 4. Val: 0
 *EVPN Preference: 170 <<<<
 Next hop type: Indirect
 Address: 0x26f8354
 Next-hop reference count: 6
 Protocol next hop: 10.255.0.1
 Indirect next hop: 0x0 - INH Session ID: 0x0
 State: <Active Int Ext>
 Age: 23:48:46
 Validation State: unverified
 Task: evpna-evpn
 Announcement bits (1): 1-BGP_RT_Background
 AS path: I
 Communities: evpn-default-gateway
 Route Label: 299776

```

## Meaning

The MAC plus IP route for the local IRB interface appears in the EVPN instance route table on Router PE1 and is learned from EVPN and tagged with the default gateway extended community.

## Verifying Remote IRB IPs

### Purpose

Verify that the remote IRB IPs are learned from BGP.

### Action

On Router PE1, determine the MAC and IP addresses of the local IRB interface.

From operational mode, run the `show interfaces irb extensive | match "Current address"` command.

```

user@PE1> show interfaces irb extensive | match
"Current address"
Current address: a8:d0:e5:54:0d:10, Hardware address: a8:d0:e5:54:0d:10

```

From operational mode, run the `show interfaces irb.0 terse | match inet` command.

```
user@PE1> show interfaces irb.0 terse | match
inet
irb.0 up up inet 10.0.0.251/24
```

On Router PE2, verify that the remote IRB IPs are learnt.

From operational mode, run the `show route table evpn.evpn.0 extensive | find "a8:d0:e5:54:0d:10::10.0.0.251"` command.

```
user@PE2> show route table evpn.evpn.0 extensive
| find "a8:d0:e5:54:0d:10::10.0.0.251"
2:10.255.0.1:100::0::100::a8:d0:e5:54:0d:10::10.0.0.251/384 (1 entry, 1 announced)
 *BGP Preference: 170/-101
 Route Distinguisher: 2.91.223.216:100
 Next hop type: Indirect
 Address: 0x26f8d6c
 Next-hop reference count: 10
 Source: 10.255.0.1
 Protocol next hop: 10.255.0.1
 Indirect next hop: 0x2 no-forward INH Session ID: 0x0
 State: <Secondary Active Int Ext>
 Local AS: 100 Peer AS: 100
 Age: 23:56:36 Metric2: 1
 Validation State: unverified
 Task: BGP_100.10.255.0.1
 Announcement bits (1): 0-evpn-evpn
 AS path: I
 Communities: target:100:100 evpn-default-gateway
 Import Accepted
 Route Label: 299776
 Localpref: 100
 Router ID: 10.255.0.1
 Primary Routing Table bgp.evpn.0
 Indirect next hops: 1
 Protocol next hop: 10.255.0.1 Metric: 1
 Indirect next hop: 0x2 no-forward INH Session ID: 0x0
 Indirect path forwarding next hops: 1
 Next hop type: Router
 Next hop: 1.0.0.1 via ge-1/0/8.0
 Session Id: 0x1
```

```

10.255.0.1/32 Originating RIB: inet.3
Metric: 1 Node path count: 1
Forwarding nexthops: 1
Nexthop: 1.0.0.1 via ge-1/0/8.0

```

## Meaning

The MAC plus IP route for the remote IRB interface appears in the EVPN instance route table on Router PE2 and is tagged with the default gateway extended community.

## Verifying CE-CE Inter-Subnet Forwarding

### Purpose

Verify inter-subnet forwarding between Routers CE1 and CE2.

### Action

From operational mode, run the `show route table inet.0` command.

```

user@CE1> show route table inet.0
inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0 *[Static/5] 00:15:09
 > to 10.0.0.251 via ge-1/1/7.0
10.0.0.0/24 *[Direct/0] 1d 23:24:30
 > via ge-1/1/7.0
10.0.0.1/32 *[Local/0] 1d 23:24:38
 Local via ge-1/1/7.0

```

From operational mode, run the `ping` command.

```

user@CE1> ping 198.51.100.2 interval 0.1 count 10
PING 198.51.100.2 (20.0.0.2): 56 data bytes
64 bytes from 198.51.100.2: icmp_seq=0 ttl=63 time=0.919 ms
64 bytes from 198.51.100.2: icmp_seq=1 ttl=63 time=0.727 ms
64 bytes from 198.51.100.2: icmp_seq=2 ttl=63 time=0.671 ms
64 bytes from 198.51.100.2: icmp_seq=3 ttl=63 time=0.671 ms
64 bytes from 198.51.100.2: icmp_seq=4 ttl=63 time=0.666 ms

```

```
64 bytes from 198.51.100.2: icmp_seq=5 ttl=63 time=0.704 ms
64 bytes from 198.51.100.2: icmp_seq=6 ttl=63 time=0.763 ms
64 bytes from 198.51.100.2: icmp_seq=7 ttl=63 time=0.750 ms
64 bytes from 198.51.100.2: icmp_seq=8 ttl=63 time=12.967 ms
64 bytes from 198.51.100.2: icmp_seq=9 ttl=63 time=0.752 ms

--- 198.51.100.2 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.666/1.959/12.967/3.670 ms
```

## Meaning

Ping from Router CE1 to Router CE2 is successful.

## RELATED DOCUMENTATION

| [An EVPN with IRB Solution on EX9200 Switches Overview](#) | 1505

# Configuring IGMP or MLD Snooping with EVPN-MPLS

## IN THIS CHAPTER

- [Overview of Multicast Forwarding with IGMP or MLD Snooping in an EVPN-MPLS Environment | 1565](#)
- [Configure Multicast Forwarding with IGMP Snooping in an EVPN-MPLS Environment | 1580](#)
- [Configure Multicast Forwarding with MLD Snooping in an EVPN-MPLS Environment | 1590](#)

## Overview of Multicast Forwarding with IGMP or MLD Snooping in an EVPN-MPLS Environment

### IN THIS SECTION

- [Benefits of Multicast Forwarding with IGMP or MLD Snooping in an EVPN-MPLS Environment | 1566](#)
- [Multicast Forwarding with IGMP or MLD Snooping in Single-homed or Multihomed Ethernet Segments | 1566](#)
- [IGMP or MLD Snooping with Multicast Forwarding Between Bridge Domains or VLANs Using IRB Interfaces | 1577](#)

To help optimize multicast traffic flow in an Ethernet VPN (EVPN) over MPLS environment, you can enable IGMP snooping for IPv4 multicast traffic or MLD snooping for IPv6 multicast traffic. In this environment, multicast receiver hosts in the EVPN instance (EVI) can be single-homed to one provider edge (PE) device or multihomed in all-active mode to multiple provider edge (PE) devices. For receivers that are multihomed to multiple PE devices, the peer PE devices synchronize IGMP or MLD state information. Receivers can be attached to PE devices in the EVI at the same site or at different sites.

Source hosts can be single-homed or multihomed to PE devices within the EVI in some supported use cases. For other use cases, the sources must reside outside the EVPN network in an external PIM

domain, and each PE device that needs to receive multicast traffic connects through a Layer 3 interface to the PIM gateway router.

You can enable IGMP and MLD snooping for:

- Multiple EVPN instances
- Specific bridge domains or VLANs in an EVPN instance
- All bridge domains or VLANs within an EVPN virtual switch instance (on supporting devices)

Devices in this environment can forward multicast traffic within a bridge domain or VLAN, and route multicast traffic across bridge domains or VLANs at Layer 3 using IRB interfaces.

### **Benefits of Multicast Forwarding with IGMP or MLD Snooping in an EVPN-MPLS Environment**

- In an environment with significant multicast traffic, IGMP or MLD snooping constrains multicast traffic in a broadcast domain or VLAN to interested receivers and multicast devices, which conserves network bandwidth.
- Synchronizing IGMP or MLD state among all EVPN PE devices for multihomed receivers ensures that all subscribed listeners receive multicast traffic, even in cases such as the following:
  - IGMP or MLD membership reports for a multicast group might arrive on a PE device that is not the Ethernet segment's designated forwarder (DF).
  - An IGMP or MLD message to leave a multicast group arrives at a different PE device than the PE device where the corresponding join message for the group was received.

### **Multicast Forwarding with IGMP or MLD Snooping in Single-homed or Multihomed Ethernet Segments**

Hosts in the network send IGMP or MLD reports expressing interest in particular multicast groups from IPv4 multicast sources (using IGMP) or IPv6 multicast sources (using MLD). PE devices with IGMP or MLD snooping enabled listen to (snoop) IGMP or MLD packets. The PEs then use the snooped information to establish multicast routes that only forward traffic to interested receivers for a multicast group.

For redundancy, your network can include multicast receivers multihomed to a set of peer PE devices in all-active mode. In some use cases, sources can be multihomed to peer PE devices within the EVI. If you enable snooping on all PE devices in the EVI, the multihomed peer PE devices synchronize the IGMP or MLD state among themselves so multicast traffic can successfully reach all listeners.

The next sections describe the IGMP and MLD group membership report processing and subsequent multicast traffic flow in this environment.

### **IGMP or MLD State Synchronization Among Multihoming Peer PE Devices**

In an EVI with receivers that are multihomed to multiple PE devices, corresponding IGMP or MLD join and leave messages for multicast group management might not be sent to the same PE device. As a result, all the PE devices must synchronize and share IGMP and MLD state.

PE devices with snooping enabled in this environment exchange BGP EVPN Type 7 (Join Sync Route) and Type 8 (Leave Synch Route) network layer reachability information (NLRI) to synchronize IGMP or MLD membership reports received on multihomed interfaces. The network must use multihoming in all-active mode.

The advertised EVPN Type 7 and Type 8 routes also carry EVI route target extended community attributes associated with multihomed EVIs to support multiple EVPN routing instances simultaneously. Only PE devices that share the same Ethernet segment ID import these routes.

PE devices serving single-homed hosts do not need to advertise or import EVPN Type 7 and Type 8 routes. However, to save bandwidth on the access side by only forwarding multicast traffic to interested receivers, you should also configure IGMP or MLD snooping in this case.

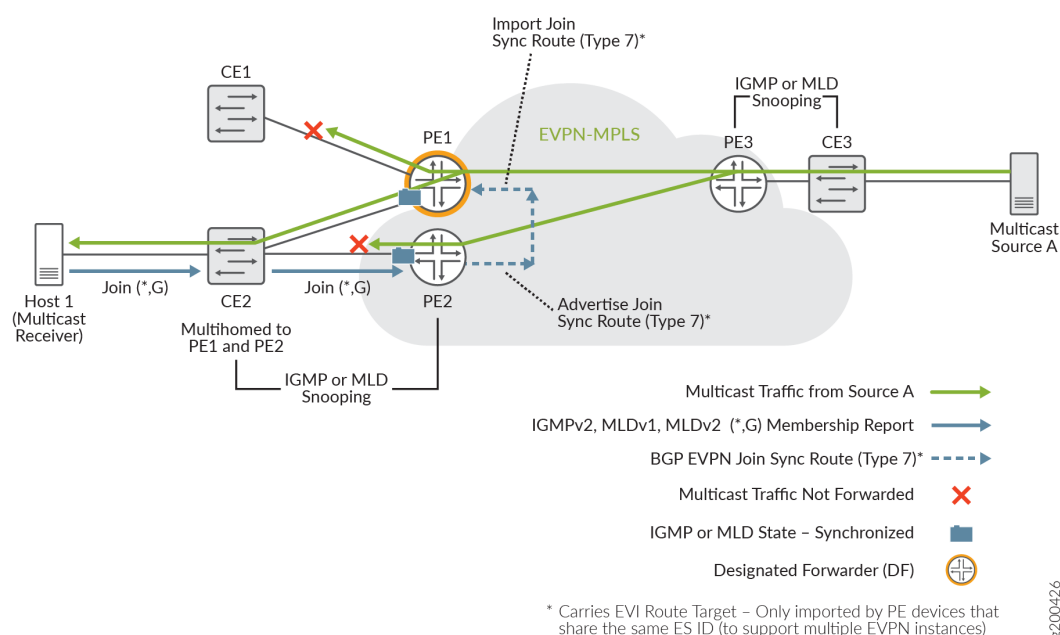
Due to platform-specific differences among the devices that support multicast with IGMP or MLD snooping in an EVPN-MPLS multihomed environment, we offer slightly different solutions on the supporting platforms in certain use cases.

### **EX9200 Switches, MX Series Routers, and vMX Routers—Multicast State Synchronization for Multihoming in EVPN-MPLS**

EX9200 switches, MX Series routers, and vMX routers support EVPN-MPLS multihoming with IGMP and MLD snooping and synchronize the IGMP or MLD state as shown in [Figure 152 on page 1568](#).



**Figure 152: Multicast Forwarding for Multihomed Receivers in EVPN-MPLS on EX9200 Switches, MX Series Routers, and VMX Routers**



This example topology includes a multicast source (Multicast Source A) behind PE3 by way of customer edge device CE3, and a multicast receiver Host 1 behind CE2 that is multihomed to PE1 and PE2.

Multihoming peer PE2 receives a membership request from Host 1, and advertises the EVPN Type 7 Join Sync Route to the EVPN core. PE1 imports the Type 7 route to synchronize the IGMP or MLD state among the PE devices to which the receiver is multihomed (PE1 and PE2). When both PE1 and PE2 receive multicast traffic from the EVPN-MPLS network, only PE1, the designated forwarder (DF), forwards the traffic to CE2 to reach the interested listener on Host 1. PE1 doesn't forward the traffic to CE1, which has no interested listeners.

EX9200 switches, MX Series routers and vMX routers support IGMP and MLD snooping with EVPN-MPLS multihoming under the following constraints:

- All multihoming peer PE devices must support BGP EVPN Type 7 Join Sync Routes and Type 8 Leave Sync Routes.
- You can configure IGMP snooping or MLD snooping for multiple routing instances of type `evpn` or type `virtual-switch`.
- You must configure all bridge domains or VLANs with multicast sources and receivers on all PE devices in the EVI.
- You must configure MX Series PE devices in enhanced IP Network Services mode. (See [Network Services Mode Overview](#).)

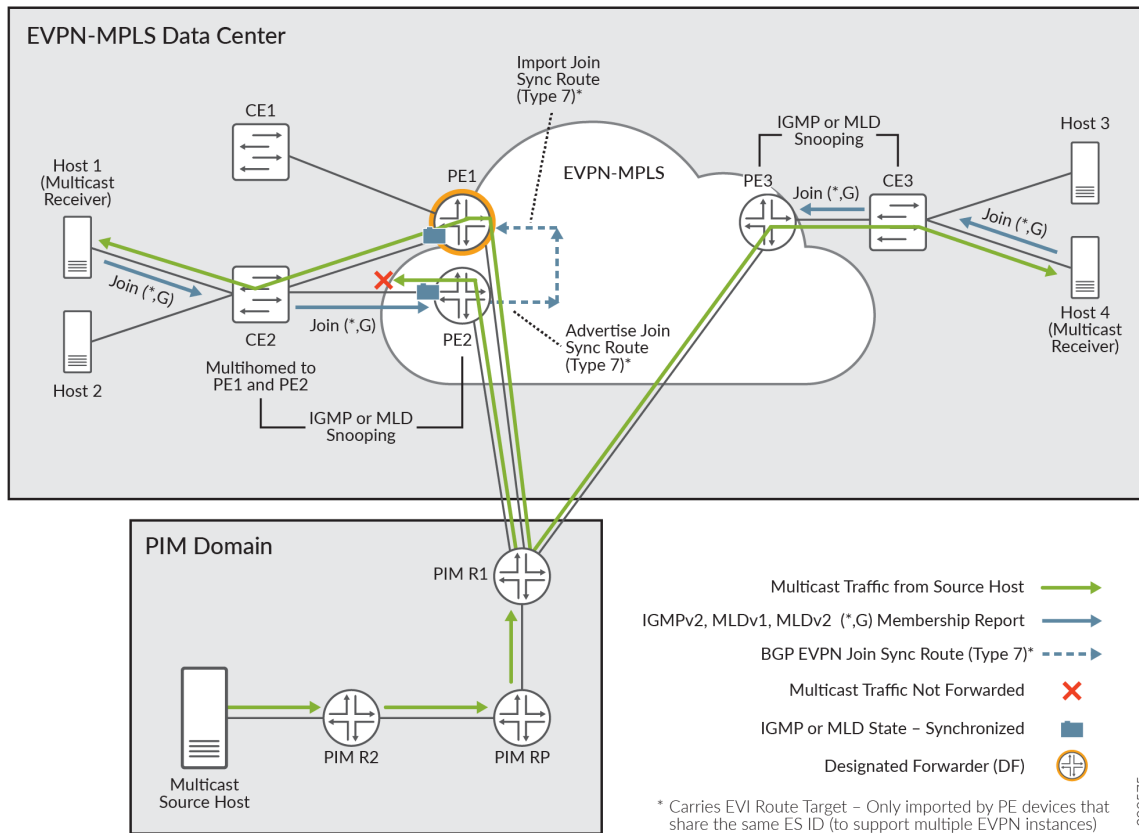
- Routing multicast traffic between bridge domains or VLANs is through IRB interfaces only. You can't have a topology that uses an external multicast router for intersubnet multicast routing in this environment. (Using an external multicast router instead of IRB interfaces is an available solution with EVPN-VXLAN networks.)
- Multicast sources and receivers must be within the EVI. Receiving or forwarding multicast traffic from or to devices outside of this environment using a PIM gateway is not supported.
- Selective multicast forwarding (advertising EVPN Type 6 Selective Multicast Ethernet Tag (SMET) routes for forwarding only to interested receivers) in the EVPN core is not supported.
- IGMP and MLD snooping are not supported with point-to-multipoint (P2MP) multipoint LDP/RSVP replication; only ingress replication is supported.

### **ACX Series Routers—Multicast State Synchronization for Multihoming in EVPN-MPLS**

On ACX Series routers with EVPN-MPLS multihoming and IGMP and MLD snooping, to support routing multicast traffic across bridge domains or VLANs (intersubnet multicast), any multicast sources must be outside of the EVPN-MPLS datacenter in a Layer 3 PIM domain. (An ACX Series router can't act as a multicast first-hop router, so multicast sources can't be connected to ACX routers for intersubnet routing.) Each ACX Series PE device must receive the multicast source traffic through a Layer 3 interface connected to the external PIM domain gateway router. See [Figure 153 on page 1570](#). If your EVPN-MPLS datacenter only needs to forward multicast traffic within bridge domains or VLANs (intrasubnet multicast), you can use a topology similar to [Figure 152 on page 1568](#) with sources inside the datacenter.

In both the intrasubnet and intersubnet use cases, IGMP or MLD state synchronization among multihoming peer PE devices works the same as the EVPN-MPLS multicast with multihoming solutions for other platforms.

Figure 153: Multicast Forwarding for Multihomed Receivers in EVPN-MPLS on ACX Series Routers



The example topology in [Figure 153 on page 1570](#) can support both intrasubnet and intersubnet multicast traffic because it includes a multicast source (Multicast Source Host) in an external PIM domain, and Layer 3 connections from the PE devices to the external PIM gateway router. Multicast receiver Host 1 behind CE2 is multihomed to peer PE devices PE1 and PE2. Single-homed multicast receiver Host 4 connects to PE3 by way of CE3. All of the EVPN PEs serving multicast receivers have IGMP (or MLD) snooping enabled. To optimize multicast forwarding on the access side, you can also enable IGMP or MLD snooping on the CE devices (if supported).

Multihoming peer PE2 receives a membership request from Host 1, and advertises the EVPN Type 7 Join Sync Route to the EVPN core. PE1 imports the Type 7 route to synchronize the IGMP or MLD state among the PE devices to which the receiver is multihomed (PE1 and PE2). PE3 receives a membership request from single-homed Host 4, and with no multihoming peers, doesn't need to synchronize IGMP or MLD state information.

ACX Series routers support IGMP and MLD snooping with EVPN-MPLS multihoming under the following constraints:

- All multihomed peer PE devices must support BGP EVPN Type 7 Join Sync Routes and Type 8 Leave Sync Routes.

- You can configure IGMP snooping or MLD snooping for one or more routing instances of type evpn only.
- You must configure all multicast bridge domains or VLANs on all PE devices in the EVI.
- Routing multicast traffic between bridge domains or VLANs is through IRB interfaces only. You can't have a topology that uses an external multicast router for intersubnet multicast routing instead of IRB interfaces. (Using an external multicast router instead of IRB interfaces is an available solution with EVPN-VXLAN networks on some platforms.)
- As mentioned previously, to support intersubnet multicast traffic routing, multicast sources must reside outside the data center in an external PIM domain. All PEs connect to the PIM gateway and send PIM join messages through the PIM rendezvous point (RP). The PIM gateway forwards traffic for the group to all PE devices.



**NOTE:** In this environment, multicast sources can be within the EVI only when the multicast traffic for a group is all within the same bridge domain or VLAN (intrasubnet multicast only).

- Selective multicast forwarding (advertising EVPN Type 6 Selective Multicast Ethernet Tag (SMET) routes for forwarding only to interested receivers) in the EVPN core is not supported.
- IGMP and MLD snooping are not supported with point-to-multipoint (P2MP) multipoint LDP/RSVP replication; only ingress replication is supported.

## Leave Message Processing with Multihoming Peer PE Devices

Processing leave messages and membership route withdrawals in a multihomed environment is more complicated when the leave message is not received by the same PE device that processed the join message. PE devices use BGP EVPN Type 8 routes to facilitate leave operations as follows:

- A PE device that receives an IGMP or MLD leave message for a group advertises a Type 8 route. Other PE devices import the Type 8 route.
- The PE device that advertised the Type 8 route originates a membership query for any remaining group members, and starts a leave timer. The other PE devices that imported the Type 8 route likewise start a leave timer.
- If no join membership reports are received by the time the timer expires, the PE device that advertised the Type 7 route withdraws the Type 7 route. The PE device that originated the Type 8 route withdraws the Type 8 route.

## IGMP or MLD Versions and Supported Group Membership Report Modes

Any-source multicast (ASM) or (\*,G) group membership mode includes all sources for a multicast group's traffic. Source-specific multicast (SSM) or (S,G) mode handles multicast group membership based on a specific source and the group address.

By default, PE devices in this EVPN-MPLS environment support ASM mode with IGMPv2, IGMPv3, MLDv1, and MLDv2 under certain conditions. You can alternatively configure PE devices to process only SSM membership reports with IGMPv3 and MLDv2 in environments that support intersubnet traffic. PE devices don't support processing both ASM and SSM membership reports at the same time.

[Table 85 on page 1572](#) summarizes membership report modes supported with each IGMP or MLD version. IGMPv1 is not supported with EVPN-MPLS multicast.

**Table 85: Group Membership Report Modes Supported for Each IGMP and MLD Version**

| IGMP or MLD Version | ASM (*,G) Only                                                                                   | SSM (S,G) Only                                          | ASM (*,G) + SSM (S,G) |
|---------------------|--------------------------------------------------------------------------------------------------|---------------------------------------------------------|-----------------------|
| IGMPv1              | -                                                                                                | -                                                       | -                     |
| IGMPv2              | Yes                                                                                              | No                                                      | No                    |
| IGMPv3              | Yes<br>(Only on ACX Series)                                                                      | Yes<br>(Only on ACX Series, when explicitly configured) | No                    |
| MLDv1               | Yes                                                                                              | No                                                      | No                    |
| MLDv2               | Yes<br>(ACX Series: Only with MLDv2 configured on all interfaces that receive multicast traffic) | Yes<br>(Only when explicitly configured)                | No                    |

The next sections give more details on platform-specific PE device behavior when processing IGMP or MLD membership reports.

## ASM and SSM Mode Behavior on EX9200 Switches, MX Series Routers, and vMX Routers

By default, the EVPN-MPLS network can process ASM (\*,G) membership reports with IGMPv2, MLDv1, and MLDv2. If the network has hosts sending both MLDv1 and MLDv2 ASM reports for a given group, PE devices will process MLDv1 and MLDv2 reports for the group as MLDv1 membership reports. They drop and do not process any SSM reports.

You can alternatively enable IGMP snooping or MLD snooping with the `evpn-ssm-reports-only` option to explicitly configure PEs to accept and process SSM (S,G) membership reports with MLDv2. With this option, PEs only accept SSM reports, and drop and do not process any ASM reports.

- You can enable SSM-only processing for one or more bridge domains in the EVI.
- When enabling this option for a virtual switch, this behavior applies to all bridge domains in the virtual switch instance.

## ASM and SSM Mode Behavior on ACX Series Routers

With IGMP snooping enabled and multicast traffic flow is *within* bridge domains or VLANs (intrasubnet):

- By default, PE devices process IGMPv2 ASM (\*,G) membership reports, but discard any IGMPv3 reports.
- If you configure all interfaces that receive multicast traffic with IGMPv3, PE devices can process IGMPv3 reports in ASM (\*,G) mode, but they discard IGMPv3 SSM (S,G) reports.

With MLD snooping enabled and multicast traffic flow is *within* bridge domains or VLANs (intrasubnet):

- By default, PE devices can process MLDv1 ASM (\*,G) membership reports but discard any MLDv2 reports.
- If you configure all interfaces that receive multicast traffic with MLDv2, PE devices can process both MLDv1 and MLDv2 reports in ASM (\*,G) mode, but they discard MLDv2 SSM (S,G) reports.

With IGMP snooping or MLD snooping enabled and routing multicast traffic *between* bridge domains or VLANs (intersubnet):

- With IGMP snooping, by default, PE devices process IGMPv2 and IGMPv3 membership reports in ASM (\*,G) mode. They discard IGMPv3 SSM (S,G) reports.
- With MLD snooping, by default, PE devices process MLDv1 and MLDv2 membership reports in ASM (\*,G) mode. They discard MLDv2 SSM (S,G) reports.
- If you enable IGMP snooping or MLD snooping with IGMPv3 or MLDv2 and the SSM-only processing option, `evpn-ssm-reports-only`, then PE devices process IGMPv3 or MLDv2 reports as SSM (S,G) only. They discard any ASM (\*,G) reports.

In this case, for best results, we recommend that you configure IGMPv3 or MLDv2 on all IRB interfaces that do intersubnet routing.

## How Multicast Traffic Forwarding Works with Single-homed or Multihomed Receivers

In an EVPN-MPLS network where hosts might be multihomed to more than one PE device, when a bridge domain (or VLAN) is configured on a PE device, the PE device signals a BGP EVPN Type 3 (Inclusive Multicast Ethernet Tag [IMET]) route to the other PE devices in the instance to build a core multicast replication tree for each configured bridge domain.

A PE device receiving multicast traffic to be forwarded is referred to as the *ingress PE device*. To ensure multicast traffic reaches all remote PE devices in the EVI, the ingress PE device uses the IMET routing information with ingress replication in the EVPN core, replicating and flooding the packets on the EVPN tunnels to all of the other PE devices (or external edge routers) in the EVI that might need to forward the traffic. IGMP or MLD snooping is not performed in the EVPN core.



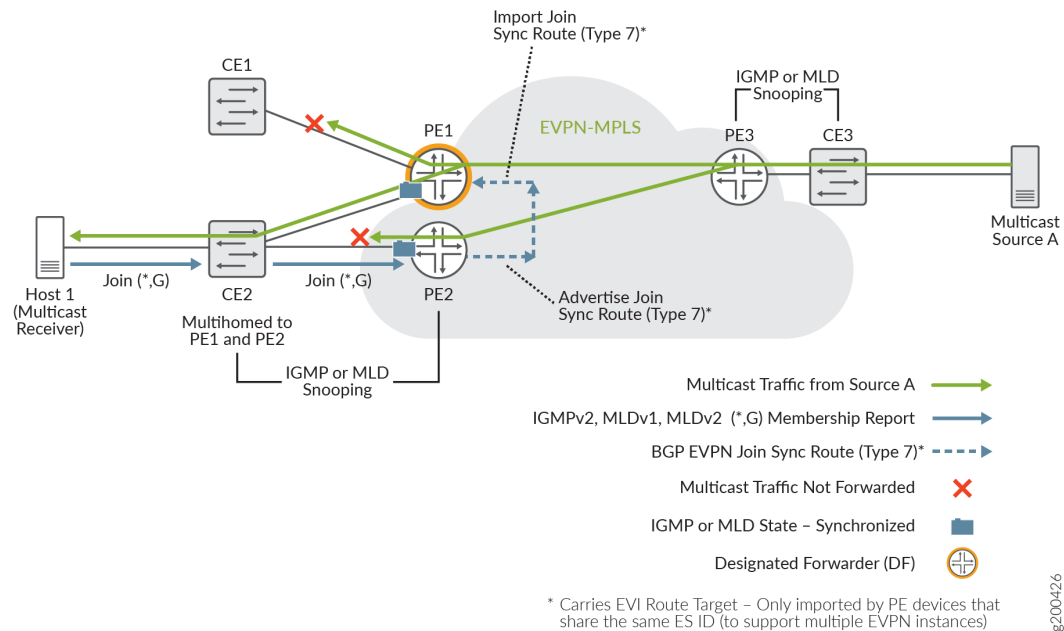
**NOTE:** When the multicast source is outside the EVPN-MPLS network in an external PIM domain (as it is in the implementation for ACX Series routers), each PE device receives the source traffic directly from the configured Layer 3 interfaces to the PIM gateway router, and not by way of IMET signaling in the EVPN core.

With IGMP or MLD snooping enabled:

- When a multihoming PE device receives multicast traffic and it is the DF for an interested receiver for the multicast group, the PE device forwards the traffic.
- When a multihoming PE device receives multicast traffic and it is not the DF for any interested receivers, the PE device does not forward the traffic.
- On the access side, upon receiving multicast data from the EVPN core, PE devices selectively forward the multicast traffic only to interested receivers. Single-homing PE devices use learned IGMP or MLD snooping information, and multihoming PE devices use both IGMP or MLD snooping information and EVPN Type 7 routes.

For example, let's look again at the use case in [Figure 154 on page 1575](#) with sources and receivers all inside the EVPN-MPLS network.

Figure 154: Multicast Traffic Flow with Sources Inside the EVPN-MPLS Network



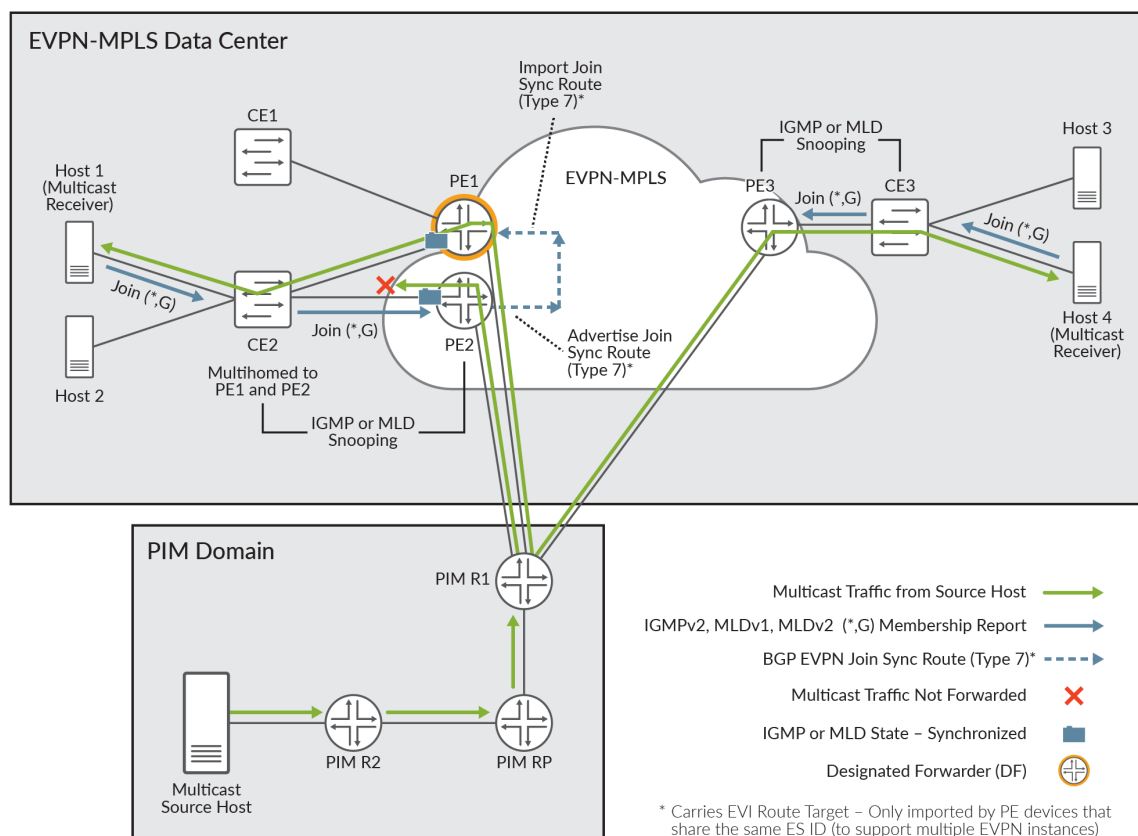
PE3 is the ingress PE device for Multicast Source A and forwards the multicast traffic into the EVPN core. PE1 and PE2 receive the multicast traffic from the EVPN core. Host 1 is a multicast receiver that is multihomed to PE1 and PE2 by way of customer edge device CE2. PE1 and PE2 have synchronized IGMP or MLD state information about receiver Host 1. The PEs forward the traffic as follows:

- Because PE1 is currently the elected DF for Host 1, PE1 forwards the data toward Host 1.
- Based on the IGMP or MLD snooping information and imported EVPN Type 7 routing information, PE1 doesn't forward the data toward CE1 because CE1 doesn't have any interested receivers.
- PE2 is not the DF, so PE2 doesn't forward the traffic toward Host 1, and avoids sending a duplicate stream.

Also consider the use case in [Figure 155 on page 1576](#) for ACX Series routers with EVPN-MPLS multihoming that supports multicast across bridge domains or VLANs, where the sources must be outside the EVPN network in an external PIM domain.



**Figure 155: Multicast Traffic Flow With Sources Outside the EVPN-MPLS Network**



PE1, PE2, and PE3 all have Layer 3 connections to the PIM gateway router PIM R1, and all PE devices are ingress PEs. In this case, all PEs receive the multicast source traffic from the source through PIM R1. Host 3 and Host 4 are single-homed to PE3 by way of CE3. Host 1 is a multicast receiver that is multihomed to PE1 and PE2 through CE2. The PEs forward the traffic as follows:

- Host 4 is a multicast receiver, so PE3 sends the traffic to CE3 to reach Host 4.
- Host 3 isn't a multicast receiver, so with snooping enabled, CE3 doesn't forward the traffic to Host 3.
- PE1 and PE2 have synchronized IGMP or MLD state information about receiver Host 1, and as peer PE devices, they behave in the same way as the use case with all sources and receivers inside the EVPN-MPLS network:
  - As the elected DF for Host 1, PE1 forwards the data to Host 1. PE2 is not the DF and doesn't send a duplicate stream to Host 1.
  - Based on the IGMP or MLD snooping information and imported EVPN Type 7 routing information, PE1 doesn't forward the data toward CE1 because CE1 doesn't have any interested receivers.

## IGMP or MLD Snooping with Multicast Forwarding Between Bridge Domains or VLANs Using IRB Interfaces

For multicast forwarding between bridge domains or VLANs in this environment, PE devices use Protocol Independent Multicast (PIM) in distributed designated router (DDR) mode on IRB interfaces.

- You must configure IRB interfaces on all PE devices in the EVI that need to route multicast traffic locally between bridge domains or VLANs on the device.



**NOTE:** If you configured a bridge domain (or VLAN) and an IRB interface on all PE devices, and an IRB interface is down on a PE device, traffic routing between bridge domains (or VLANs) for receivers served by that IRB will be impacted.

- To route Layer 3 multicast traffic between bridge domains or VLANs, you must also configure PIM distributed designated router (DDR) mode on all participating IRB interfaces. (See [distributed-dr](#) for more details.) PIM passive mode is not supported.

The IRB interfaces on the PE devices route multicast traffic between bridge domains or VLANs as follows:

- Upon receiving multicast traffic on an IRB interface from a multicast source, the PE device routes the traffic to any IRBs that have PIM enabled and are configured with bridge domains or VLANs that have interested local receivers for the multicast group.
- In PIM DDR mode, the IRB interfaces route multicast traffic to local receivers whether or not the IRB is the elected PIM designated router (DR).
- To prevent multicast traffic duplication, IRB-routed multicast traffic is not forwarded back out to the EVPN core.

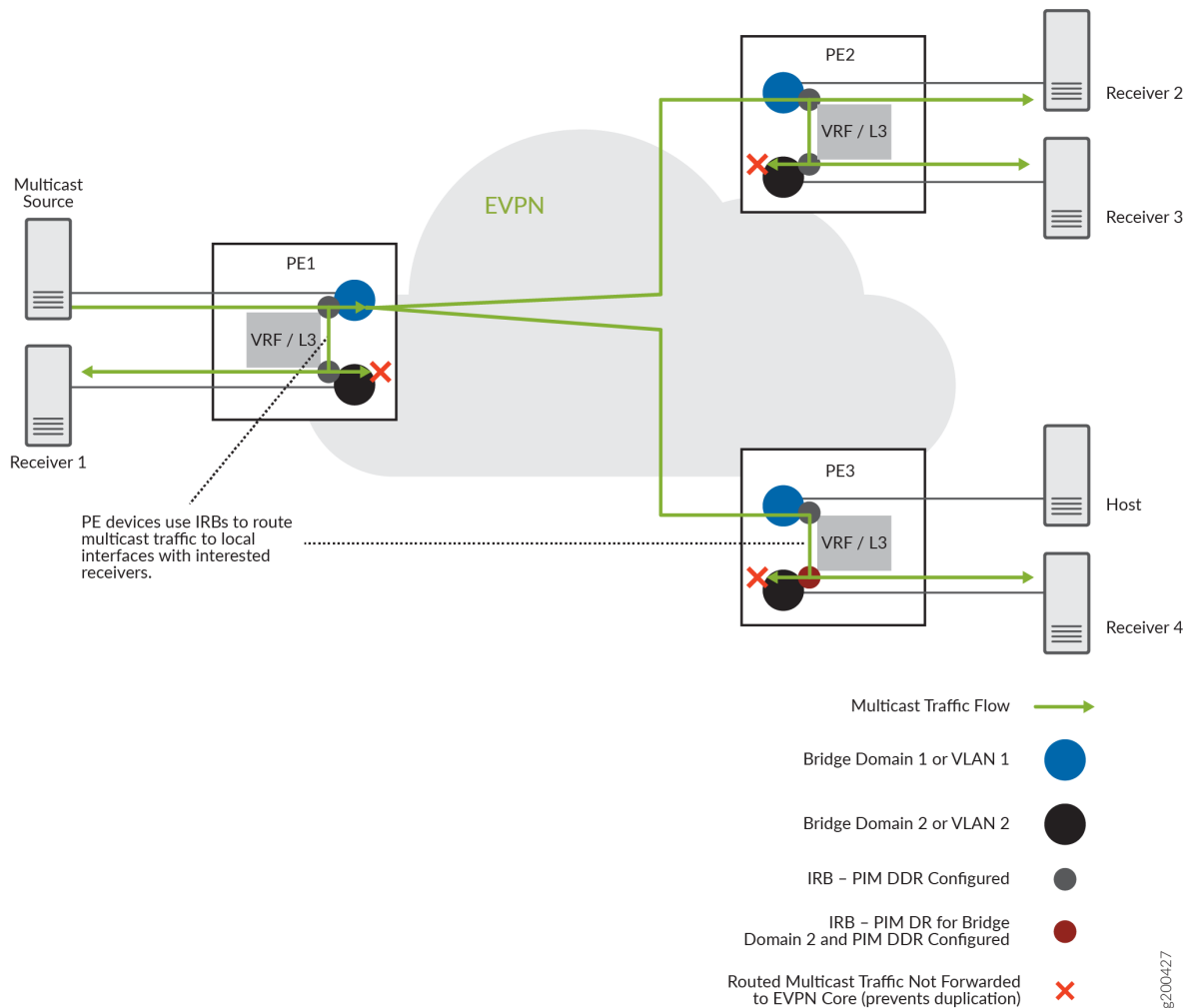
The IRB interfaces route ingress multicast traffic between bridge domains or VLANs toward receivers in the same way whether the multicast source is inside the EVPN-MPLS network or is in an external PIM domain.

For example, [Figure 156 on page 1578](#) shows PE devices with IRB interfaces where the PE devices receive source traffic through the EVPN-MPLS network.



**NOTE:** ACX Series routers support do not support having multicast sources within the EVPN-MPLS network for routing inter-VLAN multicast traffic. See [Figure 157 on page 1579](#) instead.

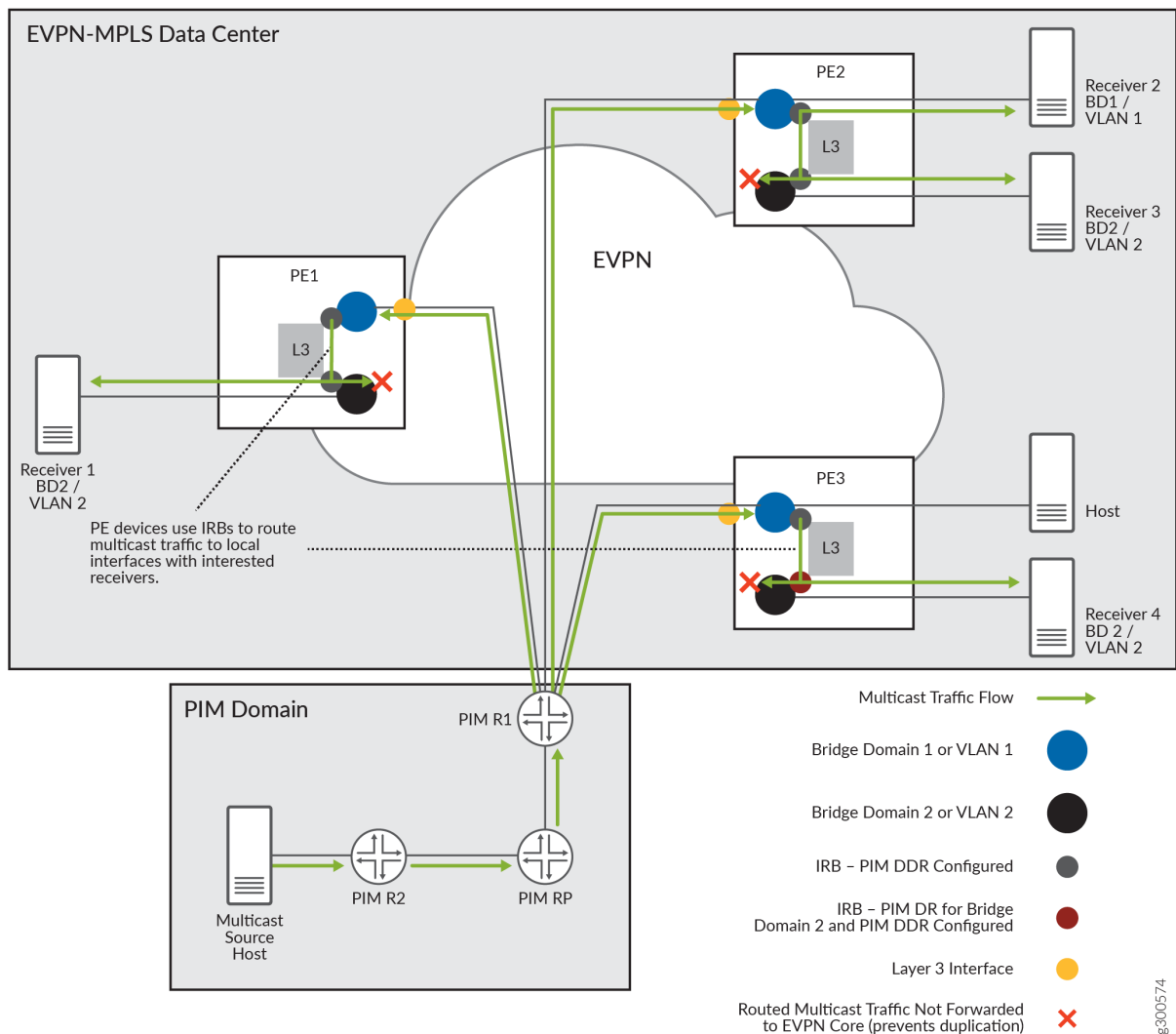
**Figure 156: Routing Multicast Traffic Between Bridge Domains (or VLANs) with IRBs and PIM DDR in EVPN-MPLS**



In this use case, the EVPN instance has three PE devices, PE1, PE2, and PE3, with IRB interfaces configured on each PE device for two bridge domains or VLANs. The multicast source on bridge domain 1 (VLAN 1) sends multicast traffic to PE1, and PE1 uses inclusive multicast forwarding with ingress replication to forward the traffic to the EVPN core to reach other PE devices with interested receivers. PE1 routes the traffic locally to interested receivers on bridge domain 2 (VLAN 2) through the IRB interfaces, even though PE3 is the PIM DR for bridge domain 2 (VLAN 2). The IRB interfaces on all PE devices that route traffic to receivers on bridge domain 2 (VLAN 2) do not forward the routed traffic back out into the EVPN core. PE2 receives the multicast traffic from the EVPN core and forwards or locally routes the traffic on each bridge domain (or VLAN) that has interested receivers. PE3 routes the traffic locally to bridge domain 2 only, because there are no interested receivers on bridge domain 1 (or VLAN 1).

Figure 157 on page 1579 shows the PE devices similarly configured with IRB interfaces in DDR mode, but each of the PE devices receives the multicast traffic from an external source through Layer 3 interfaces connected to the PIM gateway router. This is the use case supported on ACX Series routers for routing inter-VLAN multicast traffic in an EVPN-MPLS network.

**Figure 157: Routing External PIM Domain Source Traffic Between Bridge Domains (or VLANs) with IRBs and PIM DDR in EVPN-MPLS**



In this use case, PE1, PE2, and PE3 also have IRB interfaces configured on each PE device for two bridge domains or VLANs. Each PE might have interested receivers, so each one has a Layer 3 connection to the source PIM domain and receives the multicast traffic on BD1 (VLAN 1). PE1 and PE3 each route the traffic locally through the IRB interfaces to their receivers (Receiver 1 and Receiver 4) on BD2 (VLAN 2), and PE2 does the same for its Receiver 3 on BD2, even though PE3 is the PIM DR for BD2 (VLAN 2).

PE2 also forwards the traffic locally to Receiver 2 on BD1 (VLAN 1). The PE devices don't send source traffic out into the EVPN core at all in this model.

## RELATED DOCUMENTATION

[Configure Multicast Forwarding with IGMP Snooping in an EVPN-MPLS Environment | 1580](#)

[Configure Multicast Forwarding with MLD Snooping in an EVPN-MPLS Environment | 1590](#)

[EVPN Multihoming Overview | 162](#)

## Configure Multicast Forwarding with IGMP Snooping in an EVPN-MPLS Environment

### IN THIS SECTION

- [Configure IGMP Snooping for an EVPN or Virtual Switch Routing Instance | 1581](#)
- [Configure IGMP Snooping with IGMPv3 on ACX Series Routers to Process Source-Specific Multicast Group Membership Reports Only | 1583](#)
- [Configure Multicast Routing Across Bridge Domains or VLANs with PIM in EVPN-MPLS | 1584](#)
- [Viewing IGMP Snooping Multicast Information for EVPN-MPLS in the CLI | 1587](#)

IGMP snooping with multicast forwarding ensures that IPv4 multicast traffic reaches all subscribed receivers within and between bridge domains or VLANs, and preserves bandwidth on the access side by reducing the amount of multicast control and data traffic being forwarded.

You can configure multicast with IGMP snooping on provider edge (PE) devices in an Ethernet VPN (EVPN) over MPLS environment. You must enable IGMP snooping in this environment to support IPv4 multicast traffic for receivers that are multihomed to EVPN PE devices as follows:

- Peer PE devices must be operating in all-active multihoming mode.
- You enable IGMP snooping in proxy mode.
- You configure and commit the same configuration on each multihoming peer PE device.



**NOTE:** MX Series routers and EX9200 switches in this environment only support IGMPv2 with any-source multicast (ASM) membership reports. ACX Series routers in this environment support processing IGMPv2 and IGMPv3 membership reports in ASM mode and IGMPv3 membership reports in source-specific multicast (SSM) mode with a special configuration option. See ["IGMP or MLD Versions and Supported Group Membership Report Modes"](#) on page 1572 for details on how PEs process IGMP reports.

In addition, to route multicast traffic between bridge domains or VLANs, PEs with hosts in multicast groups that span bridge domains or VLANs use PIM distributed designated router (PIM DDR) mode on IRB interfaces. You set up IRB interfaces associated with each bridge domain or VLAN with interested receivers in a multicast group. With PIM DDR configured on the IRB interfaces, the PE devices send the traffic locally through the IRB interfaces to their interested receivers on the corresponding bridge domains or VLANs even if an IRB is not the elected PIM designated router (DR) for that bridge domain or VLAN.

This topic describes configuration tasks to set up IGMP snooping in proxy mode, and configure the IRB interfaces for routing between bridge domains or VLANs on EVPN PE devices. It also summarizes the CLI commands you can use to verify IGMP snooping and multicast operation in this environment.

## Configure IGMP Snooping for an EVPN or Virtual Switch Routing Instance

On ACX Series routers, you can configure IGMP snooping on one or more routing instances of type `evpn`. On other types of devices, you can configure IGMP snooping for routing instances of type `evpn` or `virtual-switch`.

- To configure IGMP snooping in proxy mode on a PE device for a routing instance with instance-type `evpn` (for any or all bridge domains or VLANs):

```
user@device# set routing-instances routing-instance-name protocols igmp-snooping proxy
```

For example, the following configuration includes enabling IGMP snooping in proxy mode for EVPN routing instance `evpn-A` configured as instance-type `evpn`:

```
routing-instances {
 evpn-A {
 instance-type evpn;
 vlan-id 600
 interface ge-0/0/2.600;
 route-distinguisher 10.255.255.1:100;
 vrf-target target:64510:100;
 protocols {
```

```

 evpn {
 interface ge-0/0/2.600;
 label-allocation per-instance;
 }
 igmp-snooping proxy;
 }
}

```

- To configure IGMP snooping on a PE device for specific bridge domains or VLANs for an EVPN instance with `instance-type virtual-switch`:

```

user@device# set routing-instances routing-instance-name bridge-domain bridge-domain-name
protocols igmp-snooping proxy

```

For example, the following configuration includes enabling IGMP snooping in proxy mode for bridge domains V200, V201, and V202 in EVPN routing instance EVPN-2, which is configured as `instance-type virtual-switch`:

```

routing-instances {
 ...
 EVPN-2 {
 instance-type virtual-switch;
 route-distinguisher 90.90.90.10:2;
 vrf-target target:64510:2;
 protocols {
 evpn {
 encapsulation mpls;
 extended-vlan-list 200-202;
 default-gateway advertise;
 }
 }
 bridge-domains {
 V200 {
 domain-type bridge;
 vlan-id 200;
 interface ae1.200;
 routing-interface irb.200;
 protocols {
 igmp-snooping proxy;
 }
 }
 }
 }
}

```

```

 }
 V201 {
 domain-type bridge;
 vlan-id 201;
 interface ae1.201;
 routing-interface irb.201;
 protocols {
 igmp-snooping proxy;
 }
 }
 V202 {
 domain-type bridge;
 vlan-id 202;
 interface ae1.202;
 routing-interface irb.202;
 protocols {
 igmp-snooping proxy;
 }
 }
}
...
}

```

## Configure IGMP Snooping with IGMPv3 on ACX Series Routers to Process Source-Specific Multicast Group Membership Reports Only

By default, the EVPN-MPLS network processes only ASM (\*,G) membership reports with IGMPv2. ACX Series PE routers also support processing IGMPv3 membership reports in ASM (\*,G) mode. You can alternatively configure ACX Series PEs to process only IGMPv3 SSM (S,G) membership reports in an EVPN-MPLS network that supports intersubnet multicast. To do this, use the `evpn-ssm-reports-only` option in the [edit protocols `igmp-snooping`] configuration statement hierarchy when you configure IGMP snooping.

When you enable this option, the device doesn't process ASM reports and drops those packets. See ["IGMP or MLD Versions and Supported Group Membership Report Modes" on page 1572](#) for details on how PEs process IGMP membership reports in this environment.

You can enable SSM-only processing when you configure IGMP snooping with IGMPv3 for all bridge domains or VLANs in a routing instance of type `evpn`.



For example, to configure IGMP snooping with IGMPv3 to process only SSM membership reports for all bridge domains or VLANs in a routing instance:

```
user@device# set routing-instances routing-instance-name protocols igmp-snooping evpn-ssm-reports-only
```

In the following sample configuration, IGMP snooping is configured to process only SSM membership reports for a routing instance of type `evpn` named `EVPN-4`:

```
.
.
.
EVPN-4 {
 instance-type evpn;
 protocols {
 evpn {
 remote-ip-host-routes;
 designated-forwarder-preference-least;
 }
 igmp-snooping {
 evpn-ssm-reports-only;
 proxy;
 }
 }
 vlan-id 400;
 interface ae2.400;
 l3-interface irb.400;
 route-distinguisher 90.90.90.10:2;
 vrf-target target:64510:4;
}
.
.
.
```

## Configure Multicast Routing Across Bridge Domains or VLANs with PIM in EVPN-MPLS

PE devices in EVPN-MPLS environments use IRB interfaces and PIM to route multicast traffic between bridge domains or VLANs (intersubnet multicast).

On ACX Series routers, any multicast sources must be outside of the EVPN-MPLS data center in a Layer 3 PIM domain. Each ACX Series PE device receives the multicast source traffic through a Layer 3 interface connected to an external PIM domain gateway router. The receivers must be within the EVPN-MPLS datacenter.

With other platforms, all multicast sources and receivers must be within the EVPN-MPLS data center, and each PE device either receives the multicast traffic locally or through the EVPN core.

In either case (multicast sources outside or inside the EVPN instance), you enable PIM in distributed DR mode on the IRB interfaces in an EVPN instance, and the IRB interfaces send the multicast group traffic to any receivers on the associated bridge domains or VLANs.

- To configure IRB interfaces to use PIM DDR mode to route multicast traffic between bridge domains or VLANs in an EVPN routing instance:

```
user@device# set routing-instances routing-instance-name protocols pim interface irb-
interface-name distributed-dr
```

For example, the following configuration shows PIM DDR mode enabled on interfaces irb.400, irb.401, and irb.402 in an instance-type evpn routing instance named evpn-400 on ACX Series routers (which require a Layer 3 interface on which to receive source traffic from an external PIM domain):

```
routing-instances {
 evpn-400 {
 instance-type evpn;
 vlan-id 400;
 interface xe-0/0/32.400;
 l3-interface irb.400;
 route-distinguisher 10.255.65.227:400;
 vrf-target target:64510:400;
 protocols {
 evpn {
 interface xe-0/0/32.400;
 }
 igmp-snooping proxy;
 }
 }
 protocols {
 pim {
 rp {
 static {
 address 10.255.67.48;
 }
 }
 }
 }
}
```

```

 }
 }
 interface lo0.0;
 interface all {
 mode {
 sparse;
 }
 }
 ...
 interface irb.400 {
 distributed-dr;
 }
 interface irb.401 {
 distributed-dr;
 }
 interface irb.402 {
 distributed-dr;
 }
}
}
...
}

```

The following configuration example shows PIM DDR mode enabled on interfaces irb.200, irb.201, and irb.202 in the VRF routing instance IPVPN-2:

```

routing-instances {
 ...
 IPVPN-2 {
 instance-type vrf;
 interface irb.200;
 interface irb.201;
 interface irb.202;
 interface lo0.2;
 route-distinguisher 90.90.90.10:222;
 vrf-target target:64510:2;
 vrf-table-label;
 protocols {
 pim {
 rp {
 local {

```

```

 address 10.88.88.10;
 }
}
interface irb.200 {
 distributed-dr;
}
interface irb.201 {
 distributed-dr;
}
interface irb.202 {
 distributed-dr;
}
}
}
}

```

## Viewing IGMP Snooping Multicast Information for EVPN-MPLS in the CLI

The following EVPN commands are supported to view IGMP snooping multicast information in an EVPN-MPLS environment. The output of these commands includes information learned from native IGMP snooping on a PE device and learned from EVPN Type 7 Join Sync Route and Type 8 Leave Sync Route messages.

- `show evpn igmp-snooping database`
- `show evpn multicast-snooping next-hops`
- `show igmp snooping evpn database`
- `show igmp snooping evpn membership`

The following CLI commands are supported to view the native IGMP snooping information on a PE. The output of these commands will not include information learned from the exchange of EVPN Type 7 and Type 8 IGMP messages.

- `show igmp snooping interface`
- `show igmp snooping membership`
- `show igmp snooping statistics`

The following commands can be used to view EVPN Type 7 IGMP Join Sync Routes or Type 8 Leave Sync Routes in BGP and EVPN routing tables:

- `show route table bgp.evpn.0 match-prefix 7*`

```

show route table __default_evpn__.evpn.0 match-prefix 7:*

show route table __default_evpn__.evpn.0 match-prefix 7:* protocol evpn

show route table __default_evpn__.evpn.0 match-prefix 7:* protocol bgp

```

- ```
show route table bgp.evpn.0 match-prefix 8:*
```

```

show route table __default_evpn__.evpn.0 match-prefix 8:*

show route table __default_evpn__.evpn.0 match-prefix 8:* protocol evpn

show route table __default_evpn__.evpn.0 match-prefix 8:* protocol bgp

```

For example, to view EVPN Type 7 IGMP routes in the __default_evpn__.evpn.0 routing table, use the following command:

```

user@host> show route table __default_evpn__.evpn.0 protocol evpn match-prefix 7:* extensive
Sep 21 02:25:12
__default_evpn__.evpn.0: 32 destinations, 32 routes (32 active, 0 holddown, 0 hidden)
7:90.90.90.10:1::1111111111111111::100::0.0.0.0::233.252.0.1::90.90.90.10/600 (1 entry, 1
announced)
TSI:
Page 0 idx 0, (group IBGP type Internal) Type 1 val 0xb23c68c (adv_entry)
  Advertised metrics:
    Nexthop: 90.90.90.10
    Localpref: 100
    AS path: [64510] I
    Communities: es-import-target:11-11-11-11-11-11 evi-rt:64510:1 Path
7:90.90.90.10:1::1111111111111111::100::0.0.0.0::233.252.0.1::90.90.90.10 Vector len 4. Val: 0
  *EVPN Preference: 170
    Next hop type: Indirect, Next hop index: 0
    Address: 0xb2e5e10
    Next-hop reference count: 61
    Protocol next hop: 90.90.90.10
    Indirect next hop: 0x0 - INH Session ID: 0x0
    State: <Active Int Ext>
    Age: 2:12:30
    Validation State: unverified
    Task: __default_evpn__-evpn
    Announcement bits (1): 1-BGP_RT_Background
    AS path: I
    Communities: es-import-target:11-11-11-11-11-11 evi-rt:65530:1
    IGMP flags: 0x2

```

```

7:90.90.90.10:3::333333333333333333::100::0.0.0.0::233.252.0.1::90.90.90.10/600 (1 entry, 1
announced)
TSI:
Page 0 idx 0, (group IBGP type Internal) Type 1 val 0xb23c6a8 (adv_entry)
  Advertised metrics:
    Nexthop: 90.90.90.10
    Localpref: 100
    AS path: [64510] I
    Communities: es-import-target:33-33-33-33-33-33 evi-rt:64510:3 Path
7:90.90.90.10:3::333333333333333333::100::0.0.0.0::233.252.0.1::90.90.90.10 Vector len 4. Val: 0
  *EVPN Preference: 170
    Next hop type: Indirect, Next hop index: 0
    Address: 0xb2e5e10
    Next-hop reference count: 61
    Protocol next hop: 90.90.90.10
    Indirect next hop: 0x0 - INH Session ID: 0x0
    State: <Active Int Ext>
    Age: 2:12:30
    Validation State: unverified
    Task: __default_evpn__-evpn
    Announcement bits (1): 1-BGP_RT_Background
    AS path: I
    Communities: es-import-target:33-33-33-33-33-33 evi-rt:65530:3
    IGMP flags: 0x2

```

RELATED DOCUMENTATION

[Overview of Multicast Forwarding with IGMP or MLD Snooping in an EVPN-MPLS Environment | 1565](#)

Configure Multicast Forwarding with MLD Snooping in an EVPN-MPLS Environment

IN THIS SECTION

- [Configure MLD Snooping for Default Any-Source Multicast \(ASM\) Group Membership Processing with MLDv1 or MLDv2 | 1591](#)
- [Configure MLD Snooping with MLDv2 to Process Source-Specific Multicast Group Membership Reports Only | 1594](#)
- [Viewing MLD Snooping Multicast Information for EVPN-MPLS in the CLI | 1595](#)

MLD snooping with multicast forwarding ensures that IPv6 multicast traffic reaches all subscribed receivers within and between bridge domains or VLANs, and preserves bandwidth on the access side by reducing the amount of multicast control and data traffic being forwarded.

You can configure multicast with MLD snooping on provider edge (PE) devices in an Ethernet VPN (EVPN) over MPLS environment. You must enable MLD snooping in this environment to support IPv6 multicast traffic for receivers that are multihomed to EVPN PE devices as follows:

- Peer PE devices must be operating in all-active multihoming mode.
- You enable MLD snooping in proxy mode.
- You configure and commit the same configuration on each multihoming peer PE device.



NOTE: PEs in this environment support processing MLDv1 and MLDv2 membership reports in any-source multicast (ASM) mode and MLDv2 membership reports in source-specific multicast (SSM) mode with a special configuration option. See ["IGMP or MLD Versions and Supported Group Membership Report Modes" on page 1572](#) for details on how PEs process MLD reports.

In addition, to route multicast traffic between bridge domains or VLANs, PEs with hosts in multicast groups that span bridge domains or VLANs use PIM distributed designated router (PIM DDR) mode on IRB interfaces. With PIM DDR configured on the IRB interfaces, all the PE devices send the traffic locally through the IRB interfaces to their interested receivers on the corresponding bridge domains or VLANs even if an IRB is not the elected PIM designated router (DR) for that bridge domain or VLAN. See ["Configure Multicast Routing Across Bridge Domains or VLANs with PIM in EVPN-MPLS" on page 1584](#) for details.

This topic describes configuration tasks to set up MLD snooping on PE devices in a multihoming EVPN-MPLS environment with either the default ASM group membership report processing (MLDv1 and MLDv2) or SSM-only group membership report processing (MLDv2). This topic also summarizes the CLI commands you can use to verify MLD snooping and multicast operation in this environment.

Configure MLD Snooping for Default Any-Source Multicast (ASM) Group Membership Processing with MLDv1 or MLDv2

By default, the EVPN-MPLS network processes only ASM (*,G) membership reports with MLDv1 and MLDv2.

On ACX Series routers, you can configure MLD snooping only on one or more routing instances of type `evpn`. On other types of devices, you can configure MLD snooping with ASM for routing instances of type `evpn` or `virtual-switch` and for all bridge domains or VLANs or only for a specific bridge domain or VLAN.

For example:

- To configure MLD snooping in proxy mode on PE devices in an EVPN-MPLS network for the default-switch routing instance:

```
user@device# set protocols mld-snooping proxy
```

- To configure MLD snooping in proxy mode on a PE device for a particular routing instance configured as `instance-type evpn` or `instance-type virtual-switch` for all bridge domains or VLANs in the instance:

```
user@device# set routing-instances routing-instance-name protocols mld-snooping proxy
```

- To configure MLD snooping on a PE device for a specific bridge domain or VLAN for an EVPN instance with `instance-type virtual-switch`:

```
user@device# set routing-instances routing-instance-name bridge-domain bridge-domain-name protocols mld-snooping proxy
```

PE devices can't process both ASM (*,G) reports and SSM (S,G) reports at the same time, but you can alternatively configure PEs to process only SSM (S,G) membership reports. See ["Configure MLD Snooping with MLDv2 to Process Source-Specific Multicast Group Membership Reports Only"](#) on page 1594.

In the following sample configuration, MLD snooping is enabled in proxy mode (and the same for IGMP snooping) for an instance of type evpn named EVPN-1:

```
.
.
.
    EVPN-1 {
        instance-type evpn;
        protocols {
            evpn {
                remote-ip-host-routes;
                designated-forwarder-preference-least;
            }
            igmp-snooping {
                proxy;
            }
            mld-snooping {
                proxy;
            }
        }
        vlan-id 100;
        interface ae1.100;
        l3-interface irb.100;
        route-distinguisher 90.90.90.10:1;
        vrf-target target:64510:1;
    }
.
.
.
```

In following sample configuration, MLD snooping is enabled for three bridge domains (V100, V200, V300) in a routing instance of type virtual-switch named CUST-2:

```
.
.
.
    CUST-2 {
        instance-type virtual-switch;
        route-distinguisher 10.255.255.1:100;
    }
```

```

vrf-target target:64510:100;
protocols {
    evpn {
        extended-vlan-list [ 100 200 300 ];
    }
}
bridge-domains {
    V100 {
        domain-type bridge;
        vlan-id 100;
        interface ae0.100;
        routing-interface irb.100;
        protocols {
            igmp-snooping;
            mld-snooping;
        }
    }
    V200 {
        domain-type bridge;
        vlan-id 200;
        interface ae0.200;
        routing-interface irb.200;
        protocols {
            igmp-snooping;
            mld-snooping;
        }
    }
    V300 {
        domain-type bridge;
        vlan-id 300;
        interface ge-0/0/5.300;
        routing-interface irb.300;
        protocols {
            igmp-snooping;
            mld-snooping;
        }
    }
}
}
.
.
.

```

Configure MLD Snooping with MLDv2 to Process Source-Specific Multicast Group Membership Reports Only

By default, the EVPN-MPLS network processes only ASM (*,G) membership reports with MLDv1 and MLDv2. You can alternatively configure PEs to process only MLDv2 SSM (S,G) membership reports. To do this, use the `evpn-ssm-reports-only` option in the `[edit protocols mld-snooping]` configuration statement hierarchy when you configure MLD snooping.

When you enable this option, the device doesn't process ASM reports and drops those packets. See ["IGMP or MLD Versions and Supported Group Membership Report Modes" on page 1572](#) for details on how PEs process MLD reports in this environment.

You can enable SSM-only processing when you configure MLD snooping for:

- All bridge domains or VLANs in a routing instance of type `evpn`.
- All bridge domains or a specific bridge domain or VLAN in a `virtual-switch` instance.



NOTE: On ACX Series routers, you can configure MLD snooping only on one or more routing instances of type `evpn`. On other devices, you can configure MLD snooping for routing instances of type `evpn` or `virtual-switch`.

For example:

- To configure MLD snooping to process only SSM membership reports with MLDv2 for the default-switch EVPN routing instance:

```
user@device# set protocols mld-snooping evpn-ssm-reports-only
```

- To configure MLD snooping with MLDv2 to process only SSM membership reports for all bridge domains or VLANs in a routing instance:

```
user@device# set routing-instances routing-instance-name protocols mld-snooping evpn-ssm-reports-only
```

- To configure MLD snooping in proxy mode to process only SSM membership reports for a specific bridge domain or VLAN for an EVPN instance with instance-type `virtual-switch`:

```
user@device# set routing-instances routing-instance-name bridge-domain bridge-domain-name protocols mld-snooping evpn-ssm-reports-only proxy
```

In following sample configuration, MLD snooping is configured to process only SSM membership reports for a routing instance of type evpn named EVPN-2:

```
.
.
.
    EVPN-2 {
        instance-type evpn;
        protocols {
            evpn {
                remote-ip-host-routes;
                designated-forwarder-preference-least;
            }
            mld-snooping {
                evpn-ssm-reports-only;
                proxy;
            }
        }
        vlan-id 200;
        interface ae2.200;
        l3-interface irb.200;
        route-distinguisher 90.90.90.10:2;
        vrf-target target:64510:1;
    }
.
.
.
```

Viewing MLD Snooping Multicast Information for EVPN-MPLS in the CLI

The following EVPN commands are supported to view MLD snooping multicast information in an EVPN-MPLS environment. The output of these commands includes information learned from native MLD snooping on a PE device and learned from EVPN Type 7 Join Sync Route and Type 8 Leave Sync Route messages.

- `show evpn mld-snooping database`
- `show evpn multicast-snooping next-hops`
- `show mld snooping evpn database`
- `show mld snooping evpn membership`

RELATED DOCUMENTATION

Overview of Multicast Forwarding with IGMP or MLD Snooping in an EVPN-MPLS Environment | 1565

4

PART

EVPN E-LAN

- [EVPN E-LAN Services | 1598](#)
-

EVPN E-LAN Services

IN THIS CHAPTER

- [EVPN E-LAN Overview | 1598](#)
- [Supported Routing Instance Types and Services for EVPN-ELAN | 1599](#)
- [EVPN E-LAN over SRv6 | 1602](#)

EVPN E-LAN Overview

An Ethernet LAN (E-LAN), defined by the Metro Ethernet Forum (MEF), is a multipoint-to-multipoint transparent Layer 2 (L2) VLAN service that connects two or more user network interfaces (UNIs). The E-LAN provides full mesh connectivity for the UNI sites. A UNI is the dividing point between the responsibilities of the service provider and subscriber. Every UNI can communicate with any other UNI that is connected to the E-LAN service.

Ethernet VPN (EVPN) E-LAN is a framework for delivering multipoint-to-multipoint VPN service with the EVPN signaling mechanisms. E-LAN service allows service providers to offer services that manage the L2 learning very efficiently. In a multihoming scenario, the broadcast, unknown unicast, and multicast (BUM) traffic is handled by the provider edge (PE) device, acting as a designated forwarder (DF). The learned information is redistributed to other PEs in the network. The multihomed customer edge (CE) device connects a customer site to two or more PE devices providing redundant services.

The MEF standard has two different services for EVPN E-LAN:

- Ethernet Private LAN (EP-LAN), which offers a multipoint-to-multipoint ethernet virtual connection (EVC) between dedicated UNIs. EP-LAN is a port-based service.
- Ethernet Virtual Private LAN (EVP-LAN), which offers VLAN based service multiplexing, which means EVCs are paired per UNI.

Supported Routing Instance Types and Services for EVPN-ELAN

IN THIS SECTION

- Supported EVPN E-LAN Instance Types | 1599
- Supported EVPN E-LAN Services | 1599
- Junos OS and Junos OS Evolved Configuration Differences | 1600

Supported EVPN E-LAN Instance Types

EVPN E-LAN instance types supported across all Juniper platforms are:

- mac-vrf—Provides all 3 services (vlan-based, vlan-bundle, vlan-aware bundle) associated with a single EVPN instance (EVI).
- evpn—Provides vlan-based and vlan-bundle services for an EVI.
- virtual-switch—Provides vlan-aware bundle service for an EVI.



NOTE: ACX Series devices running Junos OS Evolved support only the `mac-vrf` instance type.

Supported EVPN E-LAN Services

Table 86: Supported EVPN E-LAN Services

Services	MAC-VRF	EVPN	VIRTUAL-SWITCH
Control word enabled by default (see <i>control-word</i>)	Yes	No	No
VLAN normalization or no normalization (see <i>no-normalization</i>)	Yes	Yes	No

Table 86: Supported EVPN E-LAN Services (Continued)

Services	MAC-VRF	EVPN	VIRTUAL-SWITCH
Default <i>normalization</i> (<i>Protocols EVPN</i>)	no-normalization	normalization	N/A
Full RFC7432 compliance	Yes	Yes	Yes
Core isolation (see "Understanding When to Disable EVPN-VXLAN Core Isolation" on page 514)	Yes	Yes	Yes

Junos OS and Junos OS Evolved Configuration Differences

See this section for configuration differences between normalization and no-normalization on Junos OS and Junos OS Evolved.

Consider the following examples where two provider edge (PE) devices represent different ESIs, and their connected customer edge (CE) devices are part of the same VLAN.

[Table 87 on page 1601](#) depicts use cases with no normalization to prevent VLAN translation for traffic transiting between PEs. The device that uses the `evpn` instance type must specify `no-normalization`. The device that uses the `mac-vrf` instance type doesn't specify normalization since that is the configuration default.

Table 87: Configuration Differences with No Normalization

Junos OS	Junos OS Evolved
<pre> user@PE3# show routing-instances evpn-vlan-based { instance-type evpn; protocols { evpn; } vlan-id none; no-normalization; . . . } </pre>	<pre> user@PE1# show routing-instances evpn-vlan-based { instance-type mac-vrf; protocols { evpn { no-control-word; encapsulation mpls; } } service-type vlan-based; . . . } </pre>

[Table 88 on page 1601](#) illustrates a scenario where normalization is desired. The device that uses the evpn instance type has normalization as the device's default behavior. On the device that uses the mac-vrf instance type, you must specify normalization.

Table 88: Configuration Differences with Normalization

Junos OS	Junos OS Evolved
<pre> user@PE3# show routing-instances evpn-vlan-based { instance-type evpn; protocols { evpn; } vlan-id 20; . . . } </pre>	<pre> user@PE1# show routing-instances evpn-vlan-based { instance-type mac-vrf; protocols { evpn { normalization; no-control-word; encapsulation mpls; } } service-type vlan-based; . . . } </pre>

EVPN E-LAN over SRv6

SUMMARY

This article shows the necessary steps to configure segment routing over IPv6 (SRv6), using an Ethernet VPN (EVPN) Ethernet LAN (E-LAN) network.

IN THIS SECTION

- [Overview | 1602](#)
- [Configure EVPN E-LAN | 1602](#)
- [Configure SRv6 | 1603](#)

Overview

EVPN E-LAN serves as a framework for delivering multipoint-to-multipoint VPN service using EVPN signaling mechanisms. The egress provider edge (PE) device signals an SRv6 segment identifier (SID) with the VPN route. The ingress PE encapsulates the SRv6 SID in the VPN packet using an outer IPv6 header. The destination address is the SRv6 SID advertised by the egress PE, and is routable in the IPv6 underlay. The nodes between the PEs only need to support plain IPv6 forwarding. We support SRv6 micro-SID (uSID) and segment routing header (SRH) based control planes. Different endpoint behaviors are defined for SRv6 services on the egress node. See [RFC8986](#) for information regarding the various endpoint behaviors.

Configure EVPN E-LAN

You can configure your EVPN routing instance following the steps below. You'll need to provide details specific to your implementation.

- Configure a MAC-VRF routing instance with the name of your choice.

```
set routing-instances <instance-name> instance-type mac-vrf
```

- Configure the E-LAN service. Here we're using `vlan-based`, however you can also use `vlan-aware` or `vlan-bundle`. If you require more than one VLAN, then `vlan-based` won't work.

```
set routing-instances <instance-name> service-type vlan-based
```

- Configure one or more VLAN IDs to fit your implementation.

```
set routing-instances <instance-name> vlan-id vlan
```

- Configure one or more VLANs to fit your implementation. You may configure more than one interface per VLAN, depending on your needs.

```
set routing-instances <instance-name> vlans <vlan-name> vlan-id <vlan>
set routing-instances <instance-name> vlans <vlan-name> interface <interface>
```

- Configure a unique RD and VRF target.

```
set routing-instances <instance-name> route-distinguisher <value>
set routing-instances <instance-name> vrf-target <target:x:x>
```

Configure SRv6

- Configure SRv6 encapsulation in EVPN.

```
set routing-instances <instance-name> protocols evpn encapsulation srv6
```

- Configure the SRv6 locator type in EVPN. You can choose either SRH (end-dt2-sid) or uSID (micro-dt2-sid).

```
set routing-instances <instance-name> protocols evpn source-packet-routing srv6 locator
<locator-name> (end-dt2-sid | micro-dt2-sid)
```

- Configure one or more locator blocks of various sizes. Juniper recommends a prefix length of 64. A format example, using private addressing, would look like `fd33:7ce6:27bc:168b::/64`.

```
set routing-options source-packet-routing srv6 locator <locator-name> <locator-block>
```

- Depending on the devices in your configuration, you should keep the following in mind:
 - (ACX Series) Only a 32-bit block size and a 16-bit uSID size are supported.

- (ACX Series) The same locator configured under [edit protocols isis source-packet-routing srv6 locator *locator-name* micro-node-sid], and under [edit routing-instances *instance-name* mac-vrf protocols evpn source-packet-routing srv6 locator *locator-name*] is not supported.
- Configure additional supporting statements for SRv6.

```
set chassis network-services enhanced-ip;  
  
set interfaces lo0.0 family inet6 address <ipv6-address>;  
  
set routing-options resolution preserve-nexthop-hierarchy;
```

- Check and commit your configuration.

RELATED DOCUMENTATION

dynamic-tunnels

multipath (Protocols BGP)

5

PART

EVPN-VPWS

- [Configuring VPWS Service with EVPN Mechanisms | 1606](#)
-

Configuring VPWS Service with EVPN Mechanisms

IN THIS CHAPTER

- Overview of VPWS with EVPN Signaling Mechanisms | 1606
- Control word for EVPN-VPWS | 1610
- Overview of Flexible Cross-Connect Support on VPWS with EVPN | 1614
- Overview of Headend Termination for EVPN VPWS for Business Services | 1621
- Configuring VPWS with EVPN Signaling Mechanisms | 1629
- Example: Configuring VPWS with EVPN Signaling Mechanisms | 1632
- FAT Flow Labels in EVPN-VPWS Routing Instances | 1657
- Configuring EVPN-VPWS over SRv6 | 1661
- Configuring Micro-SIDs in EVPN-VPWS | 1667
- Configuring EVPN-VPWS over SRv6 with Traffic Engineering | 1674

Overview of VPWS with EVPN Signaling Mechanisms

An Ethernet VPN (EVPN) enables you to connect dispersed customer sites using a Layer 2 virtual bridge. As compared with other types of Layer 2 VPNs, an EVPN consists of customer edge (CE) devices (host, router, or switch) connected to provider edge (PE) routers. The PE routers can include an MPLS edge switch (MES) that acts at the edge of the MPLS infrastructure. Either an MX Series 5G Universal Routing Platform or a standalone switch can be configured to act as an MES. You can deploy multiple EVPNs within a service provider network, each providing network connectivity to a customer while ensuring that the traffic sharing on that network remains private.

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. The service provisioned with these Layer 2 VPNs is known as VPWS. You can configure a VPWS instance on each associated edge device for each VPWS Layer 2 VPN.

An EVPN-VPWS network provides a framework for delivering VPWS with EVPN signaling mechanisms. The advantages of VPWS with EVPN mechanisms are single-active or all-active multihoming capabilities

and support for Inter-autonomous system (AS) options associated with BGP-signaled VPNs. Metro Ethernet Forum (MEF) describes the following two service models for VPWS:

- Ethernet private line (EPL)— EPL provides a point-to-point Ethernet virtual connection (EVC) between a pair of dedicated user-network interfaces (UNIs) that is between a pair of Ethernet segment identifiers (ESIs) with a high degree of transparency.
- Ethernet virtual private line (EVPL)— EVPL provides point-to-point EVC between {ESI, VLAN} pairs. EVPL allows service multiplexing; that is multiple EVCs or Ethernet services per UNI.

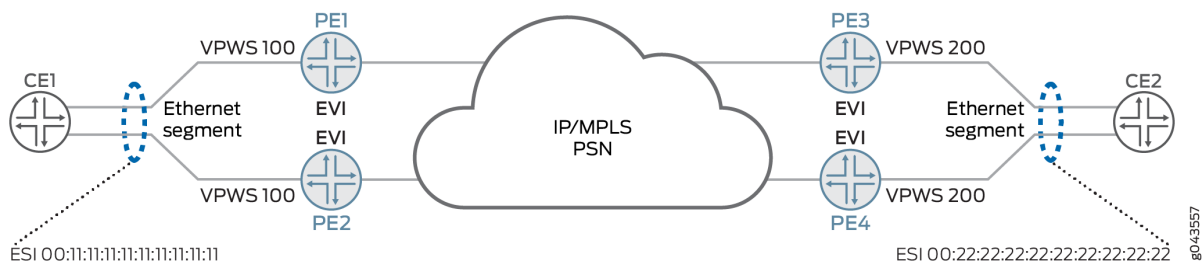
An EVPN-VPWS network is supported on an EVPN-MPLS network.



NOTE: We don't support IGMP snooping, MLD snooping, or PIM snooping multicast optimizations with EVPN-VPWS.

Figure 158 on page 1607 illustrates an EVPN-VPWS network. Device CE1 is multihomed to Routers PE1 and PE2 and Device CE2 is multihomed to Routers PE3 and PE4. Device CE2 has two potential paths to reach CE1, and depending on the multihoming mode of redundancy, only one path or all paths can be active at any one time. When a CE device is multihomed to two or more PE routers, the set of Ethernet links constitutes an Ethernet segment. An Ethernet segment appears as a link aggregation group (LAG) to the CE device. The links from PE1 and PE2 to CE1 and PE3 and PE4 to CE2 form an Ethernet segment.

Figure 158: VPWS in EVPN



An Ethernet segment must have a unique nonzero identifier, called the *Ethernet segment identifier (ESI)*. The ESI is encoded as a 10-octet integer. When manually configuring an ESI value, the most significant octet, known as the *type byte*, must be 00. When a single-homed CE device is attached to an Ethernet segment, the entire ESI value is zero. The Ethernet segment of the multihomed Device CE1 has an ESI value of 00:11:11:11:11:11:11:11:11:11 and the Ethernet segment of the multihomed Device CE2 has an ESI value of 00:22:22:22:22:22:22:22:22:22.

An EVPN instance (EVI) is an EVPN routing and forwarding instance spanning across all the PE routers participating in that VPN. An EVI is configured on the PE routers on a per-customer basis. Each EVI has a unique route distinguisher and one or more route targets. An EVI is configured on PE1, PE2, PE3, and

PE4. An Ethernet tag identifies a particular broadcast domain, such as a VLAN. An EVI consists of one or more broadcast domains. Ethernet tags are assigned to the broadcast domains of a given EVI by the provider of that EVPN. Each PE router in that EVI performs a mapping between broadcast domain identifiers understood by each of its attached CE devices and the corresponding Ethernet tag.

Depending on the multihoming mode of redundancy, only one path or all paths can be active at any one time.

The multihoming mode of operation along with VPWS service identifiers determine which PE router or routers forward and receive traffic in the EVPN-VPWS network. The VPWS service identifier identifies the endpoints of the EVPN-VPWS network. These endpoints are autodiscovered by BGP and are used to exchange the service labels (learned from the respective PE routers) that are used by autodiscovered routes per EVI route type. The service identifier is of two types:

- Local—Unique local VPWS service identifier. This is a logical identifier mapped to a physical interface of a PE router connected to the customer site that is forwarding the traffic to a remote VPWS service identifier.
- Remote—Unique remote VPWS service identifier. This is a logical identifier mapped to a physical interface of a PE router connected to the customer site that is receiving the traffic forwarded from the local VPWS service identifier.

The PE router forwarding traffic to the CE device uses MPLS LSP to forward traffic. If a failure occurs over this path, a new designated forwarder is elected to forward the traffic to the CE device.

The EVPN-VPWS network uses only an autodiscovered route per ESI and an autodiscovered router per EVI route types. In autodiscovered routes per EVI, the 24-bit values of the Ethernet tag are encoded with the VPWS service identifier. The autodiscovered route per ESI is encoded with the route targets of all the EVPN-VPWS instances connected to the ESI on the advertising PE router. When the PE router loses its connectivity to the ESI, it withdraws the autodiscovered route per ESI, resulting in faster convergence. The receiving PE router updates the forwarding next hop of the VPWS service identifier impacted by the failure. Depending on the mode of operation, these two endpoints of the EVPN-VPWS network can be colocated on the same PE router or on different PE routers. The different modes of operation in an EVPN-VPWS network are as follows:

- Local switching—In this mode, the VPWS endpoints (that is, local and remote service identifiers) are connected through the local interfaces configured on the same PE router. Traffic from one CE router is forwarded to another CE router through the same PE router.
- Single-homing—When a PE router is connected to a single-homed customer site, this mode is in operation.
- Active-standby multihoming—In this mode, only a single PE router among a group of PE routers with the same VPWS service identifier attached to an Ethernet segment is allowed to forward traffic to and from that Ethernet segment. The process of electing one among many PE routers with the same VPWS service identifier is known as the *designated forwarder (DF) election*. Each PE router

connected to the Ethernet segment with the same VPWS service identifier participates in the DF election and informs the network of its status. The status can be as follows:

- Designated forwarder(DF)—This is the designated forwarder for forwarding the current traffic.
- Backup designated forwarder(BDF)—This becomes the DF in case the current DF encounters a failure.
- Non-designated forwarder(non-DF)—This is neither the DF nor the BDF. When more than two PE routers are part of an ESI redundancy group, then this PE router becomes a non-DF.

To configure the active-standby mode, include the ESI value and the `single-active` statement under the `[edit interfaces]` hierarchy level. Configure the local and the remote VPWS service identifier under the `[edit routing-instances vpws-routing-instance protocols evpn interface interface-name vpws-service-id]` for each PE router connected to the Ethernet segment.

In [Figure 158 on page 1607](#), for the PE routers connected to Device CE1, Router PE1 is assumed to be the DF and PE2 is assumed to be the BDF for VPWS service identifier 100. For the PE routers connected to CE2, PE3 is assumed to be the DF and Router PE4 is assumed to be the BDF for VPWS service identifier 200. PE2 and PE4 indicate their BDF status to CE1 and CE2 by setting their circuit cross-connect (CCC)-Down flag on their respective interfaces.

- Active-active multi-homing—In active-active multi homing, the CE device is connected to the PE routers with the same local VPWS identifier through the LAG interface so that the traffic is load-balanced among the set of multihomed PE routers with the same remote VPWS service identifiers. Here, election of DF is not required, because all the PE routers connected to the LAG interface participate in forwarding the traffic. In [Figure 158 on page 1607](#), Device CE1 forwards traffic to PE1 and PE2 with VPWS service identifier 100 and CE2 forwards traffic to PE3 and PE4 with VPWS service identifier 200. PE1 forwards the traffic to PE3 and PE4 with remote VPWS service identifier 200. PE2 forwards the traffic to PE3 and PE4. Similarly, traffic from CE2 to PE3 and PE4 with VPWS service identifier is load-balanced across PE routers with VPWS service identifier 100 connected to CE1.

Starting in cRPD Release 20.3R1, EVPN VPWS with EVPN Type 5 is supported to provide VPWS with EVPN signaling mechanisms on cRPD. VPWS with EVPN network is supported on single-homing.

NSR and Unified ISSU Support on VPWS with EVPN

Nonstop active routing (NSR) and graceful Routing Engine switchover (GRES) minimize traffic loss when there is a Routing Engine switchover. When a Routing Engine fails, NSR and GRES enable a routing platform with redundant Routing Engines to switch over from a primary Routing Engine to a backup Routing Engine and continue forwarding packets. Unified in-service software upgrade (ISSU) allows you to upgrade your Junos OS software on your MX Series router with no disruption on the control plane and with minimal disruption of traffic. Both GRES and NSR must be enabled to use unified ISSU.

To enable GRES, include the `graceful-switchover` statement at the `[edit chassis redundancy]` hierarchy level.

To enable NSR, include the `nonstop-routing` statement at the `[edit routing-options]` hierarchy level and the `commit synchronize` statement at the `[edit system]` hierarchy level.

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
20.3R1	Starting in cRPD Release 20.3R1, EVPN VPWS with EVPN Type 5 is supported to provide VPWS with EVPN signaling mechanisms on cRPD. VPWS with EVPN network is supported on single-homing.

RELATED DOCUMENTATION

EVPN Multihoming Overview 162
Configuring VPWS with EVPN Signaling Mechanisms 1629
Example: Configuring VPWS with EVPN Signaling Mechanisms 1632
<code>vpws-service-id</code>
<code>show evpn vpws-instance</code>
<code>instance-type</code>

Control word for EVPN-VPWS

EVPN-VPWS is built on a MPLS network and the transit device's load-balancing hashing algorithm can cause out-of-order delivery of packets. The transit device can incorrectly identify an Ethernet payload as an IPv4 or IPv6 payload if the first nibble of the destination address MAC is 0x4 or 0x6, respectively. By inserting a control word between the label stack and the L2 header of the packet on the MPLS packet switched network, you can ensure that the top nibble is 0, thus preventing the packet from being identified as an IPv4 or IPv6 packet. The PE devices then negotiate support for control word in the EVPN-VPWS service. When you enable control word, the PE devices advertises their support in the auto-discovery route for each EVPN instance (EVI). Before a control word is inserted into the data packet, you must configure all the PE devices in an EVI on the EVPN-VPWS service and all the PE devices in the EVI agree to support control word. If any PE device in an EVI does not support control word, then PE devices will not include the control word in their packet.

Control word is disabled by default on the following platforms:

- EX 9200 switches on Junos OS
- MX series routers on Junos OS

- PTX routers on Junos OS.

To enable control word, use the **set routing-instances *routing-instance-name* protocols evpn control-word** command.

Control word is enabled by default on the following platforms:

- ACX series routers on Junos OS Evolved
- PTX routers on Junos OS Evolved

To disable the control word feature, use the **set routing-instances *routing-instance-name* protocols evpn no-control-word** command.



NOTE: If your transit network comprises only of Juniper EX 9200 switches, MX series routers or PTX series routers on Junos OS, then you do not need to enable control word on the devices. These Juniper devices correctly identify the Ethernet payload as an IPv4/IPv6 payloads, even when the Ethernet destination MAC address starts with 0x4 or 0x6 nibble. The Juniper devices perform hashing based on the IP header fields inside the Ethernet frame and will not send out-of-order packets. In this case, we recommend not using control word as there are no benefits.

Figure 159 on page 1611 and Figure 160 on page 1612 illustrate a network with EVPN-VPWS service terminating in a Layer 3 VPN. In Figure 159 on page 1611, the customer device connects to an access device (A-PE1) which in turn connects to a service-edge device (PE1), that terminates into the layer 3 VPN. You must enable control word on A-PE1 and PE1, so that both devices can advertise their control word support in their route advertisement. Once control word support is established, the PEs will start inserting the control word in their packet.

Figure 159: Single-homed network with EVPN-VPWS service terminating in a Layer 3 VPN

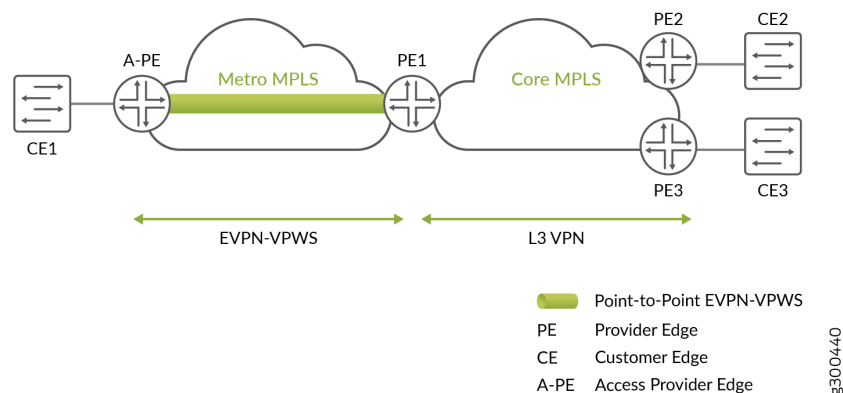
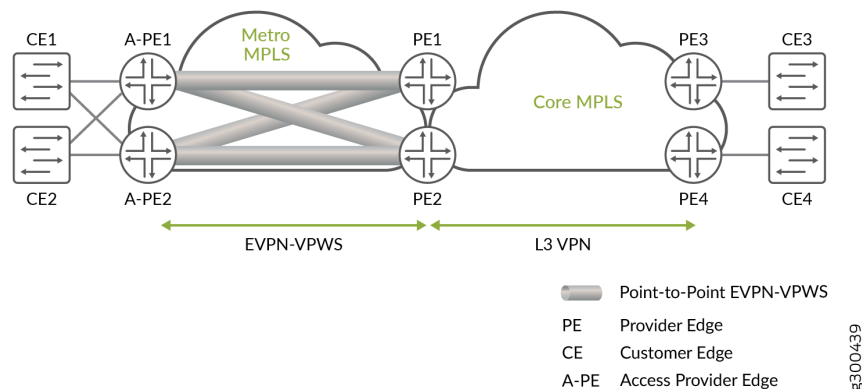


Figure 160 on page 1612 illustrates a topology where the customer device is multihomed to two access devices (A-PE1 and A-PE2), which in turn are multihomed to two service devices (PE1 and PE2). In both single-active and all-active multihoming, you must enable control word on A-PE1, A-PE2, PE1, and PE2 so that the devices can exchange their control word support. When control word support is confirmed for all the PEs in the EVPN-VPWS service, the PEs will start inserting the control word in the packet.

Figure 160: Multihomed network with EVPN-VPWS service terminating in a Layer 3 VPN



To enable control word, set control-word for the evpn protocol for a specified routing instance.

The following output shows a sample multihomed routing instance with control word configured.

```
user@router1# show routing-instances
MHEVPN {
  instance-type evpn-vpws;
  interface ge-0/0/1.0;
  interface ge-0/0/3.100;
  route-distinguisher 10.255.0.1:1;
  vrf-target target:123:123;
  protocols {
    evpn {
      control-word;
      interface ge-0/0/1.0 {
        vpws-service-id {
          local 9999;
          remote 1111;
        }
      }
    }
    interface ge-0/0/3.100 {
      no-control-word;
    }
  }
}
```

```

        vpws-service-id {
            local 500;
            remote 200;
        }
    }
}

```



NOTE: The configuration for the interface takes precedence over the configuration for the EVPN protocol.

To view routes where control word is supported, use the `show route table mpls.0 protocol evpn operational` command. Egress routes display an offset of 252. Ingress routes display an offset of 4. When control word is not enable, the offset is not displayed.

```

show route table mpls.0 protocol evpn
300064          *[EVPN/7] 03:23:31, remote-pe 10.255.0.1, routing-instance mhevpn, route-type
Egress, vlan-id 9999
                > to 10.1.1.2 via ge-0/0/4.0, Push 299840, Push 300768(top) Offset: 252
ge-0/0/1.0      *[EVPN/7] 03:23:27, route-type Egress
                > to 10.1.1.2 via ge-0/0/4.0, Push 299840, Push 300768(top) Offset: 252
...
299984          *[EVPN/7] 03:24:48
                > via ge-0/0/1.0, Pop      Offset: 4
...

```

RELATED DOCUMENTATION

Control Word for BGP VPLS Overview

control-word

no-control-word

Overview of Flexible Cross-Connect Support on VPWS with EVPN

IN THIS SECTION

- [Benefits of Pseudowire Headend Termination \(PWHT\) with EVPN-VPWS FXC Service | 1615](#)
- [FXC on Service Edge Routers | 1615](#)
- [VLAN Unaware FXC on Service Edge Routers | 1616](#)
- [VLAN Aware FXC on Service Edge Routers | 1617](#)
- [VLAN Tagging | 1619](#)
- [Signaling EVPN FXC Optional Bits | 1619](#)
- [Access Side Multihoming | 1619](#)

Ethernet VPN virtual private wire service (EVPN-VPWS) delivers point-to-point Ethernet services between a pair of attachment circuits. An attachment circuit is an access interface participating in an EVPN E-LINE service. EVPN-VPWS also provides multihoming and fast convergence capabilities. With EVPN-VPWS flexible cross-connect (FXC), you can multiplex a large number of attachment circuits across several physical interfaces onto a single VPWS service tunnel.

EVPN-VPWS FXC was introduced as a standard to address label resource issues that can occur on some low end access routers. FXC can bundle a group of attachment circuits under the same EVPN-VPWS instance to share the same MPLS label. You can use FXC to bundle attachment circuits into a single point-to-point Ethernet service group with the normalized VLAN tags used in the data plane. Attachment circuits that share the same label share the same EVPN-VPWS service tunnel.



NOTE: The MPLS label resource issue isn't applicable on provider edge (PE) devices (also called *service edge routers*), MX access routers, or vMX access routers that use pseudowire subscriber logical interfaces. (See *MPLS Pseudowire Subscriber Logical Interfaces* for more on this type of logical interface.)

With FXC, the control plane signaling is based on the exchange of EVPN Ethernet Auto-Discovery (A-D) per EVPN instance (EVI) routes between a pair of PE devices. All the point-to-point Ethernet services under the same EVI are uniquely identified by either a single VLAN tag or double VLAN tags. To ensure the uniqueness, the device must perform VLAN normalization at the ingress. This requirement applies on both PE devices for VLAN unaware and VLAN aware services.

When you use the FXC service on access routers, the forwarding plane uses the MPLS label to find the EVPN-VPWS instance to which a point-to-point Ethernet service belongs. The access router and the

service edge router (PE device) use the VLAN(s) carried in the data packet as the de-multiplexer to uniquely identify each local attachment circuit for a point-to-point Ethernet service.

The two types of EVPN-VPWS FXC services are:

1. VLAN unaware service—The default EVPN-VPWS FXC service.

With this service type, the device signals a single Ethernet A-D per EVI route for each bundle of attachment circuits. The device continues to advertise this route until all attachment circuits in the same group go down. In contrast, a traditional EVPN-VPWS service uses an Ethernet A-D per EVI route for each attachment circuit. The device withdraws that route when the corresponding attachment circuit goes down.

2. VLAN aware service—A VLAN-signaled EVPN-VPWS FXC service.

With this service type, the device advertises one Ethernet A-D per EVI route for each attachment circuit in the same bundle. All the Ethernet A-D per EVI routes for those attachment circuits share the same service label. However, in this case, the device advertises and withdraws each Ethernet A-D per EVI route in the same way as the traditional EVPN-VPWS service. In other words, when a specific attachment circuit goes down, the device withdraws its corresponding Ethernet A-D per EVI route.

Benefits of Pseudowire Headend Termination (PWHT) with EVPN-VPWS FXC Service

EVPN-VPWS FXC is an extension to basic EVPN-VPWS that:

- Bundles different attachment circuits under the same EVI on an access node to share the same EVPN-VPWS service tunnel (also called an EVPN-VPWS pseudowire). In this way, multiple E-LINE services can share the same EVPN-VPWS service tunnel or pseudowire.
- Supports all true VLAN aware bundle services for the VLAN-signaled FXC.
- Uses MPLS label resources more efficiently to support lower-end access devices.

FXC on Service Edge Routers

With PWHT that uses EVPN-VPWS FXC services, the PE device acts as a service edge router. The PE device:

- Performs PWHT through each pseudowire subscriber interface.
- Allocates a label for each pseudowire subscriber interface (with a VLAN unaware FXC service) or for the EVI (with a VLAN aware FXC service).
- Interoperates with the access router, which uses a different label allocation scheme (the access router might use a traditional EVPN-VPWS service or an EVPN-VPWS FXC service).

The pseudowire subscriber transport logical interfaces use encapsulation type ethernet-ccc, and provide a VLAN bundle service.

Similarly, FXC bundles together a group of attachment circuits. We identify each point-to-point Ethernet service in the same group by its unique normalized VLAN(s) in the data plane.

VLAN Unaware FXC on Service Edge Routers

Access-side Single-homing Support for a VLAN Unaware FXC Service

We support single-homed end devices on access-side router attachment circuits. In this case, all attachment circuits in the same EVI are multiplexed (bundled) into a single EVPN-VPWS service tunnel. This bundle shares a service instance ID and MPLS service label. The access routers advertise only one Ethernet A-D per EVI route for the attachment circuit bundle.



NOTE: The access router doesn't withdraw an advertised Ethernet A-D per EVI route for an attachment circuit bundle until either of the following happen:

- All of the attachment circuits in the bundle go down.
- You deactivate or delete the EVI.

To terminate an attachment circuit bundle at the access router, on the PE device you must configure a separate pseudowire subscriber transport logical interface with the corresponding service instance ID. Do this for each bundle of access interfaces at the access router. The pseudowire subscriber transport logical interface works in VLAN bundle mode.

Access-side Multihoming Support for VLAN Unaware FXC Service

Attachment circuits that share an Ethernet segment identifier (ESI) are bundled together. Attachment circuits in the same group share the same local service instance ID. With FXC multihoming, the access router advertises a separate Ethernet A-D route for each multihomed Ethernet segment. The same PE device acts as the designated forwarder (DF) or non-designated forwarder (NDF) for the attachment circuits in the same Ethernet segment at the same time. However, note that the same PE might also be the DF and NDF independently for different Ethernet segments.

You must use different pseudowire subscriber transport logical interfaces to terminate different multihomed attachment circuit bundles. You can't bundle different multihomed Ethernet segments at the access routers into the same group. This is because the device only has one Ethernet A-D per EVI route for each multihomed Ethernet segment, and each multihomed Ethernet segment must work independently in the forwarding path.

To support interoperability with VLAN unaware multihoming FXC at the access router:

- Bundle attachment circuits on the same multihomed Ethernet segment to use the same local service instance ID on the access router.
- Assign a unique local service instance ID to each attachment circuit bundle.
- At the PE device, use a separate pseudowire subscriber transport logical interface to terminate an access router group of attachment circuits on the same multihomed Ethernet segment.
- Assign a unique local service instance ID corresponding to each pseudowire subscriber transport logical interface.

Here's an example configuration to create a group that bundles the attachment circuits that share the same multihomed ESI. You create the group at the [edit <routing-instances *name*> protocols evpn] hierarchy level for a VLAN unaware FXC service at an access router as follows:

```
set routing-instances EVPN-VPWS-FXC-AE protocols evpn group G1 esi 00:71:81:00:00:00:00:00:01;
set routing-instances EVPN-VPWS-FXC-AE protocols evpn group G1 interface ge-0/0/3.1;
set routing-instances EVPN-VPWS-FXC-AE protocols evpn group G1 interface ge-0/0/3.2;
set routing-instances EVPN-VPWS-FXC-AE protocols evpn group G1 service-id local 300;
set routing-instances EVPN-VPWS-FXC-AE protocols evpn group G1 service-id remote 300;
```

VLAN Aware FXC on Service Edge Routers

On an access router, VLAN aware (VLAN signaled) FXC shares the same MPLS label for a group of attachment circuits in the same EVI. For each attachment circuit in the same group, we:

- Assign a unique local service instance ID per attachment circuit.
- Identify the attachment circuit in the forwarding plane with:
 - One unique normalized VLAN ID for single tagged frames.
 - Multiple VLAN IDs for Q-in-Q tagged frames.

With a VLAN aware FXC service, we signal each individual point-to-point Ethernet service using a pair of Ethernet A-D routes in the control plane, one for each of its attachment circuits. This is the same as the point-to-point Ethernet service that EVPN-VPWS offers. If you use the FXC service on the access routers running in either single-active or all-active multihomed mode, we recommend you use the VLAN aware service instead of the VLAN unaware service.

The VLAN aware FXC service uses pseudowire subscriber *service* logical interfaces at the service edge router. This service associates the local service instance ID with the pseudowire subscriber service logical interface (instead of the pseudowire subscriber *transport* logical interface).

With single-active multihoming, you can configure each pseudowire subscriber service logical interface for an Ethernet segment using an ESI per logical interface.



NOTE: To support multihoming on the service edge router, you need an ESI configuration for the pseudowire subscriber service logical interface. As a result, the device load-balances the traffic among a set of service edge routers based on the VLAN.

For PE devices interoperating with VLAN aware FXC access routers:

- This mode uses only one pseudowire subscriber logical interface for each EVI. You can manually configure a pair of local and remote service instance IDs on each of the pseudowire subscriber logical interfaces. Otherwise, the device auto-derives the local service instance ID from the normalized VLAN ID(s), which would produce the same local and remote service instance ID.
- For single-active multihoming mode, configure a non-reserved ESI for the pseudowire subscriber service logical interface.
- Each attachment circuit has a separate Ethernet A-D per EVI route advertised with its Ethernet tag ID set to the local service instance ID. You can manually configure the Ethernet tag ID, or the device auto-derives it based on the normalized VLAN ID(s).

Load-Balancing FXC VLAN Aware Service Traffic on PE Devices

To load-balance FXC VLAN aware service traffic on PE devices, you must configure the single-active multihomed Ethernet Segment ID (ESI) under the PS interface with PWHT as shown in the following sample configuration:

```
Ps0 {
  anchor-point {
    lt-0/2/10;
  }
  flexible-vlan-tagging;
  unit 0 {
    encapsulation ethernet-ccc;
  }
  unit 1 {
    esi {
      00:02:03::01;
      single-active;
    }
    encapsulation vlan-vpls;
    vlan-id 200;
  }
}
```

```

        family vpls;
    }
    unit 2 {
        esi {
            00:03:03::02;
            single-active;
        }
        encapsulation vlan-vpls;
        vlan-id 201;
        family vpls;
    }
}

```

VLAN Tagging

Pseudowire subscriber logical interfaces support untagged, single-tagged and double-tagged frames. For interoperability with EVPN FXC, the pseudowire subscriber logical interface must support demultiplexing on a single VLAN ID or double VLAN tags (Q-in-Q).

Signaling EVPN FXC Optional Bits

The M-bit and V-bit are optional bits in the EVPN FXC Layer 2 extended community. Devices signal only the M-bit in the EVPN Layer 2 extended community to indicate VLAN unaware or VLAN aware (VLAN signaled) FXC.

Access Side Multihoming

The CE devices can be multihomed to the access routers running in either single-active or all-active mode. The service side routers can be single-homed, or multihomed in single-active mode only.

The next sections describe use cases where the access side routers can operate in single-active or all-active multihoming mode, while the service side routers act in single-active multihoming mode.



NOTE: We support ACX Series routers only as access side routers for FXC, including ACX5448, ACX710, and ACX7xxxx routers. MX Series routers can be access side or service side routers for FXC.

We support access side ACX Series and MX Series routers only in the following configurations:

- As access side routers single-homed to a service side PE router
- As access side routers multihomed to a service side PE router where the service side PE router is in single-homing or single-active multihoming mode

Access Side Single-active with Service Side Single-active

Figure 161 on page 1620 shows a typical square topology with:

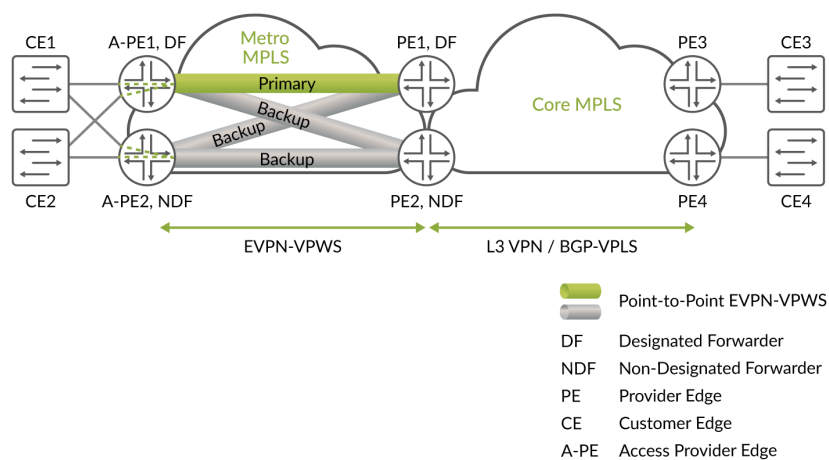
- Two service edge routers, PE1 and PE2.
- Two access side PE routers, A-PE1 and A-PE2.



NOTE: We call the access routers A-PE devices in all of the figures in this document to distinguish them from the service edge routers.

A-PE1 and A-PE2 work in single-active mode. One of the service edge routers is elected as DF. One of the access routers is elected as DF. The DF access router and the DF service edge router have only one active or primary pseudowire between them. If one of the DF PE devices has an access link down or suffers node failure, the NDF PE router becomes the DF. As a result, if the existing primary pseudowire goes down, a new primary pseudowire is established among the DF PE devices. This happens per pseudowire subscriber service logical interface (not per PE device).

Figure 161: EVPN-VPWS Pseudowire Subscriber Interface Single Active



8200363

Access Side All-active with Service Side Single-active

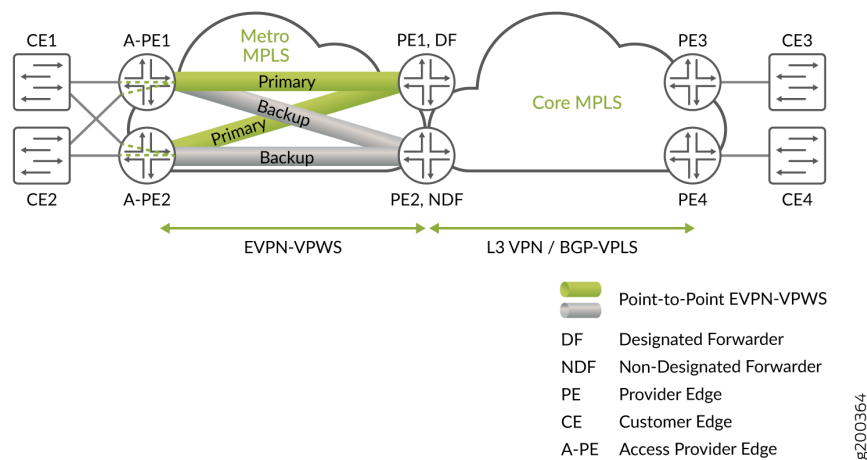
Figure 162 on page 1621 shows a square topology with:

- Two service edger routers, PE1 and PE2.
- Two A-PE routers, A-PE1 and A-PE2.

A-PE1 and A-PE2 are multihoming peer PE devices in all-active mode for CE devices CE-1 and CE2 on their right side. Both PE devices A-PE1 and A-PE2 can forward the traffic for EVPN-VPWS in this mode.

PE1 and PE2 are multihoming peer PE devices in single-active mode for the service. Only one of the PE devices PE1 and PE2 can forward traffic for EVPN-VPWS in this mode. In this figure, PE1 is the DF, so only PE1 forwards the traffic. As a result, this setup establishes two primary pseudowires, one from PE1 to A-PE1 and one from PE1 to A-PE2. To reach CE1 or CE2, PE1 sends the traffic by way of either A-PE1 or A-PE2. PE1 balances the traffic load between those two A-PE devices.

Figure 162: EVPN-VPWS Pseudowire Subscriber Interface All Active



Overview of Headend Termination for EVPN VPWS for Business Services

IN THIS SECTION

- Benefits of Pseudowire Headend Termination (PWHT) with EVPN-VPWS Pseudowires: | 1622
- Pseudowire Subscriber Logical Interface Support for EVPN-VPWS | 1622

Currently, Junos OS only supports pseudowire subscriber logical interface to be used with Layer 2 circuit or Layer 2 VPN for pseudowire headend termination. An Ethernet VPN (EVPN) enables you to connect dispersed customer sites using a Layer 2 virtual bridge. 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. EVPN-VPWS as a next generation of pseudowire technology brings the benefit of EVPN to point-to-point service by providing fast convergence upon node failure and link failure through its multi-homing feature. As a result, you can use EVPN-VPWS and pseudowire subscriber interface for headend termination into different services.

The pseudowire headend termination support with pseudowire subscriber interface works with EVPN-VPWS Flexible Cross Connect (FXC) in the vMX (Virtual MX) scale out solution. The vMX scale out support with EVPN-VPWS FXC is terminated into Layer 3 VRF or BGP-VPLS through pseudowire subscriber interface on vMX



NOTE: The multi-home scenario includes support for either two or more than two PEs.

Benefits of Pseudowire Headend Termination (PWHT) with EVPN-VPWS

Pseudowires:

- PWHT framework can be used to attach remote CEs (reachable via metro aggregation network) to the services (for example Layer 3 VPN) available on service PE in a similar way to CEs which are directly attached to service PEs.
- Multiple VLANs can be transported inside pseudowire and demultiplexed to different services on service PE using pseudowire subscriber logical interfaces just like VLANs on regular physical interfaces.
- PWHT with EVPN-VPWS as transport pseudowire brings further numerous benefits; for example unified BGP based control plane for all services in metro aggregation and core network, or redundant active-standby transport connectivity between multiple service PEs and multiple aggregation nodes.

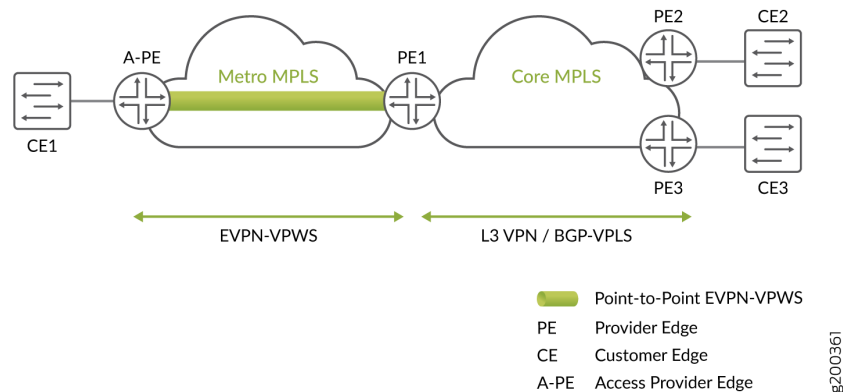
Pseudowire Subscriber Logical Interface Support for EVPN-VPWS

Figure 163 on page 1623 illustrates an EVPN-VPWS network with pseudowire service. A-PE (Access Provider Edge) is the access router and PE1 is the service edge router. A pseudowire subscriber logical interface is used on PE1 to terminate the pseudowire established by EVPN-VPWS into either Layer 3

VPN or Layer 2 BGP-VPLS service. There is only one single service edge router to terminate the pseudowire, in this scenario.

For a given pseudowire subscriber logical interface, for example ps0, only the transport interface of the pseudowire subscriber logical interface, that is ps0.0, is used as an access interface for the EVPN-VPWS on PE1. The service interfaces of ps0, that is ps0.1 to ps0.n, are used at the service side (either under Layer 3 VRF or BGP-VPLS instance).

Figure 163: EVPN-VPWS with Pseudowire Subscriber Interface



NOTE: Prior to Junos OS 18.2 Release, the only encapsulation type that pseudowire subscriber transport logical interface (as an access interface for the pseudowire) supports for Layer 2 circuit and Layer 2 VPN is ethernet-ccc. This encapsulation type will remain the same for EVPN-VPWS with pseudowire subscriber logical interface support as well.

Single Pseudowire Headend Termination

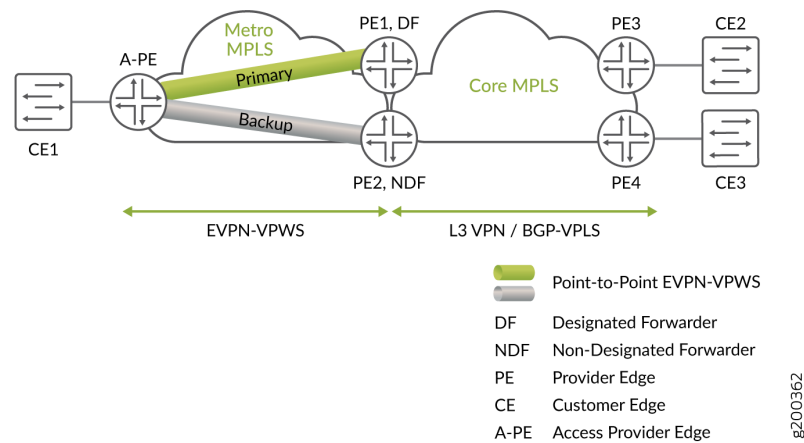
The pseudowire subscriber transport logical interface is used as the access interface for EVPN-VPWS instance. EVPN establishes a point-to-point Ethernet service with pseudowire subscriber transport logical interface as an access interface in the control plane.

Active-Standby Pseudowire Headend Termination

As shown in [Figure 164 on page 1624](#), to achieve resilience for pseudowire headend termination into Layer 3 VPN or BGP-VPLS, you can use a pair of redundant pseudowires on the EVPN-VPWS side. Also, in [Figure 164 on page 1624](#), Layer 3 VPN or BGP-VPLS is treated as if it is multi-homed to the set of redundant service edge routers, PE1 and PE2. PE1 and PE2 work in active-standby mode for EVPN-

VPWS. One of them is elected as the primary PE per EVPN normal designated forwarder (DF) election procedure. Only the primary PE is used to forward Layer 2 traffic.

Figure 164: EVPN-VPWS Active/standby with Pseudowire Subscriber Interface



Following are the reasons for the pseudowire failure, when redundancy is provided by the EVPN-VPWS active-standby interface:

- Service edge router node failure.
- Pseudowire subscriber transport logical interface failure on the primary PE.
- Failure on the path towards the primary PE.



NOTE: If any of the above failure cases are detected, the backup PE takes over to become the primary PE. As a result, the customer traffic from the access router, A-PE, is switched to the new primary PE.

The EVPN-VPWS also supports the service side of the network multi-homing to EVPN-VPWS through the pseudowire subscriber logical interface in an active-standby mode.

To achieve active-standby pseudowire headend termination, the following is required:

- *Ethernet segment identifier (ESI)* support on the pseudowire subscriber transport logical interface.
- Synchronizing data path between active pseudowire and Layer 3 VPN.
- MAC flush in the BGP-VPLS when the active-standby pseudowires switch. MAC flush is triggered when the active service edge node suffers a node failure.

ES Support on Pseudowire Subscriber Transport Logical Interface

An Ethernet segment must have a unique nonzero identifier, called the *Ethernet segment identifier (ESI)*. The ESI is encoded as a 10-octet integer. When manually configuring an ESI value, the most significant octet, known as the *type byte*, must be 00. ESI is assigned to pseudowire subscriber transport logical interfaces associated with the PE1 and PE2 the same way as in EVPN multi-homing.

To configure the active-standby mode for the pseudowire subscriber transport logical interface, include the ESI value and the `single-active` statement under the `[edit interfaces]` hierarchy level.

DF Election and Pseudowire Subscriber Transport Logical Interface

Designated forwarder(DF)—This is the designated forwarder for forwarding the current traffic. The DF election procedure ensures each VLAN is associated with single PE acting in DF role.

Synchronizing Data Path between EVPN-VPWS and Layer 3 VPN or BGP-VPLS

There is a need for the data path coordination between the point-to-point pseudowire and the services side of the network (Layer 3 VPN or BGP-VPLS), with active-standby pseudowire headend termination. This is to select the same service edge router for passing the current traffic between the point-to-point pseudowire and Layer 3 VPN or BGP-VPLS domain.

EVPN-VPWS determines the primary path, for both Layer 3 VPN and BGP-VPLS, based on its DF election result. Both the primary and backup PEs for EVPN-VPWS then influence the services side of network to use the primary PE for data traffic.

Layer 3 VPN

When pseudowires are headend terminated into the Layer 3 VPN in an active-standby mode, the primary and the backup PEs use routing-policy and policy-condition to increase or decrease the Local Preference (LP). As a result, the BGP path selection algorithm is carried out on the Layer 3 side and picks up the primary PE.

BGP-VPLS

EVPN-VPWS establishes active-standby pseudowires based on its DF election. The active-standby pseudowires are essentially the two redundant spoke pseudowires attached to BGP-VPLS instance. The BGP-VPLS PEs remain single homed PEs and data plane learning happens through the active pseudowire in the EVPN-VPWS primary path.



NOTE:

- If the pseudowire subscriber service logical interface that belongs to a BGP-VPLS instance is in down state, this does not trigger the active and standby pseudowire switch on the EVPN-VPWS side. This is because the 1 to many relationships between the point-to-point pseudowire and the BGP-VPLS. This may cause traffic loss for traffic coming from the VPLS to the point-to-point pseudowire direction.
- BGP-VPLS does not trigger MAC flush when its access link to the CE device goes up or down. Further, since multi-homed EVPN-VPWS pseudowire headend termination into BGP-VPLS is based on single-homed mode on BGP-VPLS side, MAC flush (achieved via manipulation of F-bit in BGP-VPLS messages) associated with multi-homed BGP-VPLS mode does not apply here, neither. Therefore, when there is DF or NDF state change on EVPN-VPWS side, or if the pseudowire subscriber service logical interface that belongs to a BGP-VPLS instance goes up or down, MAC flush is not triggered. Null-route filtering might occur until expiration of MAC aging timer, unless there is constant traffic from the CE behind the EVPN-VPWS to BGP-VPLS. MAC flush is triggered only during PE node failures.

The following configuration example shows how condition manager is used for Layer 3 VPN routing instance through vrf-export when active-standby EVPN-VPWS terminates into Layer 3 VPN service.

```
[edit]
routing-instances {
  l3vpn_1 {
    instance-type vrf;
    interface ps0.1;
    interface lo0.1;
    route-distinguisher 2.2.2.3:6500;
    vrf-import l3vpn_1_import;
    vrf-export l3vpn_1_export;
    vrf-table-label;
    protocols {
      bgp {
        group toPW-CE1 {
          type external;
          export send-direct;
          peer-as 1;
          neighbor 10.1.1.1;
        }
      }
    }
  }
}
```

```

    }
}
policy-options {
    policy-statement l3vpn_1_export {
        term 1 {
            from condition primary;
            then {
                local-preference add 300;
                community set l3vpn_1;
                accept;
            }
        }
        term 2 {
            from condition standby;
            then {
                community set l3vpn_1;
                accept;
            }
        }
    }
}
policy-statement l3vpn_1_import {
    term 1 {
        from community l3vpn_1;
        then accept;
    }
    term default {
        then reject;
    }
}
community l3vpn_1 members target:65056:100;
condition primary {
    if-route-exists {
        address-family {
            ccc {
                ps0.0;
                table mpls.0;
            }
        }
    }
}
condition standby {
    if-route-exists {
        address-family {

```

```

        ccc {
            ps0.0;
            table mpls.0;
            standby;
        }
    }
}
}
}

```

Following is the BGP-VPLS instance configuration when EVPN-VPWS terminates into the BGP-VPLS:

```

[edit]
routing-instances {
    vpls-1 {
        instance-type vpls;
        interface ps0.1;
        route-distinguisher 100:2;
        vrf-target target:100:100;
        protocols {
            vpls {
                site ce3 {
                    site-identifier 3;
                }
            }
        }
    }
}
}

```

Active-Active Pseudowire Headend Termination

Active-active pseudowire headend termination is not supported both in Layer 3 VPN and in BGP-VPLS.

vMX Scale Out

The vMX router is a virtual version of the MX Series 5G Universal Routing Platform. Like the MX Series router, the vMX router runs the Junos operating system (Junos OS) and supports Junos OS packet handling and forwarding modeled after the Trio chipset. The vMX instance contains two separate virtual

machines (VMs), one for the virtual forwarding plane (VFP) and one for the virtual control plane (VCP). The VFP VM runs the virtual Trio forwarding plane software and the VCP VM runs Junos OS.

The vMX can now be used to scale out the bandwidth, achieve service isolation, and resilience. As the service edge router, vMX supports pseudowire headend termination with active and standby mode.

vMX

A vMX can have active and backup VCP with one or more VFPs. The VCP is the RE and the VFP is the line-card. The communication between VCP and VFP is facilitated through vRouter, if the VCP and VFP reside on the same server. Else, it is through the external network to which the servers are connected. There is no direction communication between VFPs.

Service Isolation

For a given point-to-point pseudowire with only one VFP per vMX, the data traffic is handled by one VFP for both ingress and egress traffic.

Active-Standby Pseudowire Headend Termination into VMX

The vMX acts as a regular MX for the overlay point-to-point Ethernet service. Similar to physical MX router, redundancy is achieved by using multiple vMXs, and only active-standby pseudowire headend termination is supported. The pseudowire is terminated at the loopback IP address that is used as the protocol nexthop address for the pseudowire.

RELATED DOCUMENTATION

[Example: Configuring VPWS with EVPN Signaling Mechanisms](#) | 1632

Configuring VPWS with EVPN Signaling Mechanisms

The virtual private wire service (VPWS) with Ethernet VPN (EVPN) provides single-active or all-active multihoming capabilities along with support for Inter-AS options associated with BGP-signaled VPNs. The VPWS service identifiers identify the endpoints of the EVPN-VPWS network. Each provider edge (PE) router in EVPN-VPWS network is configured with local and remote VPWS service identifiers.

Before you configure VPWS service with EVPN mechanisms, you must do the following:

1. Configure the router interfaces.
2. Configure the router ID and autonomous system number for the device.

3. Configure OSPF.
4. Configure a BGP internal group.
5. Configure ISIS.
6. Include the EVPN signaling network layer reachability information (NLRI) to the internal BGP group.
7. Configure LDP.
8. Configure MPLS.
9. Configure MPLS LSP or GRE tunnels.
10. Configure EVPN all-active multihoming and EVPN single-active multihoming.

To configure a PE device with a VPWS service identifier in an EVPN-VPWS network, do the following:

1. Configure the EVPN-VPWS routing instance.

```
[edit routing-instances]
user@PE# set evpn-vpws-instance instance-type instance-type-value
```

For example, configure routing instance vpws1004 with instance type evpn-vpws.

```
[edit routing-instances]
user@PE# set vpws1004 instance-type evpn-vpws
```

2. Configure the interface names for the EVPN-VPWS routing instance.

```
[edit routing-instances]
user@PE# set evpn-vpws-instance interface interface-name
```

For example, configure interface ge-0/0/1.1004 for the EVPN-VPWS instance vpws1004.

```
[edit routing-instances]
user@PE# set vpws1004 interface ge-0/0/1.1004
```

3. Configure the route distinguisher for the EVPN-VPWS routing instance.

```
[edit routing-instances]
user@PE# set evpn-vpws-instance route-distinguisher route-distinguisher-value
```

For example, configure route distinguisher 10.255.0.1:100 for EVPN-VPWS routing instance vpws1004.

```
[edit routing-instances]
user@PE# set vpws1004 route-distinguisher 10.255.0.1:100
```

4. Configure the VPN routing and forwarding (VRF) target community for the EVPN-VPWS routing instance.

```
[edit routing-instances]
user@PE# set evpn-vpws-instance vrf-target vrf-target
```

For example, configure VRF target target:100:1004 for EVPN-VPWS routing instance vpws1004.

```
[edit routing-instances]
user@PE# set vpws1004 vrf-target target:100:1004
```

5. Configure the interface of a routing instance with local and remote service identifiers. These identifiers identify the PE routers that forward and receive the traffic in the EVPN-VPWS network. The local service identifier is used to identify the PE router that is forwarding the traffic, and the remote service identifier is used to identify the PE router that is receiving the traffic in the network.

```
[edit routing-instances evpn-vpws-instance protocols evpn interface interface-name]
user@PE# set vpws-service-id local local-service-id
user@PE# set vpws-service-id remote remote-service-id
```

For example, configure the interface ge-0/0/1.1004 with the local and remote service identifiers 1004 and 2004 for EVPN-VPWS routing instance vpws1004.

```
[edit routing-instances vpws1004 protocols evpn interface ge-0/0/1.1004]
user@PE# set vpws-service-id local 1004
user@PE# set vpws-service-id remote 2004
```

RELATED DOCUMENTATION

[Overview of VPWS with EVPN Signaling Mechanisms](#) | 1606

Example: Configuring VPWS with EVPN Signaling Mechanisms

IN THIS SECTION

- [Requirements | 1632](#)
- [Overview and Topology | 1632](#)
- [Configuration | 1634](#)
- [Verification | 1646](#)

This example shows how to implement Virtual Private Wire Service (VPWS) with Ethernet Virtual Private Network (EVPN) signaling. The use of EVPN signaling provides single-active or all-active multihoming capabilities for BGP-signaled VPNs.

Requirements

This example uses the following hardware and software components:

- Four MX Series routers acting as provider edge (PE) devices, running Junos OS Release 17.1 or later
- Two customer edge (CE) devices (MX Series routers are used in this example)

Overview and Topology

IN THIS SECTION

- [Topology | 1633](#)

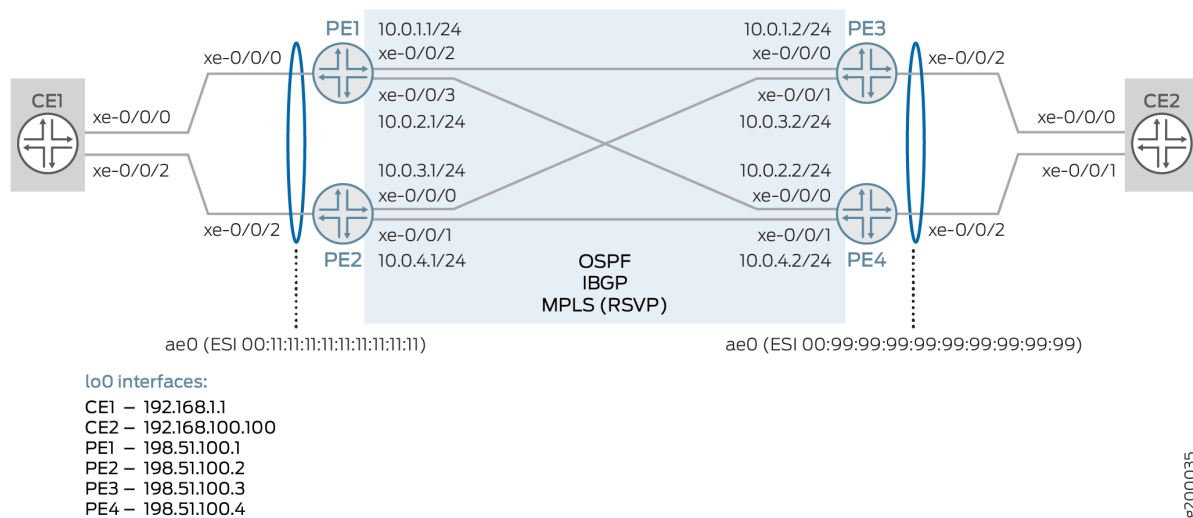
VPWS employs Layer 2 VPN services over MPLS to build a topology of point-to-point connections that connect end customer sites. EVPN enables you to connect dispersed customer sites using a Layer 2 virtual bridge. These two elements can be combined to provide an EVPN-signaled VPWS.

The `vpws-service-id` statement identifies the endpoints of the EVPN-VPWS based on local and remote service identifiers configured on the PE routers in the network. These endpoints are autodiscovered using BGP-based EVPN signaling to exchange the service identifier labels.

Topology

This example uses the topology shown in [Figure 165 on page 1633](#), consisting of four PE routers and two CE routers. Router CE1 is multihomed to Routers PE1 and PE2; Router CE2 is multihomed to Routers PE3 and PE4.

Figure 165: VPWS with EVPN Signaling



The following configuration elements are used in this scenario:

- CE devices:
 - Aggregated Ethernet (AE) interface towards related PE devices
- PE devices:
 - AE interface, with EVPN segment identifier (ESI), towards related CE device
 - OSPF and IBGP in the core
 - MPLS LSPs using RSVP in the core
 - Per-packet load balancing
 - Routing instance using instance type `evpn-vpws`, and the `vpws-service-id` statement to define the local and remote endpoints.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1634](#)
- [Procedure | 1639](#)
- [Results | 1643](#)

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.

CE1

```
set interfaces xe-0/0/0 gigether-options 802.3ad ae0
set interfaces xe-0/0/2 gigether-options 802.3ad ae0
set chassis aggregated-devices ethernet device-count 1
set interfaces ae0 description "to PE1/2"
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 unit 100 encapsulation vlan-bridge
set interfaces ae0 unit 100 vlan-id 1000
set interfaces lo0 unit 0 family inet address 192.168.1.1/32
set policy-options policy-statement LB then load-balance per-packet
set routing-options forwarding-table export LB
set bridge-domains bd100 domain-type bridge
set bridge-domains bd100 vlan-id 1000
set bridge-domains bd100 interface ae0.100
```

CE2

```
set interfaces xe-0/0/0 gigether-options 802.3ad ae0
set interfaces xe-0/0/1 gigether-options 802.3ad ae0
set chassis aggregated-devices ethernet device-count 1
```

```

set interfaces ae0 description "to PE3/4"
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 unit 100 encapsulation vlan-bridge
set interfaces ae0 unit 100 vlan-id 1000
set interfaces lo0 unit 0 family inet address 192.168.100.100/32
set policy-options policy-statement LB then load-balance per-packet
set routing-options forwarding-table export LB
set bridge-domains bd100 domain-type bridge
set bridge-domains bd100 vlan-id 1000
set bridge-domains bd100 interface ae0.100

```

PE1

```

set interfaces xe-0/0/0 description "to CE1"
set interfaces xe-0/0/0 gigether-options 802.3ad ae0
set interfaces xe-0/0/2 unit 0 description "to PE3"
set interfaces xe-0/0/2 unit 0 family inet address 10.0.1.1/24
set interfaces xe-0/0/2 unit 0 family mpls
set interfaces xe-0/0/3 unit 0 description "to PE4"
set interfaces xe-0/0/3 unit 0 family inet address 10.0.2.1/24
set interfaces xe-0/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 198.51.100.1/32
set chassis aggregated-devices ethernet device-count 1
set interfaces ae0 description "to CE1"
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:01
set interfaces ae0 unit 100 encapsulation vlan-ccc
set interfaces ae0 unit 100 vlan-id 1000
set protocols ospf area 0.0.0.0 interface xe-0/0/2.0
set protocols ospf area 0.0.0.0 interface xe-0/0/3.0
set protocols ospf area 0.0.0.0 interface lo0.0
set routing-options autonomous-system 65000
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 198.51.100.1
set protocols bgp group IBGP family evpn signaling
set protocols bgp group IBGP neighbor 198.51.100.2

```

```

set protocols bgp group IBGP neighbor 198.51.100.3
set protocols bgp group IBGP neighbor 198.51.100.4
set protocols rsvp interface xe-0/0/2.0
set protocols rsvp interface xe-0/0/3.0
set protocols mpls interface xe-0/0/2.0
set protocols mpls interface xe-0/0/3.0
set protocols mpls no-cspf
set protocols mpls label-switched-path PE1toPE3 to 198.51.100.3
set protocols mpls label-switched-path PE1toPE4 to 198.51.100.4
set policy-options policy-statement LB then load-balance per-packet
set routing-options forwarding-table export LB
set routing-instances EVPN-VPWS instance-type evpn-vpws
set routing-instances EVPN-VPWS interface ae0.100
set routing-instances EVPN-VPWS route-distinguisher 198.51.100.1:11
set routing-instances EVPN-VPWS vrf-target target:100:11
set routing-instances EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id local 1111
set routing-instances EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id remote 9999

```

PE2

```

set interfaces xe-0/0/0 unit 0 description "to PE3"
set interfaces xe-0/0/0 unit 0 family inet address 10.0.3.1/24
set interfaces xe-0/0/0 unit 0 family mpls
set interfaces xe-0/0/1 unit 0 description "to PE4"
set interfaces xe-0/0/1 unit 0 family inet address 10.0.4.1/24
set interfaces xe-0/0/1 unit 0 family mpls
set interfaces xe-0/0/2 description "to CE1"
set interfaces xe-0/0/2 gigether-options 802.3ad ae0
set interfaces lo0 unit 0 family inet address 198.51.100.2/32
set chassis aggregated-devices ethernet device-count 1
set interfaces ae0 description "to CE1"
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:01
set interfaces ae0 unit 100 encapsulation vlan-ccc
set interfaces ae0 unit 100 vlan-id 1000
set protocols ospf area 0.0.0.0 interface xe-0/0/0.0
set protocols ospf area 0.0.0.0 interface xe-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0

```

```

set routing-options autonomous-system 65000
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 198.51.100.2
set protocols bgp group IBGP family evpn signaling
set protocols bgp group IBGP neighbor 198.51.100.1
set protocols bgp group IBGP neighbor 198.51.100.3
set protocols bgp group IBGP neighbor 198.51.100.4
set protocols rsvp interface xe-0/0/0.0
set protocols rsvp interface xe-0/0/1.0
set protocols mpls interface xe-0/0/0.0
set protocols mpls interface xe-0/0/1.0
set protocols mpls no-cspf
set protocols mpls label-switched-path PE2toPE3 to 198.51.100.3
set protocols mpls label-switched-path PE2toPE4 to 198.51.100.4
set policy-options policy-statement LB then load-balance per-packet
set routing-options forwarding-table export LB
set routing-instances EVPN-VPWS instance-type evpn-vpws
set routing-instances EVPN-VPWS interface ae0.100
set routing-instances EVPN-VPWS route-distinguisher 198.51.100.2:11
set routing-instances EVPN-VPWS vrf-target target:100:11
set routing-instances EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id local 1111
set routing-instances EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id remote 9999

```

PE3

```

set interfaces xe-0/0/0 unit 0 description "to PE1"
set interfaces xe-0/0/0 unit 0 family inet address 10.0.1.2/24
set interfaces xe-0/0/0 unit 0 family mpls
set interfaces xe-0/0/1 unit 0 description "to PE2"
set interfaces xe-0/0/1 unit 0 family inet address 10.0.3.2/24
set interfaces xe-0/0/1 unit 0 family mpls
set interfaces xe-0/0/2 description "to CE1"
set interfaces xe-0/0/2 gigether-options 802.3ad ae0
set interfaces lo0 unit 0 family inet address 198.51.100.3/32
set chassis aggregated-devices ethernet device-count 1
set interfaces ae0 description "to CE2"
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:99:99:99:99:99:99:99:99
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:01

```

```

set interfaces ae0 unit 100 encapsulation vlan-ccc
set interfaces ae0 unit 100 vlan-id 1000
set protocols ospf area 0.0.0.0 interface xe-0/0/0.0
set protocols ospf area 0.0.0.0 interface xe-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0
set routing-options autonomous-system 65000
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 198.51.100.3
set protocols bgp group IBGP family evpn signaling
set protocols bgp group IBGP neighbor 198.51.100.1
set protocols bgp group IBGP neighbor 198.51.100.2
set protocols bgp group IBGP neighbor 198.51.100.4
set protocols rsvp interface xe-0/0/0.0
set protocols rsvp interface xe-0/0/1.0
set protocols mpls interface xe-0/0/0.0
set protocols mpls interface xe-0/0/1.0
set protocols mpls no-cspf
set protocols mpls label-switched-path PE3toPE1 to 198.51.100.1
set protocols mpls label-switched-path PE3toPE2 to 198.51.100.2
set policy-options policy-statement LB then load-balance per-packet
set routing-options forwarding-table export LB
set routing-instances EVPN-VPWS instance-type evpn-vpws
set routing-instances EVPN-VPWS interface ae0.100
set routing-instances EVPN-VPWS route-distinguisher 198.51.100.3:11
set routing-instances EVPN-VPWS vrf-target target:100:11
set routing-instances EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id local 9999
set routing-instances EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id remote 1111

```

PE4

```

set interfaces xe-0/0/0 unit 0 description "to PE1"
set interfaces xe-0/0/0 unit 0 family inet address 10.0.2.2/24
set interfaces xe-0/0/0 unit 0 family mpls
set interfaces xe-0/0/1 unit 0 description "to PE2"
set interfaces xe-0/0/1 unit 0 family inet address 10.0.4.2/24
set interfaces xe-0/0/1 unit 0 family mpls
set interfaces xe-0/0/2 description "to CE1"
set interfaces xe-0/0/2 gigether-options 802.3ad ae0
set interfaces lo0 unit 0 family inet address 198.51.100.4/32
set chassis aggregated-devices ethernet device-count 1
set interfaces ae0 description "to CE2"
set interfaces ae0 flexible-vlan-tagging

```

```

set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:99:99:99:99:99:99:99
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:01
set interfaces ae0 unit 100 encapsulation vlan-ccc
set interfaces ae0 unit 100 vlan-id 1000
set protocols ospf area 0.0.0.0 interface xe-0/0/0.0
set protocols ospf area 0.0.0.0 interface xe-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0
set routing-options autonomous-system 65000
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 198.51.100.4
set protocols bgp group IBGP family evpn signaling
set protocols bgp group IBGP neighbor 198.51.100.1
set protocols bgp group IBGP neighbor 198.51.100.2
set protocols bgp group IBGP neighbor 198.51.100.3
set protocols rsvp interface xe-0/0/0.0
set protocols rsvp interface xe-0/0/1.0
set protocols mpls interface xe-0/0/0.0
set protocols mpls interface xe-0/0/1.0
set protocols mpls no-cspf
set protocols mpls label-switched-path PE4toPE1 to 198.51.100.1
set protocols mpls label-switched-path PE4toPE2 to 198.51.100.2
set policy-options policy-statement LB then load-balance per-packet
set routing-options forwarding-table export LB
set routing-instances EVPN-VPWS instance-type evpn-vpws
set routing-instances EVPN-VPWS interface ae0.100
set routing-instances EVPN-VPWS route-distinguisher 198.51.100.4:11
set routing-instances EVPN-VPWS vrf-target target:100:11
set routing-instances EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id local 9999
set routing-instances EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id remote 1111

```

Procedure

Step-by-Step Procedure

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



NOTE: Only Router PE1 is shown here. Repeat this procedure for all other PE devices, using the appropriate interface names, addresses, and other parameters for each device. The step-by-step procedure for CE devices is not shown.

To configure Router PE1:

1. Configure the CE-facing interface to be part of the ae0 bundle.

The second interface for the AE bundle will be configured on the other local PE device.

```
[edit interfaces]
user@PE1# set xe-0/0/0 description "to CE1"
user@PE1# set xe-0/0/0 gigether-options 802.3ad ae0
```

2. Configure the core-facing interfaces toward Routers PE3 and PE4.

Be sure to include the MPLS protocol family.

```
[edit interfaces]
user@PE1# set xe-0/0/2 unit 0 description "to PE3"
user@PE1# set xe-0/0/2 unit 0 family inet address 10.0.1.1/24
user@PE1# set xe-0/0/2 unit 0 family mpls
user@PE1# set xe-0/0/3 unit 0 description "to PE4"
user@PE1# set xe-0/0/3 unit 0 family inet address 10.0.2.1/24
user@PE1# set xe-0/0/3 unit 0 family mpls
```

3. Configure the loopback interface.

```
[edit interfaces]
user@PE1# set lo0 unit 0 family inet address 198.51.100.1/32
```

4. Define the number of aggregated Ethernet interfaces to be supported on the device.

```
[edit chassis]
user@PE1# set aggregated-devices ethernet device-count 1
```

5. Configure the ae0 interface.

Alternate VLAN tagging and encapsulation options can be used, depending on your needs.

```
[edit interfaces]
user@PE1# set ae0 description "to CE1"
user@PE1# set ae0 flexible-vlan-tagging
user@PE1# set ae0 encapsulation flexible-ethernet-services
user@PE1# set ae0 unit 100 encapsulation vlan-ccc
user@PE1# set ae0 unit 100 vlan-id 1000
```

6. Assign an Ethernet segment identifier (ESI) value to the ae0 interface and enable EVPN active-active multihoming.

```
[edit interfaces]
user@PE1# set ae0 esi 00:11:11:11:11:11:11:11:11
user@PE1# set ae0 esi all-active
```

7. Configure link aggregation control protocol (LACP) for the ae0 interface.

The system ID used here must be the same on both local PE devices.

```
[edit interfaces]
user@PE1# set ae0 aggregated-ether-options lacp active
user@PE1# set ae0 aggregated-ether-options lacp system-id 00:00:00:00:00:01
```

8. Enable OSPF on the core-facing (and loopback) interfaces.

```
[edit protocols]
user@PE1# set ospf area 0.0.0.0 interface xe-0/0/2.0
user@PE1# set ospf area 0.0.0.0 interface xe-0/0/3.0
user@PE1# set ospf area 0.0.0.0 interface lo0.0
```

9. Configure an IBGP mesh with the other PE devices, using EVPN for signaling.

```
[edit routing-options]
user@PE1# set autonomous-system 65000
[edit protocols]
user@PE1# set bgp group IBGP type internal
user@PE1# set bgp group IBGP local-address 198.51.100.1
```

```

user@PE1# set bgp group IBGP family evpn signaling
user@PE1# set bgp group IBGP neighbor 198.51.100.2
user@PE1# set bgp group IBGP neighbor 198.51.100.3
user@PE1# set bgp group IBGP neighbor 198.51.100.4

```

10. Enable RSVP on the core-facing interfaces.

```

[edit protocols]
user@PE1# set rsvp interface xe-0/0/2.0
user@PE1# set rsvp interface xe-0/0/3.0

```

11. Enable MPLS on the core-facing interfaces, and configure LSPs to the remote PE devices.

For this example, be sure to disable CSPF.

```

[edit protocols]
user@PE1# set mpls interface xe-0/0/2.0
user@PE1# set mpls interface xe-0/0/3.0
user@PE1# set mpls no-cspf
user@PE1# set mpls label-switched-path PE1toPE3 to 198.51.100.3
user@PE1# set mpls label-switched-path PE1toPE4 to 198.51.100.4

```

12. Configure load balancing.

```

[edit policy-options]
user@PE1# set policy-statement LB then load-balance per-packet
[edit routing-options]
user@PE1# set forwarding-table export LB

```

13. Configure a routing instance using the `evpn-vpws` instance type. Add the AE (CE-facing) interface configured earlier, as well as a route distinguisher and VRF target.

In EVPN terms, this is an EVPN instance (EVI).

```

[edit routing-instances]
user@PE1# set EVPN-VPWS instance-type evpn-vpws
user@PE1# set EVPN-VPWS interface ae0.100
user@PE1# set EVPN-VPWS route-distinguisher 198.51.100.1:11
user@PE1# set EVPN-VPWS vrf-target target:100:11

```

14. In the routing instance, enable EVPN and add the AE interface. Then associate local and remote VPWS identifiers to the interface.

```
[edit routing-instances]
user@PE1# set EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id local 1111
user@PE1# set EVPN-VPWS protocols evpn interface ae0.100 vpws-service-id remote 9999
```

Results

From configuration mode, confirm your configuration. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit ]
user@PE1# show chassis
aggregated-devices {
  ethernet {
    device-count 1;
  }
}
```

```
[edit ]
user@PE1# show interfaces
xe-0/0/0 {
  description "to CE1";
  gigether-options {
    802.3ad ae0;
  }
}
xe-0/0/2 {
  unit 0 {
    description "to PE3";
    family inet {
      address 10.0.1.1/24;
    }
    family mpls;
  }
}
xe-0/0/3 {
  unit 0 {
```

```

        description "to PE4";
        family inet {
            address 10.0.2.1/24;
        }
        family mpls;
    }
}
ae0 {
    description "to CE1";
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
        00:11:11:11:11:11:11:11:11;
        all-active;
    }
    aggregated-ether-options {
        lacp {
            active;
            system-id 00:00:00:00:00:01;
        }
    }
    unit 100 {
        encapsulation vlan-ccc;
        vlan-id 1000;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 198.51.100.1/32;
        }
    }
}
}

```

```

[edit ]
user@PE1# show routing-options
autonomous-system 65000;
forwarding-table {

```

```
export LB;  
}
```

```
user@PE1# show protocols  
rsvp {  
    interface xe-0/0/2.0;  
    interface xe-0/0/3.0;  
}  
mpls {  
    no-cspf;  
    label-switched-path PE1toPE3 {  
        to 198.51.100.3;  
    }  
    label-switched-path PE1toPE4 {  
        to 198.51.100.4;  
    }  
    interface xe-0/0/2.0;  
    interface xe-0/0/3.0;  
}  
bgp {  
    group IBGP {  
        type internal;  
        local-address 198.51.100.1;  
        family evpn {  
            signaling;  
        }  
        neighbor 198.51.100.2;  
        neighbor 198.51.100.3;  
        neighbor 198.51.100.4;  
    }  
}  
ospf {  
    area 0.0.0.0 {  
        interface xe-0/0/2.0;  
        interface xe-0/0/3.0;  
        interface lo0.0;
```

```

    }
}

```

```

[edit ]
user@PE1# show policy-options
policy-statement LB {
    then {
        load-balance per-packet;
    }
}

```

```

[edit ]
user@PE1# show routing-instances
EVPN-VPWS {
    instance-type evpn-vpws;
    interface ae0.100;
    route-distinguisher 198.51.100.1:11;
    vrf-target target:100:11;
    protocols {
        evpn {
            interface ae0.100 {
                vpws-service-id {
                    local 1111;
                    remote 9999;
                }
            }
        }
    }
}

```

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

Verification

IN THIS SECTION

- [Verifying Aggregated Ethernet Interfaces and LACP | 1647](#)
- [Verifying OSPF | 1648](#)

- [Verifying BGP | 1649](#)
- [Verifying MPLS | 1650](#)
- [Verifying the VPWS | 1652](#)
- [Verifying Route Exchange and ESI Autodiscovery | 1653](#)
- [Verifying Local EVPN Table Route Information | 1655](#)

Confirm that the configuration is working properly.

Verifying Aggregated Ethernet Interfaces and LACP

Purpose

Verify that the AE interfaces are up and properly.

Action

Verify that the AE interfaces are up, and LACP connections are established between the PE devices and their related CE device..

```
user@CE1> show lacp interfaces extensive
```

```
Aggregated interface: ae0
```

LACP state:	Role	Exp	Def	Dist	Col	Syn	Aggr	Timeout	Activity
xe-0/0/0	Actor	No	No	Yes	Yes	Yes	Yes	Fast	Active
xe-0/0/0	Partner	No	No	Yes	Yes	Yes	Yes	Fast	Active
xe-0/0/2	Actor	No	No	Yes	Yes	Yes	Yes	Fast	Active
xe-0/0/2	Partner	No	No	Yes	Yes	Yes	Yes	Fast	Active

LACP protocol:	Receive State	Transmit State	Mux State
xe-0/0/0	Current	Fast periodic	Collecting distributing
xe-0/0/2	Current	Fast periodic	Collecting distributing

LACP info:	Role	System priority	System identifier	Port priority	Port number	Port key
xe-0/0/0	Actor	127	44:f4:77:99:e3:c0	127	1	1
xe-0/0/0	Partner	127	00:00:00:00:00:01	127	1	1
xe-0/0/2	Actor	127	44:f4:77:99:e3:c0	127	2	1
xe-0/0/2	Partner	127	00:00:00:00:00:01	127	1	1


```
user@PE1> show lacp interfaces extensive
```

```
Aggregated interface: ae0
```

LACP state:	Role	Exp	Def	Dist	Col	Syn	Aggr	Timeout	Activity
xe-0/0/0	Actor	No	No	Yes	Yes	Yes	Yes	Fast	Active
xe-0/0/0	Partner	No	No	Yes	Yes	Yes	Yes	Fast	Active

LACP protocol:	Receive State	Transmit State	Mux State
xe-0/0/0	Current	Fast periodic	Collecting distributing

LACP info:	Role	System priority	System identifier	Port priority	Port number	Port key
xe-0/0/0	Actor	127	00:00:00:00:00:01	127	1	1
xe-0/0/0	Partner	127	44:f4:77:99:e3:c0	127	1	1

```
user@PE2> show lacp interfaces extensive
```

```
Aggregated interface: ae0
```

LACP state:	Role	Exp	Def	Dist	Col	Syn	Aggr	Timeout	Activity
xe-0/0/2	Actor	No	No	Yes	Yes	Yes	Yes	Fast	Active
xe-0/0/2	Partner	No	No	Yes	Yes	Yes	Yes	Fast	Active

LACP protocol:	Receive State	Transmit State	Mux State
xe-0/0/2	Current	Fast periodic	Collecting distributing

LACP info:	Role	System priority	System identifier	Port priority	Port number	Port key
xe-0/0/2	Actor	127	00:00:00:00:00:01	127	1	1
xe-0/0/2	Partner	127	44:f4:77:99:e3:c0	127	2	1

Meaning

The AE interface on each device is up, and there are active LACP connections between the CE device and its local PE devices. Note also that the system ID configured on the PE devices, 00:00:00:00:00:01 (and shown on the PE device outputs as the Actor), matches the Partner system ID value on the CE device.

Verifying OSPF

Purpose

Verify that OSPF is working properly.

Action

Verify that OSPF has adjacencies established with its remote neighbors.

```

user@PE1> show ospf neighbor
Address          Interface          State   ID                Pri  Dead
10.0.1.2 #PE3# xe-0/0/2.0    Full   198.51.100.3     128   37
10.0.2.2 #PE4# xe-0/0/3.0    Full   198.51.100.4     128   33

user@PE3> show ospf neighbor
Address          Interface          State   ID                Pri  Dead
10.0.1.1 #PE1# xe-0/0/0.0    Full   198.51.100.1     128   34
10.0.3.1 #PE2# xe-0/0/1.0    Full   198.51.100.2     128   34

```

Meaning

Adjacencies have been established with remote neighbors.

Verifying BGP

Purpose

Verify that BGP is working properly.

Action

Verify that IBGP has peerings established with its neighbors using EVPN signaling.

```

user@PE1> show bgp summary
Groups: 1 Peers: 3 Down peers: 0
Table          Tot Paths  Act Paths Suppressed  History Damp State   Pending
bgp.evpn.0
              7          4          0          0          0          0
Peer          AS      InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
198.51.100.2 #PE2#  65000    12        5        0        0        3:03 Establ
  bgp.evpn.0: 0/3/3/0
  EVPN-VPWS.evpn.0: 0/2/2/0
  __default_evpn__.evpn.0: 0/1/1/0

```

```

198.51.100.3 #PE3# 65000 11 9 0 0 3:03 Establ
  bgp.evpn.0: 2/2/2/0
  EVPN-VPWS.evpn.0: 2/2/2/0
  __default_evpn__.evpn.0: 0/0/0/0
198.51.100.4 #PE4# 65000 9 4 0 0 1:56 Establ
  bgp.evpn.0: 2/2/2/0
  EVPN-VPWS.evpn.0: 2/2/2/0
  __default_evpn__.evpn.0: 0/0/0/0

user@PE3> show bgp summary
Groups: 1 Peers: 3 Down peers: 0
Table Tot Paths Act Paths Suppressed History Damp State Pending
bgp.evpn.0
          7          4          0          0          0          0
Peer          AS    InPkt    OutPkt    OutQ    Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
198.51.100.1 #PE1# 65000 17 11 0 0 5:09 Establ
  bgp.evpn.0: 2/2/2/0
  EVPN-VPWS.evpn.0: 2/2/2/0
  __default_evpn__.evpn.0: 0/0/0/0
198.51.100.2 #PE2# 65000 17 14 0 0 5:04 Establ
  bgp.evpn.0: 2/2/2/0
  EVPN-VPWS.evpn.0: 2/2/2/0
  __default_evpn__.evpn.0: 0/0/0/0
198.51.100.4 #PE34 65000 13 8 0 0 4:02 Establ
  bgp.evpn.0: 0/3/3/0
  EVPN-VPWS.evpn.0: 0/2/2/0
  __default_evpn__.evpn.0: 0/1/1/0

```

Meaning

EVPN-signaled IBGP peerings have been established with all neighbors.

Verifying MPLS

Purpose

Verify that MPLS is working properly.

Action

Verify that MPLS LSPs are established with remote neighbors.

```

user@PE1> show mpls lsp
Ingress LSP: 2 sessions

```

To	From	State	Rt	P	ActivePath	LSPname
198.51.100.3	198.51.100.1	Up	0	*		PE1toPE3
198.51.100.4	198.51.100.1	Up	0	*		PE1toPE4

```

Total 2 displayed, Up 2, Down 0

Egress LSP: 2 sessions

```

To	From	State	Rt	Style	Labelin	Labelout	LSPname
198.51.100.1	198.51.100.4	Up	0	1 FF	3	-	PE4toPE1
198.51.100.1	198.51.100.3	Up	0	1 FF	3	-	PE3toPE1

```

Total 2 displayed, Up 2, Down 0

Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

user@PE3> show mpls lsp
Ingress LSP: 2 sessions

```

To	From	State	Rt	P	ActivePath	LSPname
198.51.100.1	198.51.100.3	Up	0	*		PE3toPE1
198.51.100.2	198.51.100.3	Up	0	*		PE3toPE2

```

Total 2 displayed, Up 2, Down 0

Egress LSP: 2 sessions

```

To	From	State	Rt	Style	Labelin	Labelout	LSPname
198.51.100.3	198.51.100.2	Up	0	1 FF	3	-	PE2toPE3
198.51.100.3	198.51.100.1	Up	0	1 FF	3	-	PE1toPE3

```

Total 2 displayed, Up 2, Down 0

Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

```

Meaning

LSPs have been established with remote neighbors.

Verifying the VPWS

Purpose

Verify that the VPWS is established.

Action

Verify that the PE devices have exchanged and learned service identifiers, and established the VPWS.

```
user@PE1> show evpn vpws-instance
```

```
Instance: EVPN-VPWS
```

```
Route Distinguisher: 198.51.100.1:11
```

```
Number of local interfaces: 1 (1 up)
```

Interface name	ESI	Mode	Role	Status
ae0.100	00:11:11:11:11:11:11:11:11	all-active	Primary	Up
Local SID: 1111 Advertised Label: 300496				
Remote SID: 9999				
PE addr	ESI	Label	Mode	Role
TS	Status			
198.51.100.3	00:99:99:99:99:99:99:99:99	300656	all-active	Primary
2017-05-12 07:30:01.863	Resolved			
198.51.100.4	00:99:99:99:99:99:99:99:99	300704	all-active	Primary
2017-05-12 07:31:23.804	Resolved			

```
Fast Convergence Information
```

```
ESI: 00:99:99:99:99:99:99:99:99 Number of PE nodes: 2
```

```
PE: 198.51.100.3 #PE3#
```

```
Advertised SID: 9999
```

```
PE: 198.51.100.4 #PE4#
```

```
Advertised SID: 9999
```

```
user@PE3> show evpn vpws-instance
```

```
Instance: EVPN-VPWS
```

```
Route Distinguisher: 198.51.100.3:11
```

```
Number of local interfaces: 1 (1 up)
```

Interface name	ESI	Mode	Role	Status
ae0.100	00:99:99:99:99:99:99:99:99	all-active	Primary	Up
Local SID: 9999 Advertised Label: 300656				

```

Remote SID: 1111
      PE addr      ESI      Label  Mode      Role
TS      Status
      198.51.100.1  00:11:11:11:11:11:11:11:11:11:11:11:11:11:11:11 300496 all-active Primary
2017-05-12 07:30:25.702 Resolved
      198.51.100.2  00:11:11:11:11:11:11:11:11:11:11:11:11:11:11:11 300560 all-active Primary
2017-05-12 07:30:25.711 Resolved

Fast Convergence Information
ESI: 00:11:11:11:11:11:11:11:11:11:11:11:11:11:11:11 Number of PE nodes: 2
PE: 198.51.100.1    #PE1#
    Advertised SID: 1111
PE: 198.51.100.2    #PE2#
    Advertised SID: 1111

```

Meaning

The PE devices on each side of the network have advertised their service identifiers, and received the identifiers from their remote neighbors. The VPWS is established.

Verifying Route Exchange and ESI Autodiscovery

Purpose

Verify that EVPN signaling is working properly.

Action

Verify that autodiscovery information is being shared across the VPWS.

```

user@PE1> show route table bgp.evpn.0

bgp.evpn.0: 7 destinations, 7 routes (4 active, 0 holddown, 3 hidden)
+ = Active Route, - = Last Active, * = Both

1:198.51.100.3:0::9999999999999999::FFFF:FFFF/304 AD/ESI
    *[BGP/170] 00:03:17, localpref 100, from 198.51.100.3
    AS path: I, validation-state: unverified
    > to 10.0.1.2 via xe-0/0/2.0, label-switched-path PE1toPE3
1:198.51.100.3:11::9999999999999999::9999/304 AD/EVI
    *[BGP/170] 00:03:18, localpref 100, from 198.51.100.3

```

```

        AS path: I, validation-state: unverified
        > to 10.0.1.2 via xe-0/0/2.0, label-switched-path PE1toPE3
1:198.51.100.4:0::9999999999999999::FFFF:FFFF/304 AD/ESI
        *[BGP/170] 00:01:56, localpref 100, from 198.51.100.4
        AS path: I, validation-state: unverified
        > to 10.0.2.2 via xe-0/0/3.0, label-switched-path PE1toPE4
1:198.51.100.4:11::9999999999999999::9999/304 AD/EVI
        *[BGP/170] 00:01:56, localpref 100, from 198.51.100.4
        AS path: I, validation-state: unverified
        > to 10.0.2.2 via xe-0/0/3.0, label-switched-path PE1toPE4

```

```
user@PE3> show route table bgp.evpn.0
```

```
bgp.evpn.0: 7 destinations, 7 routes (4 active, 0 holddown, 3 hidden)
```

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

```

1:198.51.100.1:0::1111111111111111::FFFF:FFFF/304 AD/ESI
        *[BGP/170] 00:04:53, localpref 100, from 198.51.100.1
        AS path: I, validation-state: unverified
        > to 10.0.1.1 via xe-0/0/0.0, label-switched-path PE3toPE1
1:198.51.100.1:11::1111111111111111::1111/304 AD/EVI
        *[BGP/170] 00:04:53, localpref 100, from 198.51.100.1
        AS path: I, validation-state: unverified
        > to 10.0.1.1 via xe-0/0/0.0, label-switched-path PE3toPE1
1:198.51.100.2:0::1111111111111111::FFFF:FFFF/304 AD/ESI
        *[BGP/170] 00:04:53, localpref 100, from 198.51.100.2
        AS path: I, validation-state: unverified
        > to 10.0.3.1 via xe-0/0/1.0, label-switched-path PE3toPE2
1:198.51.100.2:11::1111111111111111::1111/304 AD/EVI
        *[BGP/170] 00:04:53, localpref 100, from 198.51.100.2
        AS path: I, validation-state: unverified
        > to 10.0.3.1 via xe-0/0/1.0, label-switched-path PE3toPE2

```

Meaning

The outputs show the ESI routes being shared across the VPWS to the remote PE devices.

The routes beginning with 1:198.51.100.x:0:: are the per-Ethernet-segment autodiscovery Type 1 EVPN routes originating from the remote PE devices. The route distinguishers (RDs) are derived at the global level of the devices.

The routes beginning with 1:198.51.100.x:11:: are the per-EVI autodiscovery Type 1 EVPN routes from the remote PE devices. The RDs are taken from the remote PE devices' routing instances.

Verifying Local EVPN Table Route Information

Purpose

Verify that the local EVPN routing tables are being populated.

Action

Verify that both local and remote reachability information is being added into the EVPN table.

```

user@PE1> show route table EVPN-VPWS.evpn.0

EVPN-VPWS.evpn.0: 7 destinations, 7 routes (5 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both

1:198.51.100.1:11::1111111111111111::1111/304 AD/EVI
      *[EVPN/170] 00:06:55
      Indirect
1:198.51.100.3:0::9999999999999999::FFFF:FFFF/304 AD/ESI
      *[BGP/170] 00:03:24, localpref 100, from 198.51.100.3
      AS path: I, validation-state: unverified
      > to 10.0.1.2 via xe-0/0/2.0, label-switched-path PE1toPE3
1:198.51.100.3:11::9999999999999999::9999/304 AD/EVI
      *[BGP/170] 00:03:25, localpref 100, from 198.51.100.3
      AS path: I, validation-state: unverified
      > to 10.0.1.2 via xe-0/0/2.0, label-switched-path PE1toPE3
1:198.51.100.4:0::9999999999999999::FFFF:FFFF/304 AD/ESI
      *[BGP/170] 00:02:03, localpref 100, from 198.51.100.4
      AS path: I, validation-state: unverified
      > to 10.0.2.2 via xe-0/0/3.0, label-switched-path PE1toPE4
1:198.51.100.4:11::9999999999999999::9999/304 AD/EVI
      *[BGP/170] 00:02:03, localpref 100, from 198.51.100.4
      AS path: I, validation-state: unverified
      > to 10.0.2.2 via xe-0/0/3.0, label-switched-path PE1toPE4

user@PE3> show route table EVPN-VPWS.evpn.0

EVPN-VPWS.evpn.0: 7 destinations, 7 routes (5 active, 0 holddown, 2 hidden)

```


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

```

1:198.51.100.1:0::1111111111111111::FFFF:FFFF/304 AD/ESI
    *[BGP/170] 00:05:01, localpref 100, from 198.51.100.1
    AS path: I, validation-state: unverified
    > to 10.0.1.1 via xe-0/0/0.0, label-switched-path PE3toPE1
1:198.51.100.1:11::1111111111111111::1111/304 AD/EVI
    *[BGP/170] 00:05:01, localpref 100, from 198.51.100.1
    AS path: I, validation-state: unverified
    > to 10.0.1.1 via xe-0/0/0.0, label-switched-path PE3toPE1
1:198.51.100.2:0::1111111111111111::FFFF:FFFF/304 AD/ESI
    *[BGP/170] 00:05:01, localpref 100, from 198.51.100.2
    AS path: I, validation-state: unverified
    > to 10.0.3.1 via xe-0/0/1.0, label-switched-path PE3toPE2
1:198.51.100.2:11::1111111111111111::1111/304 AD/EVI
    *[BGP/170] 00:05:01, localpref 100, from 198.51.100.2
    AS path: I, validation-state: unverified
    > to 10.0.3.1 via xe-0/0/1.0, label-switched-path PE3toPE2
1:198.51.100.3:11::9999999999999999::9999/304 AD/EVI
    *[EVPN/170] 00:05:25
    Indirect

```

Meaning

In addition to the remote ESI routes being shared across the VPWS, as explained in the previous section, the EVPN table on each PE device also includes a local ESI route. This Type 1 route represents the locally configured Ethernet segment, and is derived from the locally configured RD and ESI values.

RELATED DOCUMENTATION

[EVPN Multihoming Overview | 162](#)

[Overview of VPWS with EVPN Signaling Mechanisms | 1606](#)

[Configuring VPWS with EVPN Signaling Mechanisms | 1629](#)

vpws-service-id

show evpn vpws-instance

FAT Flow Labels in EVPN-VPWS Routing Instances

IN THIS SECTION

- [How to Enable FAT Pseudowire Flow Labels in an EVPN-VPWS Instance | 1657](#)
- [Verify FAT Flow Labels Are Enabled | 1660](#)

FAT Flow Labels Overview introduces how LDP-signaled pseudowires in an MPLS network can use flow-aware transport (FAT) flow labels (defined in RFC 6391, *Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network*) to load-balance traffic in virtual private LAN service (VPLS) and virtual private wire service (VPWS) networks.

Starting in Junos OS Release 21.1R1, you can enable provider edge devices in an EVPN-MPLS network to use flow-aware transport (FAT) flow labels to load-balance traffic across pseudowires in an EVPN-VPWS routing instance. In this environment, the local and remote devices establish an EVPN connection between the local and remote provider edge (PE) devices using BGP signaling, and can use LDP or RSVP tunnels to create the pseudowire.

How to Enable FAT Pseudowire Flow Labels in an EVPN-VPWS Instance

You can enable FAT flow labels in a routing instance of type `evpn-vpws` for pseudowires associated with the routing instance.

This environment supports enabling FAT flow label push and pop operations on pseudowire traffic only with a static configuration. The devices don't actively use the signaling mechanism described in RFC 6391 to ensure both ends communicate that they can handle flow labels. As a result, you must use *flow-label-transmit-static* and *flow-label-receive-static* (instead of *flow-label-transmit* and *flow-label-receive*) on all of the PE routers that will transmit, receive, and load-balance traffic with flow labels.



NOTE: We advise that you plan to configure these options during a maintenance window on those devices.

You can configure FAT flow label push and pop operations on the device at either the global routing instance level or at the individual interface level, as follows:

1. Configure the EVPN routing instance of type evpn-vpws. For example:

```
set routing-instances VPWS-SH instance-type evpn-vpws
set routing-instances VPWS-SH protocols evpn interface ge-0/0/1.100 vpws-service-id local 100
set routing-instances VPWS-SH protocols evpn interface ge-0/0/1.100 vpws-service-id remote 200
set routing-instances VPWS-SH interface ge-0/0/1.100
set routing-instances VPWS-SH route-distinguisher 10.255.0.1:100
set routing-instances VPWS-SH vrf-target target:100:100
```

Verify the configuration:

```
user@device# show routing-instances

VPWS-SH {
  instance-type evpn-vpws;
  interface ge-0/0/1.100;
  route-distinguisher 10.255.0.1:100;
  vrf-target target:100:100;
  protocols {
    evpn {
      interface ge-0/0/1.100 {
        vpws-service-id {
          local 100;
          remote 200;
        }
      }
    }
  }
}
```

2. To configure FAT flow label push and pop operations at the evpn-vpws routing instance level:

```
set routing-instances <evpn-vpws-routing-instance-name> protocols evpn flow-label-transmit-
static
set routing-instances <evpn-vpws-routing-instance-name> protocols evpn flow-label-receive-
static
```

For example:

```
set routing-instances VPWS-SH protocols evpn flow-label-transmit-static
set routing-instances VPWS-SH protocols evpn flow-label-receive-static
```

Verify the configuration:

```
user@device# show routing-instances

VPWS-SH {
  instance-type evpn-vpws;
  interface ge-0/0/1.100;
  route-distinguisher 10.255.0.1:100;
  vrf-target target:100:100;
  protocols {
    evpn {
      interface ge-0/0/1.100 {
        vpws-service-id {
          local 100;
          remote 200;
        }
      }
      flow-label-transmit-static;
      flow-label-receive-static;
    }
  }
}
```

3. Alternatively, to enable a specific interface in the `evpn-vpws` routing instance to push and pop FAT flow labels:

```
set routing-instances <evpn-vpws-routing-instance-name> protocols evpn interface <interface-
name> flow-label-transmit-static
set routing-instances <evpn-vpws-routing-instance-name> protocols evpn interface <interface-
name> flow-label-receive-static
```

For example:

```
set routing-instances VPWS-SH protocols evpn interface ge-0/0/1.100 flow-label-transmit-static
set routing-instances VPWS-SH protocols evpn interface ge-0/0/1.100 flow-label-receive-static
```

Verify the configuration:

```
user@device# show routing-instances

VPWS-SH {
  instance-type evpn-vpws;
  interface ge-0/0/1.100;
  route-distinguisher 10.255.0.1:100;
  vrf-target target:100:100;
  protocols {
    evpn {
      interface ge-0/0/1.100 {
        vpws-service-id {
          local 100;
          remote 200;
        }
        flow-label-transmit-static;
        flow-label-receive-static;
      }
    }
  }
}
```

Verify FAT Flow Labels Are Enabled

You can enter the `show evpn vpws-instance` CLI command to see if an `evpn-vpws` routing instance is configured to handle FAT flow labels. The output from this command displays **Yes** in the **Flow-Label-Tx** or **Flow-Label-Rx** output fields if you configured the device to insert or remove FAT flow labels on pseudowire traffic in the routing instance. These fields display **No** if FAT flow label operations are not enabled.

RELATED DOCUMENTATION

FAT Flow Labels Overview

flow-label-receive-static

flow-label-transmit-static

Configuring EVPN-VPWS over SRv6

EVPN VPWS provides point to point Layer 2 VPN service using EVPN signaling. EVPN-VPWS supports both single homed and multihomed (single-active or all-active) devices. EVPN-VPWS over SRv6 (Segment Routing over IPv6). SRv6 uses the IPv6 Segment Routing Header (SRH) extension to encode an order list of network instructions. The network instruction contains explicit information about SRv6 nodes that are available for packet processing on the path. The instruction also include task or function information for the SRv6 node in the SRv6 network. The SRH contains a list of 128-bit segment identifiers (SIDs) in the form of an IPv6 addresses. SIDs consist of the following:

- **Locator**—The locator is the first part of the SID and consists of the most significant bits. It represents the address of a particular SRv6 node. The locator is similar to a network address. It is used to route the packet.
- **Function**—The function is the second part of the SID. It defines the packet processing function that the node identified by the locator performs locally. Junos OS supports End.DX2 function for EVPN-VPWS. End.DX2 specifies endpoint decapsulation and L2 cross-connect behavior.

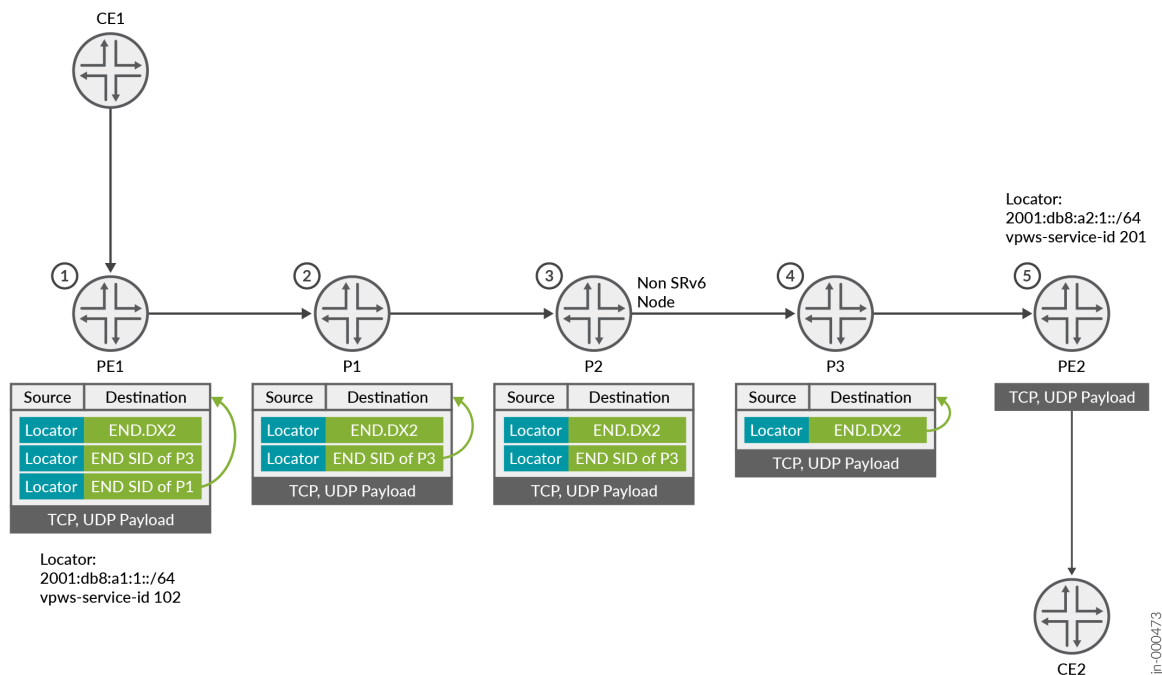
Benefits of EVPN-VPWS over SRv6

EVPN-VPWS over an SRv6 underlay has the following benefits IPv6 network:

1. Network Programming depends entirely on the IPv6 header and the header extension to transport a packet, eliminating protocols such as MPLS. This ensures a seamless deployment without any major hardware or software upgrade in a core IPv6 network.
2. Packets can be transported through an SRv6 ingress node even when the transit routers are not SRv6-capable. This eliminates the need to deploy segment routing on all nodes in an IPv6 network.

[Figure 166 on page 1662](#) illustrates how the SRH is processed by the nodes in a SRv6 topology .

Figure 166: SRH and Outer IPv6 Header Processing in a SRv6 Topology



1. PE1 encapsulates the payload with an SRH. The SRH list contains three SIDs. Each SID represent a SRv6 node along the segment path. The function on the last SID is END.DX2 endpoint.
2. P1 pops and processes the first SID at the bottom of the SRH list and copies the next SID to the outer destination. The SRH list contains 2 SIDs.
3. P2 is a non-SRv6 node. P2 forwards the packet on the current segment path with no further processing.
4. P3 pops and processes the second SID and copies the third SID in the SRH list.
5. PE2 pops and processes the third SID. END.DX2 identifies the CE facing interface and PE2 forwards the packet.

EVPN-VPWS builds upon an SRv6 baseline configuration. For more information about configuring SRv6, see [Understanding SRv6 Network Programming and Layer 3 Services over SRv6 in BGP](#).

CLI Quick Configuration

To quickly configure EVPN-VPWS over SRv6, 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.

PE1

```

set chassis network-services enhanced-ip
set routing-instances EVPN-VPWS1 instance-type evpn-vpws
set routing-instances EVPN-VPWS1 protocols evpn encapsulation srv6
set routing-options source-packet-routing srv6 locator LOC1 2001:db8:a1:1::/64
set routing-options resolution preserve-nexthop-hierarchy
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id local 102
remote 201
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-
packet-routing srv6 locator LOC1 end-dx2-sid 2001:db8:a1:1:101::
set protocols bgp group IBGPv6 family evpn signaling advertise-srv6-service
set protocols bgp group IBGPv6 family evpn signaling accept-srv6-service
set routing-instances EVPN-VPWS1 route-distinguisher 65000:100
set routing-instances EVPN-VPWS1 vrf-target target:65000:200
set routing-instances EVPN-VPWS1 interface ge-0/0/1.1

```

PE2

```

set chassis network-services enhanced-ip
set routing-instances EVPN-VPWS1 instance-type evpn-vpws
set routing-instances EVPN-VPWS1 protocols evpn encapsulation srv6
set routing-options source-packet-routing srv6 locator LOC1 2001:db8:a1:2::/64
set routing-options resolution preserve-nexthop-hierarchy
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id local 201
remote 102
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-
packet-routing srv6 locator LOC1 end-dx2-sid 2001:db8:a1:2:101::
set protocols bgp group IBGPv6 family evpn signaling advertise-srv6-service
set protocols bgp group IBGPv6 family evpn signaling accept-srv6-service
  routing-instances EVPN-VPWS1 route-distinguisher 65000:200
set routing-instances EVPN-VPWS1 vrf-target target:65000:200
set routing-instances EVPN-VPWS1 interface ge-0/0/1.1

```


Procedure

We describe these steps on the PE1 device. We note the differences in configurations between PE1 and PE2 when it applies. To configure EVPN-VPWS over SRv6 to support static SID, you must do the following:

1. Enable enhanced-ip support on all MX devices.

```
[edit]
user@PE1# set chassis network-services enhanced-ip
```

2. Configure support for SRv6 and the locator address.

PE1

```
[edit]
user@PE1# set routing-options source-packet-routing srv6 locator LOC1 2001:db8:a1:1::/64
```

PE2

```
[edit]
user@PE2# set routing-options source-packet-routing srv6 locator LOC1 2001:db8:a1:2::/64
```

3. Enable expanded nexthop hierarchy support for source packet routing.

```
[edit]
user@R1# set routing-options resolution preserve-nexthop-hierarchy
```

4. Enable an evpn-vpws routing instance.

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 instance-type evpn-vpws
```

5. Configure the SRv6 encapsulation type for the EVPN-VPWS1 routing instance.

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn encapsulation srv6
```

6. Configure the interface with the local and remote VPWS SID for the EVPN-VPWS1 routing instance.

PE1

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id local 102 remote 201
```

PE2

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id local 201 remote 102
```

7. Configure the locator to support END.DX2 on the interface in the EVPN-VPWS1 routing instance.

PE1

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-packet-routing srv6 locator LOC1 end-dx2-sid 2001:db8:a1:1:101::
```

PE2

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-packet-routing srv6 locator LOC1 end-dx2-sid 2001:db8:a1:2:101::
```

8. Enable the BGP protocol to advertise and to accept the EVPN NLRI for SRv6 services.

```
[edit]
user@PE1# set protocols bgp group IBGPv6 family evpn signaling advertise-srv6-service
user@PE1# set protocols bgp group IBGPv6 family evpn signaling accept-srv6-service
```

9. **PE1**

Configure the vrf target and route distinguisher for the routing instance.

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 route-distinguisher 6500:100
user@PE1# set routing-instances EVPN-VPWS1 vrf-target target:65000:200
```

PE2

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 route-distinguisher 6500:100
user@PE1# set routing-instances EVPN-VPWS1 vrf-target target:65000:300
```

10. Assign the interface to the routing instance.

```
set routing-instances EVPN-VPWS1 interface ge-0/0/1.1
```

11.



NOTE: Starting in Junos OS Release 24.1R1 and Junos OS Evolved Release 24.1R1, you do not need to configure a routing policy for EVPN-VPWS over SRv6.

Dynamic SID Allocation

Dynamic SID allocation allows you to provision the Junos device by only specifying the locator name. To enable dynamic provisioning, configure the locator name at the `[edit routing-instance routing-instance-name instance-type protocols evpn interface interface-name vpws-service-id source-packet-routing srv6]` hierarchy. The device dynamically allocates a SID with `en-dx2-sid` to the corresponding locator prefix when the service is needed. The following is the sample configuration for the dynamically allocated SID on EVPN-VPWS.

```
set routing-instances EVPN-VPWS1 instance-type evpn-vpws
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id local 3040
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id remote 20
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-
packet-routing srv6 locator LOC2
set routing-instances EVPN-VPWS1 protocols evpn encapsulation srv6
set routing-instances EVPN-VPWS1 interface ge-0/0/1.1
set routing-instances EVPN-VPWS1 route-distinguisher 65000:100
set routing-instances EVPN-VPWS1 vrf-target target:65000:200
```

```
set routing-options source-packet-routing srv6 locator LOC2 2001:db8:b1:1::/64
```

RELATED DOCUMENTATION

https://www.juniper.net/documentation/en_US/day-one-books/DayOne-Intro-SRv6.pdf

Configuring Micro-SIDs in EVPN-VPWS

IN THIS SECTION

- [Benefits of uSIDs | 1669](#)
- [Configuring EVPN-VPWS over SRv6 Network with uSIDs | 1669](#)
- [Checking the Static SID Range | 1672](#)
- [Dynamic uSID Allocation | 1673](#)

The segment routing headers (SRHs) for SRv6 can have a long list of SIDs when data packets are routed through many SRv6 nodes before it reaches its destination address (DA). A long list of segment identifiers (SIDs) adds overhead to the data payload and reduces the efficiency of the payload. Micro-SIDs (uSIDs) extend SRv6 network programming by compressing up to 6 SRv6 SIDs into one SRv6 address within an SRH.

Use [Feature Explorer](#) to confirm platform and release support for specific features, including Micro-SIDs in EVPN-VPWS.

For uSID, the router divides the 128-bit SID into the following:

- **Prefix/Block**—The prefix contains the locator address of the network.
- **List of uSID instructions**—The list of uSID contains either micro-node IDs or a uSID function/behavior.
- **Argument**—The argument is an optional field in the SRH.

[Figure 167 on page 1668](#) shows the DA as a packet moves through different nodes in SRv6 topology 1. The nodes, ID/function, and SIDs advertised by the nodes are listed in [Table 89 on page 1668](#).

Figure 167: Micro-SID in an SRv6 Network

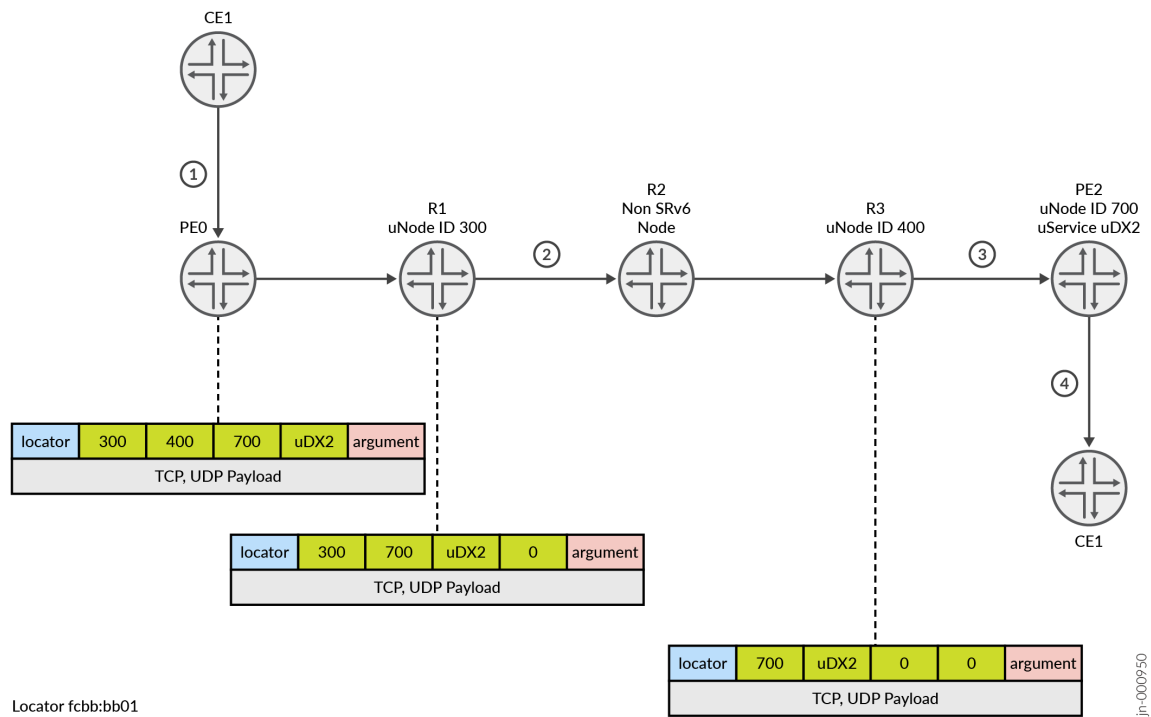


Table 89: SRv6 micro-SID

Node	Micro-Node ID/Micro-Node Function	SIDs Advertised by each Node
R1	300	2001:db8:300:0:0:0:0:0
R3	400	2001:db8:400:0:0:0:0:0
PE2	700	2001:db8:700:0:0:0:0:0
PE2	f001	2001:db8:700:f001:0:0:0:0

1. At the ingress device, PE1 compresses the micro-SIDs for the nodes (R1, R3, and PE2) into one DA. 2001:db8:300:400:700:f001.
2. R1 processes the DA by consuming its own uSID 300 and forwards the packet with a DA of 2001:db8:400:700:f001:0.

3. R3 processes the DA by consuming its own uSID of 400 and forwards the packet with a DA 2001:db8:700:f001:0:0.
4. At the egress device, PE2 consumes its own uSID of 700 and processes the microservices function.

Benefits of uSIDs

- Reduces network bandwidth by reducing the number of SRv6 addresses in the SRH.
- Reduces the SRH processing overhead on the node.

Configuring EVPN-VPWS over SRv6 Network with uSIDs

CLI Quick Configuration

To quickly configure EVPN-VPWS over SRv6 with uSIDs, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your configuration, copy and paste the commands into the CLI at the [edit] hierarchy level. Enter commit from configuration mode.

```
set chassis network-services enhanced-ip
set routing-options resolution preserve-nexthop-hierarchy
set routing-instances EVPN-VPWS1 instance-type evpn-vpws
set routing-instances EVPN-VPWS1 protocols evpn encapsulation srv6
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id local 102
remote 201
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-
packet-routing srv6 locator u_loc
set routing-options source-packet-routing srv6 block usid_blk_with_statics 2001:db8::/32
set routing-options source-packet-routing srv6 block usid_blk_with_statics local-micro-sid
maximum-static-sids 2000
set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-
packet-routing srv6 locator micro-dx2-sid 0xF900
set routing-options source-packet-routing srv6 locator u_loc 2001:db8:100::/48
set routing-options source-packet-routing srv6 locator u_loc micro-sid block-name
usid_blk_with_statics
set routing-instances EVPN-VPWS1 route-distinguisher 65000:100
set routing-instances EVPN-VPWS1 vrf-target target:65000:200
set routing-instances EVPN-VPWS1 interface ge-0/0/1.1
```

We describe the uSID configuration on PE1. To provision a static uSID, you must first configure a static local range of addresses that can be used on all the devices. Use the same statements on PE2 unless differences in the configuration on PE2 are called out.

1. Enable enhanced-ip support on all MX devices.

```
[edit]
user@PE1# set chassis network-services enhanced-ip
```

2. Enable expanded next-hop hierarchy support for source packet routing.

```
[edit]
user@R1# set routing-options resolution preserve-nexthop-hierarchy
```

3. Enable an evpn-vpws routing instance.

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 instance-type evpn-vpws
```

4. Configure the SRv6 encapsulation type for the EVPN-VPWS1 routing instance.

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn encapsulation srv6
```

5. Configure the interface with the local and remote VPWS SID for the EVPN-VPWS1 routing instance.

PE1

```
[edit]
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id local 102 remote 201
```

PE2

```
[edit]
user@PE2# set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id local 201 remote 102
```

6. Enable uSID for the EVPN-VPWS routing instance.

```
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-packet-routing srv6 locator u_loc
```

7. To configure a block for the uSID, specify the prefix and length for a block of IPv6 address. This reserves the block for local static micro-SIDs. For maximum compression, all nodes should have same block prefix.

When defining the uSID function for the locator associated with this block (as described in Step 9), be sure that the micro-SID value falls within the static SID range of the local micro-SIDs. See ["Checking the Static SID Range" on page 1672](#)

```
user@PE1# set routing-options source-packet-routing srv6 block usid_blk_with_statics 2001:db8::/32
```

8. Specify the maximum number of static SID that will be used as micro-SIDs.

```
user@PE1# set routing-options source-packet-routing srv6 block usid_blk_with_statics local-micro-sid maximum-static-sids 2000
```

9. Configure the uSID function for the locator. The uSID function for micro-dx2 is 0xF900.



NOTE: The micro-SID value must be in the static SID range of local micro-SIDs. You can check that the static SID range for the local micro-SID with the `show srv6 block` command. See ["Checking the Static SID Range" on page 1672](#).



NOTE: Junos displays hex values as decimal values in the output of the `show configuration configuration`

```
user@PE1# set routing-instances EVPN-VPWS1 protocols evpn interface ge-0/0/1.1 vpws-service-id source-packet-routing srv6 locator micro-dx2-sid 0xF900
```

10. Configure a range of address that the locator can use.

PE1

```
user@PE1# set routing-options source-packet-routing srv6 locator u_loc 2001:db8:100::/48
```

PE2

```
user@PE2# set routing-options source-packet-routing srv6 locator u_loc 2001:db8:200::/48
```

11. Enable the uSID locator by specifying the name of the locator and block name that was reserved for the uSID.

```
user@PE1# set routing-options source-packet-routing srv6 locator u_loc micro-sid block-name  
usid_blk_with_statics
```

12. Configure the vrf target and route distinguisher for the routing instance.

```
[edit]  
user@PE1# set routing-instances EVPN-VPWS1 route-distinguisher 6500:100  
user@PE1# set routing-instances EVPN-VPWS1 vrf-target target:65000:200
```

13. Assign the interface to the routing instance.

```
set routing-instances EVPN-VPWS1 interface ge-0/0/1.1
```

Checking the Static SID Range

You must assign a uSID value that is inside the static SID range of the local uSIDs. To display the range of local static SIDs in the local uSID, use the `show srv6 block` command. The acceptable uSID value is between 0xF830-0xFFFFF.

```
user@host> show srv6 block usid_blk_with_statics  
Block: usid_blk_with_statics  
Block Prefix: 2001:db8::, Block length: 32, Micro-sid length: 16  
Global Micro SIDs:  
Static SID range: 0x0-0xDFFF, Dynamic SID range: -  
Allocated static SID count: 1, Allocated dynamic SID count: 0  
Available static SID count: 57343, Available dynamic SID count: 0
```

Local Micro SIDs:

Static SID range: 0xF830-0xFFFF, Dynamic SID range: 0xE000-0xF82F

Allocated static SID count: 0, Allocated dynamic SID count: 1

Available static SID count: 2000, Available dynamic SID count: 6191

Dynamic uSID Allocation

The following is a sample configuration for configuring dynamically allocated uSID EVPN-VPWS instance. It builds on the EVPN-VPWS dynamic SID allocation configuration EVPN-VPWS.

```
set routing-options source-packet-routing srv6 block usid_blk_with_statics 2001:db8::/32
set routing-options source-packet-routing srv6 locator u_loc 2001:db8:100::/48
set routing-options source-packet-routing srv6 locator u_loc micro-sid
set routing-instances evpn-vpws-mh instance-type evpn-vpws
set routing-instances evpn-vpws-mh protocols evpn interface ae0.0 vpws-service-id local 103
set routing-instances evpn-vpws-mh protocols evpn interface ae0.0 vpws-service-id remote 301
set routing-instances evpn-vpws-mh protocols evpn interface ae0.0 vpws-service-id source-packet-
routing srv6 locator u_loc
set routing-instances evpn-vpws-mh protocols evpn interface ae0.0 vpws-service-id source-packet-
routing srv6 locator micro-dx2-sid
set routing-instances evpn-vpws-mh route-distinguisher 65000:100
set routing-instances evpn-vpws-mh vrf-target target:65000:200
set routing-instances evpn-vpws-mh interface ge-0/0/1.1
```

RELATED DOCUMENTATION

[Configuring EVPN-VPWS over SRv6 | 1661](#)

block

vpws-service-id

show evpn vpws-instance

Configuring EVPN-VPWS over SRv6 with Traffic Engineering

IN THIS SECTION

- [EVPN-VPWS FXC with SRv6 | 1678](#)

Starting in Junos OS Evolved 25.2R1, the ACX Series routers support EVPN-VPWS over SRv6 traffic engineering (TE) tunnels with fallback capability. While EVPN services themselves do not support the fallback switchover of routes, you can configure segment routing policies to define a secondary route. If the primary path becomes unavailable, the device automatically switches to the secondary route as a backup.

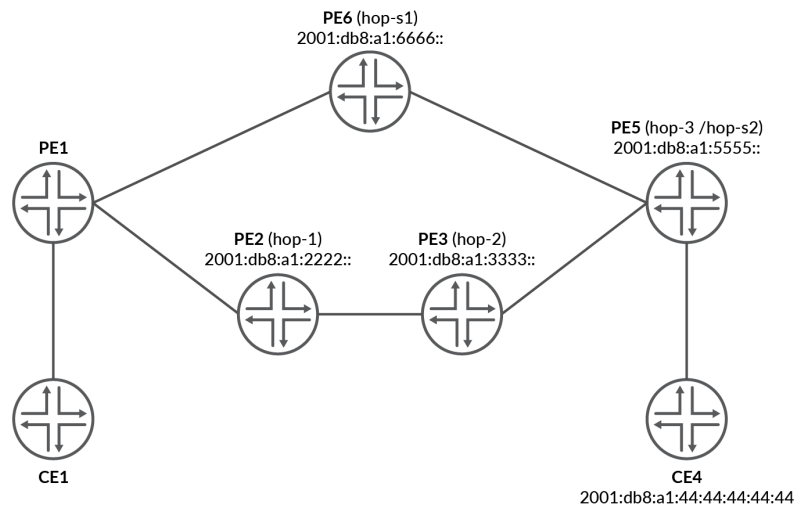
To configure EVPN-VPWS over SRv6 TE routes, you must define SRv6 TE routes and their associated transport classes. When SRv6 TE is configured with a color (transport class), the resulting route is installed in the colored routing table `rti-tc-<color>.inet6.3`. This is referred to as the intent route. The associated locator prefix route is installed in the default table `inet6.3` and is known as the non-intent route or best-effort route.

By utilizing SRv6 TE routes, you can ensure that intended routes with lower metrics are prioritized while maintaining best-effort routes as fallback options. This ensures that your primary routes are optimized for performance, and in the event of a failure, there is an automatic switch to the best-effort routes to maintain service continuity.

By default, Junos OS Evolved enables fallback to best-effort routes for auto-created transport classes. To configure TE routes without fallback, you must run the CLI command `set protocols sr-te transport-class fallback none` at the top of the hierarchy. This command ensures that only the preferred TE routes are utilized where best-effort routes are not acceptable.

Junos OS Evolved supports both single-homing and multi-homing networks for these services.

Figure 168: EVPN-VPWS SRTE topology



In [Figure 168 on page 1675](#), we have a topology with two possible paths from CE1 to CE4. The basic steps for configuring EVPN-VPWS over SRv6 with traffic engineering on the ingress device (PE1) is as follow.

1. Configure the node locator SID within the IS-IS protocol.

```
protocols {
  isis {
    source-packet-routing {
      srv6 {
        locator One micro-node-sid;
      }
    }
  }
}
```

2. Configure EVPN-VPWS services.

```
routing-instances {
  evpn-vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        interface ae0.1 {
```

```

        vpws-service-id {
            local 111;
            remote 555;
            source-packet-routing {
                srv6 locator one;
            }
        }
    }
}
interface ae0.1;
route-distinguisher 10.255.1.1:1;
vrf-export transport-class-vrf-export-evpn1;
}
}

```

3. Configure support for micro-SID and SRv6.

```

routing-options {
    source-packet-routing {
        srv6 {
            locator one{
                2001:db8:a1::/48;
                micro-sid;
            }
        }
    }
}

```

4. Define the transport class at the ingress or egress node. For this example, we define a gold transport class and assign community value.

```

routing-options {
    transport-class {
        auto-create;
        name transport-class-gold {
            color 100;
        }
    }
}

```

5. Configure import or export policies to route traffic. For this example, we use an export policy.

```

policy-statement transport-class-vrf-export-evpn1 {
    term a {
        then {
            community add transport-class-gold;
            community add transport-class-route-target1;
            accept;
        }
    }
    term b {
        then reject;
    }
}
community transport-class-gold members color:0:100;
community transport-class-route-target1 members target:100:1;

```

6. Configure source path route to the remote device and the hops for the primary and secondary paths.



NOTE: Use the loopback address of the remote device for the tunnel endpoint.

```

source-packet-routing {
    segment-list sl-primary {
        srv6;
        hop-1 {
            micro-srv6-sid {
                2001:db8:a1:2222::;
            }
        }
        hop-2 {
            micro-srv6-sid {
                2001:db8:a1:3333::;
            }
        }
        hop-3 {
            micro-srv6-sid {
                2001:db8:a1:5555::;
            }
        }
    }
}

```

```

segment-list sl-secondary {
    srv6;
    hop-s1 {
        micro-srv6-sid {
            2001:db8:a1:6666::;
        }
    }
    hop-s2 {
        micro-srv6-sid {
            2001:db8:a1:5555::;
        }
    }
}

source-routing-path sr-path-CE4 {
    srv6;
    to 2001:db8:a1:44:44:44:44:44;
    color 100;
    primary {
        sl-primary;
    }
    secondary {
        sl-secondary;
    }
}
}

```

EVPN-VPWS FXC with SRv6

The flexible cross-connect (FXC) with SRv6 provides a method for scalable pseudowire (PW) signaling across multiple user network interfaces (UNIs) or customer edge (CE) devices. This feature simplifies the encapsulation process by using a single label for MPLS encapsulation or a single DX2V SRv6 SID, allowing for streamlined auto-discovery per Ethernet VPN instance (EVI) route.

FXC supports both VLAN unaware and VLAN aware configurations.

- VLAN unaware - The FXC sends a single auto-discovery (A-D) per EVI route, simplifying signaling by treating all VLANs uniformly.
- VLAN aware - The FXC sends one A-D per EVI route for each VLAN, enabling granular control over VLAN-specific signaling.

Both VLAN-unaware and VLAN-aware modes support static and dynamic segment routing header (SRH) as well as static and dynamic micro-SID segment routing. The basic steps to configure each mode is shown below.



NOTE: Egress protection is not supported for EVPN-VPWS FXC instances.

Configure Static SRH END.DX2V for EVPN-VPWS FXC in VLAN Unaware

1. Configure the SRv6 locator and disable reduced SRH to ensure classic SRH.

```
set routing-options source-packet-routing srv6 locator loc 1001::/64
set routing-options source-packet-routing srv6 no-reduced-srh
```

2. Configure the EVPN-VPWS routing instance.

```
set routing-instances vpws1 instance-type evpn-vpws
set routing-instances vpws1 interface et-0/0/20.0
set routing-instances vpws1 interface et-0/0/20.1
set routing-instances vpws1 route-distinguisher 1:7
set routing-instances vpws1 vrf-target target:1:77
```

3. Configure EVPN protocol for VLAN-unaware FXC.

```
set routing-instances vpws1 protocols evpn encapsulation srv6
set routing-instances vpws1 protocols evpn flexible-cross-connect-vlan-unaware
```

4. Configure EVPN group with interfaces and service IDs.

```
set routing-instances vpws1 protocols evpn group g1 interface et-0/0/20.0
set routing-instances vpws1 protocols evpn group g1 interface et-0/0/20.1
set routing-instances vpws1 protocols evpn group g1 service-id local 666
set routing-instances vpws1 protocols evpn group g1 service-id remote 555
```


5. Configure static END.DX2V SID.

```
set routing-instances vpws1 protocols evpn source-packet-routing srv6 locator loc end-dx2v-
sid 1001:0:0:0:3:0:0:0
```

The following output shows a sample static SRH FXC instance configured in VLAN-unaware mode.

```
routing-options {
  source-packet-routing {
    srv6 {
      locator loc {
        prefix 1001::/64;
      }
      no-reduced-srh;
    }
  }
}
routing-instances {
  vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        flexible-cross-connect-vlan-unaware;
+      source-packet-routing {
+        srv6 {
+          locator loc end-dx2v-sid 1001:0:0:0:3:0:0:0;
+        }
+      }
      encapsulation srv6;
      group g1 {
        interface et-0/0/20.0;
        interface et-0/0/20.1;
        service-id {
          local 666;
          remote 555;
        }
      }
    }
  }
  interface et-0/0/20.0;
  interface et-0/0/20.1;
```

```

    route-distinguisher 1:7;
    vrf-target target:1:77;
}
}

```

Configure Dynamic SRH END.DX2V for EVPN-VPWS FXC in VLAN Unaware

1. Configure the SRv6 locator and disable reduced SRH to ensure classic SRH.

```

set routing-options source-packet-routing srv6 locator loc 1001::/64
set routing-options source-packet-routing srv6 no-reduced-srh

```

2. Configure the EVPN-VPWS routing instance.

```

set routing-instances vpws1 instance-type evpn-vpws
set routing-instances vpws1 interface et-0/0/20.0
set routing-instances vpws1 interface et-0/0/20.1
set routing-instances vpws1 route-distinguisher 1:7
set routing-instances vpws1 vrf-target target:1:77

```

3. Configure EVPN protocol for VLAN-unaware FXC.

```

set routing-instances vpws1 protocols evpn encapsulation srv6
set routing-instances vpws1 protocols evpn flexible-cross-connect-vlan-unaware

```

4. Configure EVPN group with interfaces and service IDs.

```

set routing-instances vpws1 protocols evpn group g1 interface et-0/0/20.0
set routing-instances vpws1 protocols evpn group g1 interface et-0/0/20.1
set routing-instances vpws1 protocols evpn group g1 service-id local 666
set routing-instances vpws1 protocols evpn group g1 service-id remote 555

```

5. Configure dynamic END.DX2V SID assignment.

```

set routing-instances vpws1 protocols evpn source-packet-routing srv6 locator loc

```

The following output shows a sample dynamic SRH FXC instance configured in VLAN-unaware mode.

```

routing-options {
  source-packet-routing {
    srv6 {
      locator loc {
        prefix 1001::/64;
      }
      no-reduced-srh;
    }
  }
}
routing-instances {
  vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        flexible-cross-connect-vlan-unaware;
+      source-packet-routing {
+        srv6 {
+          locator loc;
+        }
+      }
      encapsulation srv6;
      group g1 {
        interface et-0/0/20.0;
        interface et-0/0/20.1;
        service-id {
          local 666;
          remote 555;
        }
      }
    }
    interface et-0/0/20.0;
    interface et-0/0/20.1;
    route-distinguisher 1:7;
    vrf-target target:1:77;
  }
}

```

Configure Static micro-SID END.DX2V for EVPN-VPWS FXC in VLAN Unaware

1. Reserve the micro-SID block.

```
set routing-options source-packet-routing srv6 block usid_blk_with_statics fcbb:bb01::/32
set routing-options source-packet-routing srv6 block usid_blk_with_statics local-micro-sid
maximum-static-sids 2000
```

2. Configure the SRv6 locator from the reserved micro-SID block.

```
set routing-options source-packet-routing srv6 locator u_loc fcbb:bb01:100::/48
set routing-options source-packet-routing srv6 locator u_loc micro-sid block-name
usid_blk_with_statics
```

3. Configure the EVPN-VPWS routing instance.

```
set routing-instances vpws1 instance-type evpn-vpws
set routing-instances vpws1 interface et-0/0/20.0
set routing-instances vpws1 interface et-0/0/20.1
set routing-instances vpws1 route-distinguisher 1:7
set routing-instances vpws1 vrf-target target:1:77
```

4. Configure EVPN protocol for VLAN-unaware FXC.

```
set routing-instances vpws1 protocols evpn encapsulation srv6
set routing-instances vpws1 protocols evpn flexible-cross-connect-vlan-unaware
```

5. Configure EVPN group with interfaces and service IDs.

```
set routing-instances vpws1 protocols evpn group g1 interface et-0/0/20.0
set routing-instances vpws1 protocols evpn group g1 interface et-0/0/20.1
set routing-instances vpws1 protocols evpn group g1 service-id local 666
set routing-instances vpws1 protocols evpn group g1 service-id remote 555
```

6. Configure static micro-SID END.DX2V.

```
set routing-instances vpws1 protocols evpn source-packet-routing srv6 locator u_loc micro-
dx2v-sid 0xFF01
```

The following output shows a sample static micro-SID FXC instance configured in VLAN-unaware mode.

```
routing-options {
  source-packet-routing {
    srv6 {
      block usid_blk_with_statics {
        prefix fcbb:bb01::/32;
        local-micro-sid {
          maximum-static-sids 2000;
        }
      }
      locator u_loc {
        prefix fcbb:bb01:100::/48;
        micro-sid {
          block-name usid_blk_with_statics;
        }
      }
    }
  }
}
routing-instances {
  vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        flexible-cross-connect-vlan-unaware;
+       source-packet-routing {
+         srv6 {
+           locator u_loc micro-dx2v-sid 0xFF01;
+         }
+       }
      encapsulation srv6;
      group g1 {
        interface et-0/0/20.0;
        interface et-0/0/20.1;
        service-id {
```

```

        local 666;
        remote 555;
    }
}
}
interface et-0/0/20.0;
interface et-0/0/20.1;
route-distinguisher 1:7;
vrf-target target:1:77;
}
}

```

Configure Dynamic micro-SID END.DX2V for EVPN-VPWS FXC in VLAN Unaware

1. Configure the SRv6 locator with micro-sid.

```

set routing-options source-packet-routing srv6 locator loc 1001::/64
set routing-options source-packet-routing srv6 locator loc micro-sid

```

2. Configure the EVPN-VPWS routing instance.

```

set routing-instances vpws1 instance-type evpn-vpws
set routing-instances vpws1 interface et-0/0/20.0
set routing-instances vpws1 interface et-0/0/20.1
set routing-instances vpws1 route-distinguisher 1:7
set routing-instances vpws1 vrf-target target:1:77

```

3. Configure EVPN protocol for VLAN-unaware FXC.

```

set routing-instances vpws1 protocols evpn encapsulation srv6
set routing-instances vpws1 protocols evpn flexible-cross-connect-vlan-unaware

```

4. Configure EVPN group with interfaces and service IDs.

```

set routing-instances vpws1 protocols evpn group g1 interface et-0/0/20.0
set routing-instances vpws1 protocols evpn group g1 interface et-0/0/20.1

```

```
set routing-instances vpws1 protocols evpn group g1 service-id local 666
set routing-instances vpws1 protocols evpn group g1 service-id remote 555
```

5. Configure dynamic micro-SID END.DX2V assignment.

```
set routing-instances vpws1 protocols evpn source-packet-routing srv6 locator loc
```

The following output shows a sample dynamic micro-SID FXC instance configured in VLAN-unaware mode.

```
routing-options {
  source-packet-routing {
    srv6 {
      locator loc {
        prefix 1001::/64;
        micro-sid;
      }
    }
  }
}
routing-instances {
  vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        flexible-cross-connect-vlan-unaware;
+      source-packet-routing {
+        srv6 {
+          locator loc;
+        }
+      }
      encapsulation srv6;
      group g1 {
        interface et-0/0/20.0;
        interface et-0/0/20.1;
        service-id {
          local 666;
          remote 555;
        }
      }
    }
  }
}
```

```

        interface et-0/0/20.0;
        interface et-0/0/20.1;
        route-distinguisher 1:7;
        vrf-target target:1:77;
    }
}

```

Configure Static SRH END.DX2V for EVPN-VPWS FXC in VLAN Aware

1. Configure the SRv6 locator and disable reduced SRH to ensure classic SRH.

```

set routing-options source-packet-routing srv6 locator loc 1001::/64
set routing-options source-packet-routing srv6 no-reduced-srh

```

2. Configure the EVPN-VPWS routing instance.

```

set routing-instances vpws1 instance-type evpn-vpws
set routing-instances vpws1 interface et-0/0/20.0
set routing-instances vpws1 interface et-0/0/20.1
set routing-instances vpws1 route-distinguisher 1:7
set routing-instances vpws1 vrf-target target:1:77

```

3. Configure EVPN protocol for VLAN-aware FXC.

```

set routing-instances vpws1 protocols evpn encapsulation srv6
set routing-instances vpws1 protocols evpn flexible-cross-connect-vlan-aware

```

4. Configure PW service IDs.

```

set routing-instances vpws1 protocols evpn interface et-0/0/20.0 vpws-service-id local 666
set routing-instances vpws1 protocols evpn interface et-0/0/20.0 vpws-service-id remote 555
set routing-instances vpws1 protocols evpn interface et-0/0/20.1 vpws-service-id local 444
set routing-instances vpws1 protocols evpn interface et-0/0/20.1 vpws-service-id remote 333

```


5. Configure static END.DX2V SID.

```
set routing-instances vpws1 protocols evpn source-packet-routing srv6 locator loc end-dx2v-
sid 1001:0:0:0:3:0:0:0
```

The following output shows a sample static SRH FXC instance configured in VLAN-aware mode.

```
routing-options {
  source-packet-routing {
    srv6 {
      locator loc {
        prefix 1001::/64;
      }
      no-reduced-srh;
    }
  }
}
routing-instances {
  vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        interface et-0/0/20.0 {
          vpws-service-id {
            local 666;
            remote 555;
          }
        }
        interface et-0/0/20.1 {
          vpws-service-id {
            local 444;
            remote 333;
          }
        }
      }
    }
    flexible-cross-connect-vlan-aware;

+     source-packet-routing {
+       srv6 {
+         locator loc end-dx2v-sid 1001:0:0:0:3:0:0:0;
+       }
+     }
```

```

        encapsulation srv6;

    }
}
interface et-0/0/20.0;
interface et-0/0/20.1;
route-distinguisher 1:7;
vrf-target target:1:77;
}
}

```

Configure Dynamic SRH END.DX2V for EVPN-VPWS FXC in VLAN Aware

1. Configure the SRv6 locator and disable reduced SRH to ensure classic SRH.

```

set routing-options source-packet-routing srv6 locator loc 1001::/64
set routing-options source-packet-routing srv6 no-reduced-srh

```

2. Configure the EVPN-VPWS routing instance.

```

set routing-instances vpws1 instance-type evpn-vpws
set routing-instances vpws1 interface et-0/0/20.0
set routing-instances vpws1 interface et-0/0/20.1
set routing-instances vpws1 route-distinguisher 1:7
set routing-instances vpws1 vrf-target target:1:77

```

3. Configure EVPN protocol for VLAN-aware FXC.

```

set routing-instances vpws1 protocols evpn encapsulation srv6
set routing-instances vpws1 protocols evpn flexible-cross-connect-vlan-aware

```

4. Configure PW service IDs.

```

set routing-instances vpws1 protocols evpn interface et-0/0/20.0 vpws-service-id local 666
set routing-instances vpws1 protocols evpn interface et-0/0/20.0 vpws-service-id remote 555
set routing-instances vpws1 protocols evpn interface et-0/0/20.1 vpws-service-id local 444
set routing-instances vpws1 protocols evpn interface et-0/0/20.1 vpws-service-id remote 333

```

5. Configure dynamic END.DX2V SID assignment.

```
set routing-instances vpws1 protocols evpn source-packet-routing srv6 locator loc
```

The following output shows a sample dynamic SRH FXC instance configured in VLAN-aware mode.

```
routing-options {
  source-packet-routing {
    srv6 {
      locator loc {
        prefix 1001::/64;
      }
      no-reduced-srh;
    }
  }
}
routing-instances {
  vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        interface et-0/0/20.0 {
          vpws-service-id {
            local 666;
            remote 555;
          }
        }
        interface et-0/0/20.1 {
          vpws-service-id {
            local 444;
            remote 333;
          }
        }
      }
    }
  }
}
flexible-cross-connect-vlan-aware;
+   source-packet-routing {
+     srv6 {
+       locator loc;
+     }
+   }
encapsulation srv6;
```

```

    }
  }
  interface et-0/0/20.0;
  interface et-0/0/20.1;
  route-distinguisher 1:7;
  vrf-target target:1:77;
}
}

```

Configure Static micro-SID END.DX2V for EVPN-VPWS FXC in VLAN Aware

1. Reserve the micro-SID block.

```

set routing-options source-packet-routing srv6 block usid_blk_with_statics fcbb:bb01::/32
set routing-options source-packet-routing srv6 block usid_blk_with_statics local-micro-sid
maximum-static-sids 2000

```

2. Configure the SRv6 locator from the reserved micro-SID block.

```

set routing-options source-packet-routing srv6 locator u_loc fcbb:bb01:100::/48
set routing-options source-packet-routing srv6 locator u_loc micro-sid block-name
usid_blk_with_statics

```

3. Configure the EVPN-VPWS routing instance.

```

set routing-instances vpws1 instance-type evpn-vpws
set routing-instances vpws1 interface et-0/0/20.0
set routing-instances vpws1 interface et-0/0/20.1
set routing-instances vpws1 route-distinguisher 1:7
set routing-instances vpws1 vrf-target target:1:77

```

4. Configure EVPN protocol for VLAN-aware FXC.

```

set routing-instances vpws1 protocols evpn encapsulation srv6
set routing-instances vpws1 protocols evpn flexible-cross-connect-vlan-aware

```

5. Configure PW service IDs.

```
set routing-instances vpws1 protocols evpn interface et-0/0/20.0 vpws-service-id local 666
set routing-instances vpws1 protocols evpn interface et-0/0/20.0 vpws-service-id remote 555
set routing-instances vpws1 protocols evpn interface et-0/0/20.1 vpws-service-id local 444
set routing-instances vpws1 protocols evpn interface et-0/0/20.1 vpws-service-id remote 333
```

6. Configure static micro-SID END.DX2V.

```
set routing-instances vpws1 protocols evpn source-packet-routing srv6 locator u_loc micro-
dx2v-sid 0xFF01
```

The following output shows a sample static micro-SID FXC instance configured in VLAN-aware mode.

```
routing-options {
  source-packet-routing {
    srv6 {
      block usid_blk_with_statics {
        prefix fcbb:bb01::/32;
        local-micro-sid {
          maximum-static-sids 2000;
        }
      }
      locator u_loc {
        prefix fcbb:bb01:100::/48;
        micro-sid {
          block-name usid_blk_with_statics;
        }
      }
    }
  }
}
routing-instances {
  vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        interface et-0/0/20.0 {
          vpws-service-id {
            local 666;
          }
        }
      }
    }
  }
}
```

```

        remote 555;
    }
}
interface et-0/0/20.1 {
    vpws-service-id {
        local 444;
        remote 333;
    }
}
flexible-cross-connect-vlan-aware;
+   source-packet-routing {
+       srv6 {
+           locator u_loc micro-dx2v-sid 0xFF01;
+       }
+   }
    encapsulation srv6;

}
}
interface et-0/0/20.0;
interface et-0/0/20.1;
route-distinguisher 1:7;
vrf-target target:1:77;
}
}

```

Configure Dynamic micro-SID END.DX2V for EVPN-VPWS FXC in VLAN Aware

1. Configure the SRv6 locator with micro-sid.

```

set routing-options source-packet-routing srv6 locator loc 1001::/64
set routing-options source-packet-routing srv6 locator loc micro-sid

```

2. Configure the EVPN-VPWS routing instance.

```

set routing-instances vpws1 instance-type evpn-vpws
set routing-instances vpws1 interface et-0/0/20.0
set routing-instances vpws1 interface et-0/0/20.1
set routing-instances vpws1 route-distinguisher 1:7
set routing-instances vpws1 vrf-target target:1:77

```

3. Configure EVPN protocol for VLAN-aware FXC.

```
set routing-instances vpws1 protocols evpn encapsulation srv6
set routing-instances vpws1 protocols evpn flexible-cross-connect-vlan-aware
```

4. Configure PW service IDs.

```
set routing-instances vpws1 protocols evpn interface et-0/0/20.0 vpws-service-id local 666
set routing-instances vpws1 protocols evpn interface et-0/0/20.0 vpws-service-id remote 555
set routing-instances vpws1 protocols evpn interface et-0/0/20.1 vpws-service-id local 444
set routing-instances vpws1 protocols evpn interface et-0/0/20.1 vpws-service-id remote 333
```

5. Configure dynamic micro-SID END.DX2V assignment.

```
set routing-instances vpws1 protocols evpn source-packet-routing srv6 locator loc
```

The following output shows a sample dynamic micro-SID FXC instance configured in VLAN-aware mode.

```
routing-options {
  source-packet-routing {
    srv6 {
      locator loc {
        prefix 1001::/64;
        micro-sid;
      }
    }
  }
}

routing-instances {
  vpws1 {
    instance-type evpn-vpws;
    protocols {
      evpn {
        interface et-0/0/20.0 {
          vpws-service-id {
            local 666;
            remote 555;
          }
        }
      }
    }
  }
}
```

```

        }
    }
    interface et-0/0/20.1 {
        vpws-service-id {
            local 444;
            remote 333;
        }
    }
flexible-cross-connect-vlan-aware;
+     source-packet-routing {
+         srv6 {
+             locator loc;
+         }
+     }
        encapsulation srv6;

    }
}
interface et-0/0/20.0;
interface et-0/0/20.1;
route-distinguisher 1:7;
vrf-target target:1:77;
}
}

```

RELATED DOCUMENTATION

| [Configuring EVPN over Transport Class Tunnels](#) | 89



EVPN E-Tree

- Overview | **1697**
 - Configuring EVPN E-Tree | **1701**
-

Overview

IN THIS CHAPTER

- [EVPN E-Tree Overview | 1697](#)

EVPN E-Tree Overview

IN THIS SECTION

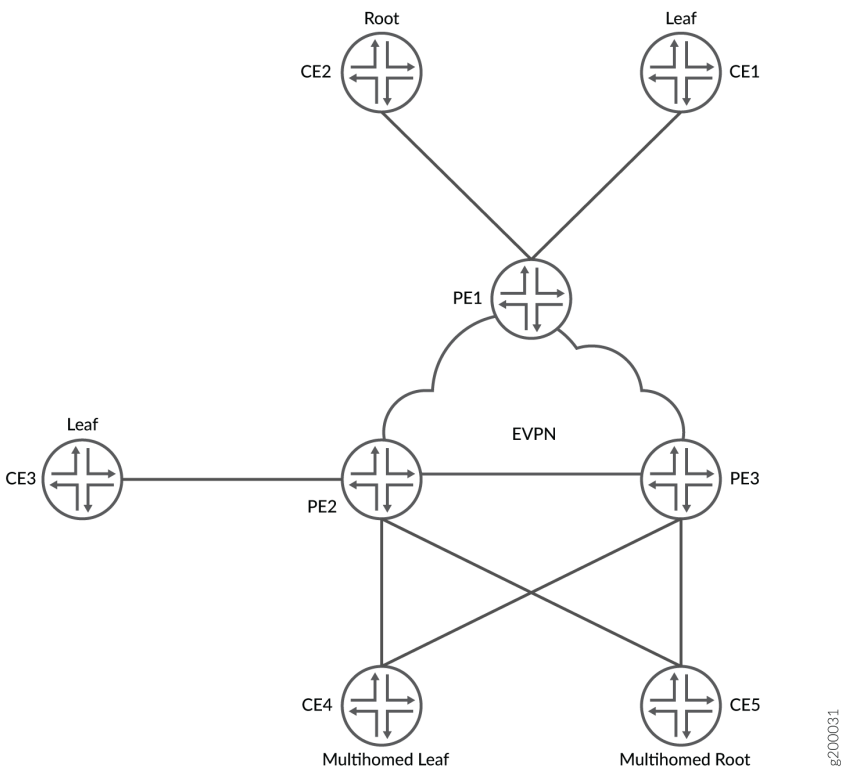
- [NSR and Unified ISSU Support for EVPN E-Tree | 1699](#)
- [Platform-Specific EVPN E-Tree Behavior | 1700](#)

You can use the Ethernet VPN (EVPN) Ethernet-Tree (E-Tree) feature to create a rooted-multipoint service supported with EVPN-MPLS in the core, as defined by the Metro Ethernet Forum (MEF) in [RFC 8317](#). In an EVPN E-Tree service, you categorize interfaces as either root or leaf interfaces in a routing instance. You also define each customer edge (CE) device as a root or a leaf device. The EVPN E-Tree service follows these forwarding rules:

- A leaf can send or receive traffic only from a root.
- A root can send traffic to another root or any leaf.

Leaf and root CE devices can be single-homed or multihomed to the provider edge (PE) devices in the network.

Figure 169: EVPN E-Tree Service



The EVPN E-Tree service has all the benefits of EVPN such as all-active multihoming and load balancing loop detection for E-Tree.

In an EVPN E-Tree service, the forwarding rule depends on the traffic source and destination for known unicast traffic or unknown unicast, broadcast, and multicast (BUM) traffic. [Table 90 on page 1698](#) shows the forwarding rules within the E-Tree service.


**NOTE:** We don't support IGMP snooping, MLD snooping, or PIM snooping multicast optimizations with EVPN E-Tree.

Table 90: EVPN E-Tree Forwarding Rule Based on Traffic Source and Destination

Type of Traffic	Allowed/Not-Allowed	Filtering Location
Known Unicast Traffic from Root to Root	Allowed	-

Known Unicast Traffic from Root to Leaf	Allowed	-
BUM Traffic from Root to Root	Allowed	-
BUM Traffic from Root to Leaf	Allowed	-
Known Unicast Traffic from Leaf to Leaf	Not Allowed	At the ingress Packet Forwarding Engine
Known Unicast Traffic from Leaf to Root	Allowed	-
BUM Traffic from Leaf to Leaf	Not Allowed	At the Egress Packet Forwarding Engine
BUM Traffic from Leaf to Root	Allowed	-

If you don't configure a leaf or root role for an interface, it will be assigned the "root" role by default. All leaf interfaces are assigned a new mesh group with no local switching set to TRUE, which:

- Enables ingress filtering for unicast traffic.
- Drops all the leaf-to-leaf traffic at the ingress leaf interface.

For BUM traffic, the egress PE does the filtering based on the root or leaf label carried in the packet.

NSR and Unified ISSU Support for EVPN E-Tree

Nonstop active routing (NSR) and graceful Routing Engine switchover (GRES) minimize traffic loss when a Routing Engine switchover occurs. When a Routing Engine fails, NSR and GRES enable a routing platform with redundant Routing Engines to switch over from a primary Routing Engine to a backup Routing Engine and continue forwarding packets. With unified in-service software upgrade (ISSU), you can upgrade your Junos OS software on your MX Series router with no disruption on the control plane, and with minimal disruption of traffic. You must enable both GRES and NSR to use unified ISSU.

To enable GRES, include the `graceful-switchover` statement at the `[edit chassis redundancy]` hierarchy level.

Junos OS mirrors essential data when NSR is enabled. For EVPN E-Tree, the local EVPN E-Tree leaf label that is advertised to other PE as part of the E-Tree extended community will be mirrored on the standby Routing Engine. For information on other mirrored data and NSR data flow, see ["NSR and Unified ISSU Support for EVPN" on page 565](#).

To enable NSR, include the `nonstop-routing` statement at the `[edit routing-options]` hierarchy level and the `commit synchronize` statement at the `[edit system]` hierarchy level.

Platform-Specific EVPN E-Tree Behavior

Use the following table to review platform-specific behaviors for your platforms.

Platform	Difference
ACX5448 Routers	When you enable EVPN E-Tree on ACX5448 routers, you must also set the system profile option <code>evpn-mh-profile</code> at the <code>[edit system packet-forwarding-options firewall-profile]</code> hierarchy level and commit the configuration. To start the new profile, run the <code>restart chassis-control</code> CLI command to restart the chassis management process (mgd). You will see a syslog warning to restart the Packet Forwarding Engine.

RELATED DOCUMENTATION

[Example: Configuring EVPN E-Tree Service](#) | 1701

Configuring EVPN E-Tree

IN THIS CHAPTER

- [Example: Configuring EVPN E-Tree Service | 1701](#)

Example: Configuring EVPN E-Tree Service

IN THIS SECTION

- [Requirements | 1701](#)
- [Overview | 1702](#)
- [Configuration | 1703](#)
- [Verification | 1712](#)

This example shows how to configure the Ethernet VPN (EVPN) Ethernet-Tree (E-Tree) service.

Requirements

This example uses the following hardware and software components:

- Three MX Series 5G Universal Routing Platforms configured as provider edge (PE) routers.
- Three customer edge (CE) routers, each connected to the PE routers.
- Junos OS Release 17.2 or later running on all the PE routers.

Before you begin:

- Configure the device interfaces.
- Configure an IGP, such as OSPF, on all the devices.

- Establish a BGP session between the PE devices.
- Configure MPLS and LDP on the PE devices.

Overview

IN THIS SECTION

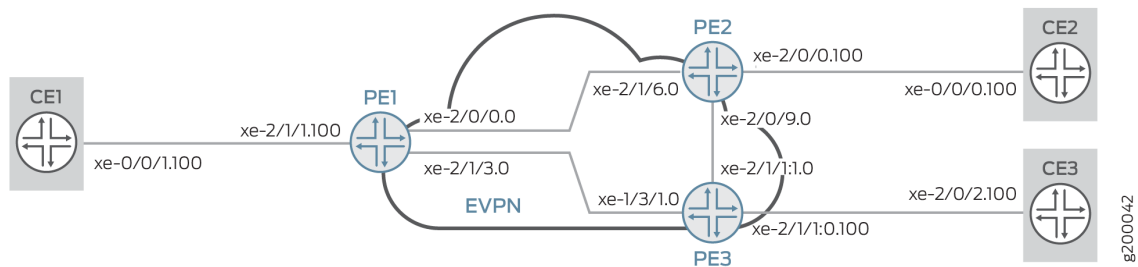
- [Platform-Specific EVPN E-Tree Behavior | 1702](#)

You can use the EVPN E-Tree feature to create a rooted-multipoint service supported with EVPN-MPLS in the core, as defined by the Metro Ethernet Forum (MEF) in [RFC 8317](#). In an EVPN E-Tree service, you categorize interfaces as either root or leaf interfaces in a routing instance. You also define each customer edge (CE) device as a root or a leaf device. The EVPN E-Tree service follows these forwarding rules:

- A leaf can send or receive traffic only from a root.
- A root can send traffic to another root or any leaf.

Leaf and root CE devices can be single-homed or multihomed to the provider edge (PE) devices in the network.

Figure 170: EVPN E-Tree Service



Platform-Specific EVPN E-Tree Behavior

Use the following table to review platform-specific behaviors for your platforms.

Platform	Difference
ACX5448 Routers	When you enable EVPN E-Tree on ACX5448 routers, you must also set the system profile option <code>evpn-mh-profile</code> at the <code>[edit system packet-forwarding-options firewall-profile]</code> hierarchy level and commit the configuration. To start the new profile, run the <code>restart chassis-control</code> CLI command to restart the chassis management process (mgd). You will see a syslog warning to restart the Packet Forwarding Engine.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1703](#)
- [Procedure | 1707](#)
- [Results | 1710](#)

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.



NOTE: When you want to change the `etree-ac-role` from leaf to root or vice versa, use the following procedure.

1. Deactivate the interface unit configuration.
2. Change the `etree-ac-role` option under the deactivated unit configuration.
3. Reactivate the updated interface unit configuration.

Deactivating the interface unit before changing the `etree-ac-role` option ensures that the system applies the role change correctly to all the E-Tree components.

CE1

```

set interfaces xe-0/0/1 vlan-tagging
>set interfaces xe-0/0/1 unit 100 vlan-id 100
>set interfaces xe-0/0/1 unit 100 family inet address 10.100.0.1/24

```

PE1

```

>set interfaces xe-2/0/0 unit 0 family inet address 10.0.0.1/30
>set interfaces xe-2/0/0 unit 0 family mpls
>set interfaces xe-2/1/3 unit 0 family inet address 10.0.0.5/30
>set interfaces xe-2/1/3 unit 0 family mpls
>set interfaces lo0 unit 0 family inet address 10.255.0.1/32 primary
>set interfaces lo0 unit 0 family inet address 10.255.0.1/32 preferred
>set interfaces xe-2/1/1 flexible-vlan-tagging
>set interfaces xe-2/1/1 encapsulation flexible-ethernet-services
>set interfaces xe-2/1/1 unit 100 encapsulation vlan-bridge
>set interfaces xe-2/1/1 unit 100 vlan-id 100
>set interfaces xe-2/1/1 unit 100 etree-ac-role root
>set routing-options router-id 10.255.0.1
>set routing-options autonomous-system 65000
>set protocols mpls interface all
>set protocols mpls interface fxp0.0 disable
>set protocols bgp group evpn local-address 10.255.0.1
>set protocols bgp group evpn family evpn signaling
>set protocols bgp group evpn peer-as 65000
>set protocols bgp group evpn local-as 65000
>set protocols bgp group evpn neighbor 10.255.0.2
>set protocols bgp group evpn neighbor 10.255.0.3
>set protocols ospf area 0.0.0.0 interface all
>set protocols ospf area 0.0.0.0 interface fxp0.0 disable
>set protocols ldp interface all
>set protocols ldp interface fxp0.0 disable
>set routing-instances evpna instance-type evpn
>set routing-instances evpna vlan-id 100
>set routing-instances evpna interface xe-2/1/1.100
>set routing-instances evpna route-distinguisher 10.255.0.1:100
>set routing-instances evpna vrf-target target:65000:100
>set routing-instances evpna protocols evpn interface xe-2/1/1.100
>set routing-instances evpna protocols evpn evpn-etree

```

PE2

```

>set interfaces xe-2/1/6 unit 0 family inet address 10.0.0.2/30
>set interfaces xe-2/1/6 unit 0 family mpls
>set interfaces xe-2/0/9 unit 0 family inet address 10.0.0.9/30
>set interfaces xe-2/0/9 unit 0 family mpls
>set interfaces lo0 unit 0 family inet address 10.255.0.2/32 primary
>set interfaces lo0 unit 0 family inet address 10.255.0.2/32 preferred
>set interfaces xe-2/0/0 flexible-vlan-tagging
>set interfaces xe-2/0/0 encapsulation flexible-ethernet-services
>set interfaces xe-2/0/0 unit 100 encapsulation vlan-bridge
>set interfaces xe-2/0/0 unit 100 vlan-id 100
>set interfaces xe-2/0/0 unit 100 etree-ac-role leaf
>set routing-options router-id 10.255.0.2
>set routing-options autonomous-system 65000
>set protocols mpls interface all
>set protocols mpls interface fxp0.0 disable
>set protocols bgp group evpn local-address 10.255.0.2
>set protocols bgp group evpn family evpn signaling
>set protocols bgp group evpn peer-as 65000
>set protocols bgp group evpn local-as 65000
>set protocols bgp group evpn neighbor 10.255.0.1
>set protocols bgp group evpn neighbor 10.255.0.3
>set protocols ospf area 0.0.0.0 interface all
>set protocols ospf area 0.0.0.0 interface fxp0.0 disable
>set protocols ldp interface all
>set protocols ldp interface fxp0.0 disable
>set routing-instances evpna instance-type evpn
>set routing-instances evpna vlan-id 100
>set routing-instances evpna interface xe-2/0/0.100
>set routing-instances evpna route-distinguisher 10.255.0.2:100
>set routing-instances evpna vrf-target target:65000:100
>set routing-instances evpna protocols evpn interface xe-2/0/0.100
>set routing-instances evpna protocols evpn evpn-etree

```

PE3

```

>set interfaces xe-1/3/1 unit 0 family inet address 10.0.0.6/30
>set interfaces xe-1/3/1 unit 0 family mpls
>set interfaces xe-2/1/1:1 unit 0 family inet address 10.0.0.10/30
>set interfaces xe-2/1/1:1 unit 0 family mpls

```

```

>set interfaces lo0 unit 0 family inet address 10.255.0.3/32 primary
>set interfaces lo0 unit 0 family inet address 10.255.0.3/32 preferred
>set interfaces xe-2/1/1:0 flexible-vlan-tagging
>set interfaces xe-2/1/1:0 encapsulation flexible-ethernet-services
>set interfaces xe-2/1/1:0 unit 100 encapsulation vlan-bridge
>set interfaces xe-2/1/1:0 unit 100 vlan-id 100
>set interfaces xe-2/1/1:0 unit 100 etree-ac-role leaf
>set routing-options router-id 10.255.0.3
>set routing-options autonomous-system 65000
>set protocols mpls interface all
>set protocols mpls interface fxp0.0 disable
>set protocols bgp group evpn local-address 10.255.0.3
>set protocols bgp group evpn family evpn signaling
>set protocols bgp group evpn peer-as 65000
>set protocols bgp group evpn local-as 65000
>set protocols bgp group evpn neighbor 10.255.0.1
>set protocols bgp group evpn neighbor 10.255.0.2
>set protocols ospf area 0.0.0.0 interface all
>set protocols ospf area 0.0.0.0 interface fxp0.0 disable
>set protocols ldp interface all
>set protocols ldp interface fxp0.0 disable
>set routing-instances evpna instance-type evpn
>set routing-instances evpna vlan-id 100
>set routing-instances evpna interface xe-2/1/1:0.100
>set routing-instances evpna route-distinguisher 10.255.0.3:100
>set routing-instances evpna vrf-target target:65000:100
>set routing-instances evpna protocols evpn interface xe-2/1/1:0.100
>set routing-instances evpna protocols evpn evpn-etree

```

CE2

```

>set interfaces xe-0/0/0 vlan-tagging
>set interfaces xe-0/0/0 unit 100 vlan-id 100
>set interfaces xe-0/0/0 unit 100 family inet address 10.100.0.2/24

```

CE3

```

>set interfaces xe-2/0/2 vlan-tagging
>set interfaces xe-2/0/2 unit 100 vlan-id 100
>set interfaces xe-2/0/2 unit 100 family inet address 10.100.0.3/24

```

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 Router PE1:



NOTE: Repeat this procedure for Routers PE2 and PE3, after modifying the appropriate interface names, addresses, and other parameters.

1. Configure Router PE1 interfaces.

```
[edit interfaces]
user@PE1#set xe-2/0/0 unit 0 family inet address 10.0.0.1/30
user@PE1#set xe-2/0/0 unit 0 family mpls
user@PE1#set xe-2/1/3 unit 0 family inet address 10.0.0.5/30
user@PE1#set xe-2/1/3 unit 0 family mpls
user@PE1#set lo0 unit 0 family inet address 10.255.0.1/32 primary
user@PE1#set lo0 unit 0 family inet address 10.255.0.1/32 preferred
user@PE1#set xe-2/1/1 flexible-vlan-tagging
user@PE1#set xe-2/1/1 encapsulation flexible-ethernet-services
user@PE1#set xe-2/1/1 unit 100 encapsulation vlan-bridge
user@PE1#set xe-2/1/1 unit 100 vlan-id 100
```

2. Assign the interface as leaf or root.

```
user@PE1#[edit interfaces]

set xe-2/1/1 unit 100 etree-ac-role root
```

3. Set the router ID and autonomous system number for Router PE1.

```
[edit routing-options]
```

```

user@PE1#set  routing-options router-id 10.255.0.1
user@PE1#set  routing-options autonomous-system 65000

```

4. Enable LDP on all the interfaces of Router PE1, excluding the management interface.

```

[edit protocols]
user@PE1# set ldp interface all
user@PE1# set ldp interface fxp0.0 disable

```

5. Assign local and neighbor addresses to the BGP group for Router PE1 to peer with Routers PE2 and PE3.

```

[edit protocols]
user@PE1#set  bgp group evpn local-address 10.255.0.1
user@PE1#set  bgp group evpn neighbor 10.255.0.2
user@PE1#set  bgp group evpn neighbor 10.255.0.3

```

6. Set up the local and peer autonomous systems.

```

user@PE1#set  protocols bgp group evpn peer-as 65000
user@PE1#set  protocols bgp group evpn local-as 65000

```

7. Include the EVPN signaling Network Layer Reachability Information (NLRI) to the bgp BGP group.

```

[edit protocols]
user@PE1#set  bgp group evpn family evpn signaling

```

8. Configure OSPF on all the interfaces of Router PE1, excluding the management interface.

```

[edit protocols]
user@PE1#set ospf area 0.0.0.0 interface all
user@PE1#set ospf area 0.0.0.0 interface fxp0.0 disable

```

9. Configure MPLS on all the interfaces of Router PE1, excluding the management interface.

```
[edit protocols]
user@PE1#set mpls interface all
user@PE1#set mpls interface fxp0.0 disable
```

10. Configure the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpna instance-type evpn
```

11. Set the VLAN identifier for the bridging domain in the evpna routing instance.

```
[edit routing-instances]
user@PE1# set evpna vlan-id 100
```

12. Configure the interface name for the evpna routing instance.

```
[edit routing-instances]
user@PE1#set evpna interface xe-2/1/1.100
```

13. Configure the route distinguisher for the evpna routing instance.

```
[edit routing-instances]
user@PE1#set evpna route-distinguisher 10.255.0.1:100
```

14. Assign the interface name that connects the PE1 site to the VPN.

```
[edit routing-instances]
user@PE1#set evpna protocols evpn interface xe-2/1/1.100
```

15. Configure Ethernet VPN E-Tree service on PE1.

```
[edit routing-instances]
user@PE1#set evpna protocols evpn evpn-etree
```

16. Configure the VPN routing and forwarding (VRF) target community for the evpna routing instance.

```
[edit routing-instances]
user@PE1#set evpna vrf-target target:65000:100
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `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
xe-2/0/0 {
  unit 0 {
    family inet {
      address 10.0.0.1/30;
    }
    family mpls;
  }
}
xe-2/1/3 {
  unit 0 {
    family inet {
      address 10.0.0.5/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 10.255.0.1/32 {
        primary;
        preferred;
      }
    }
  }
}
```

```

    }
  }
}
xe-2/1/1 {
  flexible-vlan-tagging;
  encapsulation flexible-ethernet-services;
  unit 100 {
    encapsulation vlan-bridge;
    vlan-id 100;
    etree-ac-role root;
  }
}

```

```

user@PE1 show routing-options
router-id 10.255.0.1;
autonomous-system 65000;

```

```

user@PE1 show protocols
mpls {
  interface all;
  interface fxp0.0 {
    disable;
  }
}
bgp {
  group evpn {
    local-address 10.255.0.1;
    family evpn {
      signaling;
    }
    peer-as 65000;
    local-as 65000;
    neighbor 10.255.0.2;
    neighbor 10.255.0.3;
  }
}
ospf {
  area 0.0.0.0 {
    interface all;
    interface fxp0.0 {

```



```

        disable;
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}

```

```

user@PE1 #show routing-instances
evpna {
    instance-type evpn;
    vlan-id 100;
    interface xe-2/1/1.100;
    route-distinguisher 10.255.0.1:100;
    vrf-target target:65000:100;
    protocols {
        evpn {
            interface xe-2/1/1.100;
            evpn-etree;
        }
    }
}

```

Verification

IN THIS SECTION

- [Verifying the EVPN Instance Status | 1713](#)
- [Verifying local and remote MAC property | 1714](#)
- [Verifying EVPN E-Tree Instances property | 1715](#)
- [Verifying traffic between leaf and root | 1716](#)
- [Verifying traffic flow between leaf and leaf is not allowed | 1716](#)

Confirm that the configuration is working properly.

Verifying the EVPN Instance Status

Purpose

Verify the EVPN routing instances and their status.

Action

From operational mode, run the show evpn instance extensive command.

```

user@PE1>show evpn instance extensive
Instance: __default_evpn__
  Route Distinguisher: 10.255.0.1:0
  Number of bridge domains: 0
  Number of neighbors: 0

Instance: evpna
  Route Distinguisher: 10.255.0.1:100
  VLAN ID: 100
  Per-instance MAC route label: 16
  Etree Leaf label: 20
  MAC database status
    Local Remote
  MAC advertisements:      1      1
  MAC+IP advertisements:   0      0
  Default gateway MAC advertisements: 0      0
  Number of local interfaces: 1 (1 up)
    Interface name ESI Mode Status AC-Role
    xe-2/1/1.100 00:00:00:00:00:00:00:00 single-homed Up Root
  Number of IRB interfaces: 0 (0 up)
  Number of bridge domains: 1
    VLAN Domain ID Intfs / up IRB intf Mode MAC sync IM route label SG
    sync IM core nexthop
    100 1 1 Extended Enabled 30
  Disabled
  Number of neighbors: 2
    Address MAC MAC+IP AD IM ES Leaf-label
    10.255.0.2 0 0 1 1 0 20
    10.255.0.3 1 0 1 1 0 20
  Number of ethernet segments: 0

```

Meaning

The output provides the following information:

- List of EVPN and virtual switch routing instances
- Mode of operation of each interface
- Neighbors of each routing instance
- Number of different routes received from each neighbor
- Number of Ethernet segments on each routing instance
- VLAN ID and MAC labels for each routing instance

Verifying local and remote MAC property

Purpose

Verify EVPN MAC table information.

Action

From operational mode, run the `show evpn mac-table` command.

```
user@PE1>show evpn 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 : evpn_100
```

```
Bridging domain : __evpn_100__, VLAN : 100
```

MAC address	MAC flags	Logical interface	NH Index	MAC property
00:1d:b5:a2:15:2c	DC		1048579	Leaf
64:87:88:5f:05:c0	DC		1048578	Leaf
a8:d0:e5:54:38:21	D	xe-2/1/1.100		Root

Meaning

The output provides the following information:

- List of MAC addresses learned locally and via control-plane.
- Property of MAC whether it is learned on a leaf or root interface.

Verifying EVPN E-Tree Instances property

Purpose

Verify EVPN E-Tree Instances property.

Action

From operational mode, run the `show evpn instance evpna extensive` command.

```
user@PE1>show evpn instance evpna extensive
Instance: evpna
Route Distinguisher: 10.255.0.1:100
VLAN ID: 100
Per-instance MAC route label: 16
Etree Leaf label: 20
MAC database status
MAC advertisements:
MAC+IP advertisements:
Default gateway MAC advertisements:
Local Remote
0 0
0 0
0 0
Number of local interfaces: 1 (1 up)
Interface name ESI Mode Status AC-Role
xe-2/1/1.100 00:00:00:00:00:00:00:00:00:00 single-homed Up Root
Number of IRB interfaces: 0 (0 up)
Number of bridge domains: 1
VLAN Domain ID Intfs / up IRB intf Mode MAC sync IM route label SG
sync IM core nexthop
100 1 1 Extended Enabled 30
Disabled
Number of neighbors: 2
Address MAC MAC+IP AD IM ES Leaf-label
10.255.0.2 0 0 1 1 0 20
10.255.0.3 0 0 1 1 0 20
Number of ethernet segments: 0
```

Meaning

The output provides the following information:

- List the details of specific instance “evpna”.
- Lists the interfaces associated to this routing instance and its property (leaf or root).
- Lists the bridge-domains associated to this routing instance.
- Lists the neighbors and routes received.

Verifying traffic between leaf and root

Purpose

Verifying traffic flow between leaf and root

Action

From operational mode of CE2 (leaf), ping CE1 (root) to check traffic flow.

```
user@CE2> ping 10.100.0.1

PING 10.100.0.1 (10.100.0.1): 56 data bytes
64 bytes from 10.100.0.1: icmp_seq=0 ttl=64 time=1.063 ms
64 bytes from 10.100.0.1: icmp_seq=1 ttl=64 time=1.057 ms
64 bytes from 10.100.0.1: icmp_seq=2 ttl=64 time=1.038 ms
^C
--- 10.100.0.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.038/1.053/1.063/0.011 ms
```

Meaning

The output shows Ping is successful between CE2 (leaf) and CE1 (root).

Verifying traffic flow between leaf and leaf is not allowed

Purpose

Verifying traffic flow between leaf and leaf is not allowed.

Action

From operational mode of CE2 (leaf), ping CE3 (leaf) to check traffic flow.

```
user@CE2> ping 10.100.0.1

PING 10.100.0.3 (10.100.0.3): 56 data bytes
^C
--- 10.100.0.3 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
```

Meaning

The output shows Ping failed between CE2 and CE3 because traffic is not allowed between leaf and leaf interfaces.

RELATED DOCUMENTATION

| [EVPN E-Tree Overview](#) | 1697

7

PART

Using EVPN for Interconnection

- Interconnecting VXLAN Data Centers With EVPN | **1719**
 - Interconnecting EVPN-VXLAN Data Centers Through an EVPN-MPLS WAN | **1760**
 - Extending a Junos Fusion Enterprise Using EVPN-MPLS | **1845**
-

Interconnecting VXLAN Data Centers With EVPN

IN THIS CHAPTER

- [VXLAN Data Center Interconnect Using EVPN Overview | 1719](#)
- [Example: Configuring VXLAN Data Center Interconnect Using EVPN | 1739](#)

VXLAN Data Center Interconnect Using EVPN Overview

IN THIS SECTION

- [Technology Overview of VXLAN-EVPN Integration for DCI | 1719](#)
- [Implementation Overview of VXLAN-EVPN Integration for DCI | 1731](#)
- [Active-Active Redundancy Use Case | 1736](#)
- [Supported and Unsupported Features for VXLAN DCI Using EVPN | 1737](#)

Ethernet VPN (EVPN) technology can be used to interconnect Virtual Extensible Local Area Network (VXLAN) networks over an MPLS/IP network to provide data center connectivity. This is done through Layer 2 intra-subnet connectivity and control-plane separation among the interconnected VXLAN networks.

The following sections describe the technology and implementation overview of integrating EVPN with VXLAN to be used as a data center interconnect (DCI) solution.

Technology Overview of VXLAN-EVPN Integration for DCI

The following sections provide a conceptual overview of VXLAN, EVPN, the need for their integration for DCI and the resulting benefits.

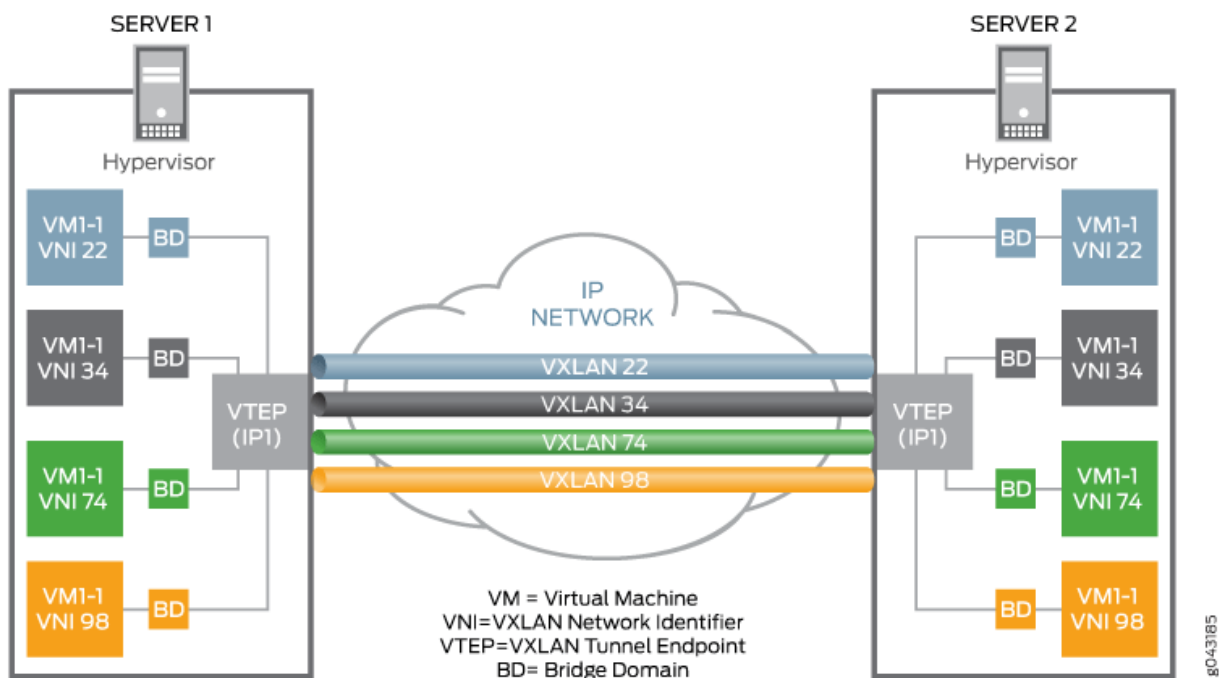
Understanding VXLAN

Virtual Extensible Local Area Network (VXLAN) is a Layer 3 encapsulation protocol that enables MX Series routers to push Layer 2 or Layer 3 packets through a VXLAN tunnel to a virtualized data center or the Internet. Communication is established between two virtual tunnel endpoints (VTEPs), which can be end hosts or network switches or routers, that encapsulate and de-encapsulate the virtual machine (VM) traffic into a VXLAN header.

VXLAN is often described as an overlay technology because it allows you to stretch Layer 2 connections over an intervening Layer 3 network by encapsulating (tunneling) Ethernet frames in a VXLAN packet that includes IP addresses. This feature of VXLAN addresses the requirements of a multi-tenant datacenter, where each tenant's VM might be sharing the physical server with other tenants that are distributed across physical servers within or across different data centers, by meeting the growing need to provide seamless Layer 2 connectivity between all the VMs owned by a tenant, in addition to isolating each tenant's traffic for security and potential MAC address overlaps.

VXLAN tunnels are created between the physical servers by the hypervisors. Since a physical server can host multiple tenants, each hypervisor creates multiple VXLAN tunnels.

Figure 171: VXLAN Overview



VXLAN is a technology that allows you to segment your networks (as VLANs do) but that also solves the scaling limitation of VLANs and provides benefits that VLANs cannot. Some of the important benefits of using VXLANs include:

- You can theoretically create as many as 16 million VXLANs in an administrative domain (as opposed to 4094 VLANs on a Juniper Networks device).

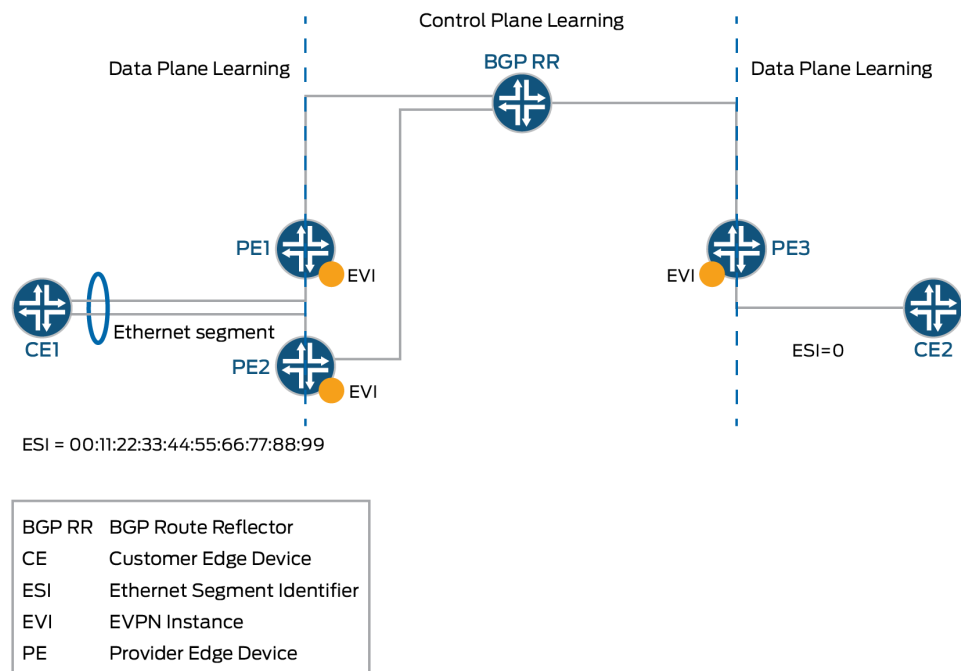
MX Series routers support as many as 32K VXLANs. This means that VXLANs provide network segmentation at the scale required by cloud builders to support very large numbers of tenants.

- You can enable migration of virtual machines between servers that exist in separate Layer 2 domains by tunneling the traffic over Layer 3 networks. This functionality allows you to dynamically allocate resources within or between data centers without being constrained by Layer 2 boundaries or being forced to create large or geographically stretched Layer 2 domains.

Understanding EVPN

EVPN is a new standards-based technology that provides virtual multi-point bridged connectivity between different Layer 2 domains over an IP or IP/MPLS backbone network. Similar to other VPN technologies, such as IPVPN and VPLS, EVPN instances (EVIs) are configured on PE routers to maintain logical service separation between customers. The PEs connect to CE devices which can be a router, switch, or host. The PE routers then exchange reachability information using Multi-Protocol BGP (MP-BGP) and encapsulated traffic is forwarded between PEs. Because elements of the architecture are common with other VPN technologies, EVPN can be seamlessly introduced and integrated into existing service environments.

Figure 172: EVPN Overview



The EVPN technology provides mechanisms for next generation Data Center Interconnect (DCI) by adding extended control plane procedures to exchange the Layer 2 (MAC address) and Layer 3 (IP address) information among the participating Data Center Border Routers (DCBRs). These features help to address some of the DCI challenges, such as seamless VM mobility and optimal IP routing. Seamless VM mobility refers to the challenge of Layer 2 extension and maintaining connectivity in the face of VM mobility, and optimal IP routing refers to the challenge of supporting default gateway behavior for a VM's outbound traffic and triangular routing avoidance of a VM's inbound traffic.

The EVPN technology is used by the data center operator to offer multi-tenancy, flexible and resilient services that can be extended on demand. This flexibility and resiliency can require using compute resources among different physical data centers for a single service (Layer 2 extension), and VM motion.

EVPN supports all-active multihoming which allows a CE device to connect to two or more PE routers such that traffic is forwarded using all of the links between the devices. This enables the CE to load balance traffic to the multiple PE routers. More importantly, it allows a remote PE to load balance traffic to the multihomed PEs across the core network. This load balancing of traffic flows between data centers is known as aliasing. EVPN also has mechanisms that prevent the looping of broadcast, unknown unicast, and multicast (BUM) traffic in an all-active multi-homed topology.

Multihoming provides redundancy in the event that an access link or one of the PE routers fails. In either case, traffic flows from the CE towards the PE use the remaining active links. For traffic in the other direction, the remote PE updates its forwarding table to send traffic to the remaining active PEs connected to the multihomed Ethernet segment. EVPN provides a fast convergence mechanism so that

the time it takes to make this adjustment is independent of the number of MAC addresses learned by the PE.

EVPN's MP-BGP control plane allows live virtual machines to be dynamically moved from one data center to another, also known as VM motion. After a VM is moved to a destination server/hypervisor it transmits a Gratuitous ARP that updates the Layer 2 forwarding table of the PE at the destination data center. The PE then transmits a MAC route update to all remote PEs, which in turn update their forwarding tables. In this manner, an EVPN tracks the movement of the VM, also known as MAC Mobility. EVPN also has mechanisms to detect and stop MAC flapping.

The EVPN technology, similar to Layer 3 MPLS VPN, is a technology that introduces the concept of routing MAC addresses using MP-BGP over MPLS core. Some of the important benefits of using EVPNs include:

- Ability to have a dual active multihomed edge device.
- Provides load balancing across dual-active links.
- Provides MAC address mobility.
- Provides multi-tenancy.
- Provides aliasing.
- Enables fast convergence.

VXLAN-EVPN Integration Overview

VXLAN defines a tunneling scheme to overlay Layer 2 networks on top of Layer 3 networks. It allows for optimal forwarding of Ethernet frames with support for multipathing of unicast and multicast traffic with the use of UDP/IP encapsulation for tunneling, and is mainly used for intra-datacenter site connectivity.

On the other hand, a unique characteristic of EVPN is that MAC address learning between PE devices occurs in the control plane. A new MAC address detected from a CE device is advertised by the local PE, using MP-BGP, to all the remote PE devices. This method differs from existing Layer 2 VPN solutions such as VPLS, which learn by flooding unknown unicast in the data plane. This control plane-based MAC learning method is the key enabler of the many useful features provided by EVPN.

Because MAC learning is handled in the control plane, this leaves EVPN with the flexibility to support different data plane encapsulation technologies between PEs. This is important because not every backbone network may be running MPLS, especially in Enterprise networks.

There is a lot of interest in EVPN today because it addresses many of the challenges faced by network operators that are building data centers to offer cloud and virtualization services. The main application of EVPN is Data Center Interconnect (DCI), the ability to extend Layer 2 connectivity between different data centers that are deployed to improve the performance of delivering application traffic to end users and for disaster recovery.

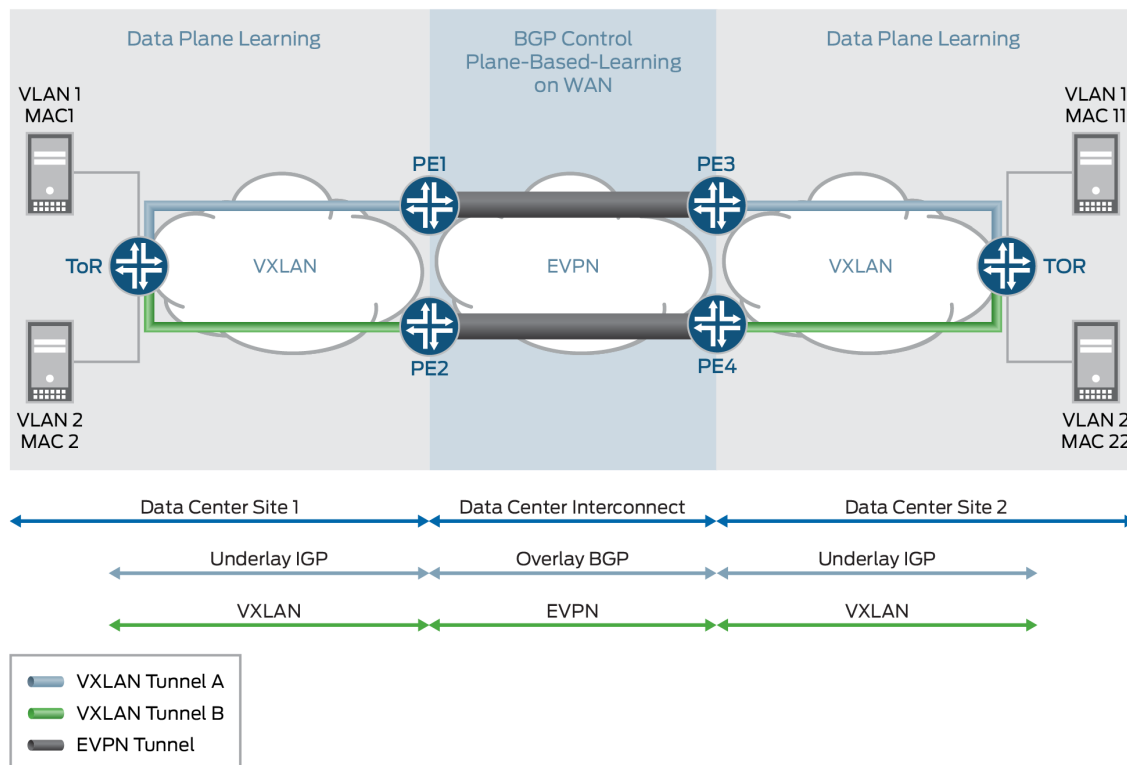
Although there are various DCI technologies available, EVPN has an added advantage over the other MPLS technologies because of its unique features, such as active-active redundancy, aliasing, and mass MAC withdrawal. As a result, to provide a solution for DCI, VXLAN is integrated with EVPN.

Every VXLAN network, which is connected to the MPLS or IP core, runs an independent instance of the IGP control plane. Each PE device participates in the IGP control plane instance of its VXLAN network. Here, each customer is a datacenter so it has its own virtual router for VXLAN underlay.

Each PE node may terminate the VXLAN data plane encapsulation where each VNI or VSID is mapped to a bridge domain. The PE router performs data plane learning on the traffic received from the VXLAN network.

Each PE node implements EVPN to distribute the client MAC addresses learnt over the VXLAN tunnel into BGP. Each PE node encapsulates the VXLAN or Ethernet frames with MPLS when sending the packets over the MPLS core and with the VXLAN tunnel header when sending the packets over the VXLAN network

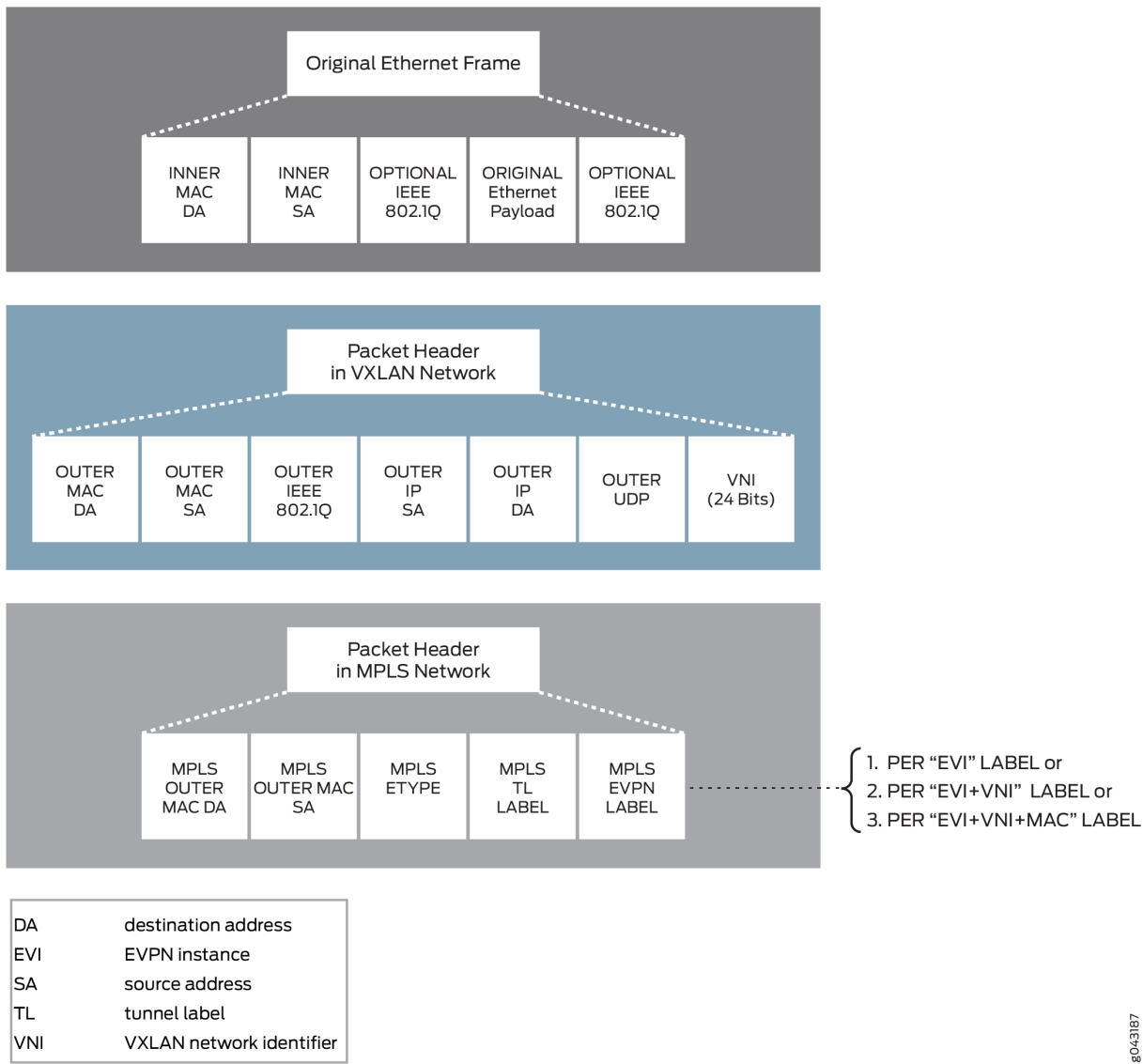
Figure 173: VXLAN-EVPN Integration Overview



VXLAN-EVPN Packet Format

The VXLAN and EVPN packet format is as follows:

Figure 174: VXLAN-EVPN Packet Format



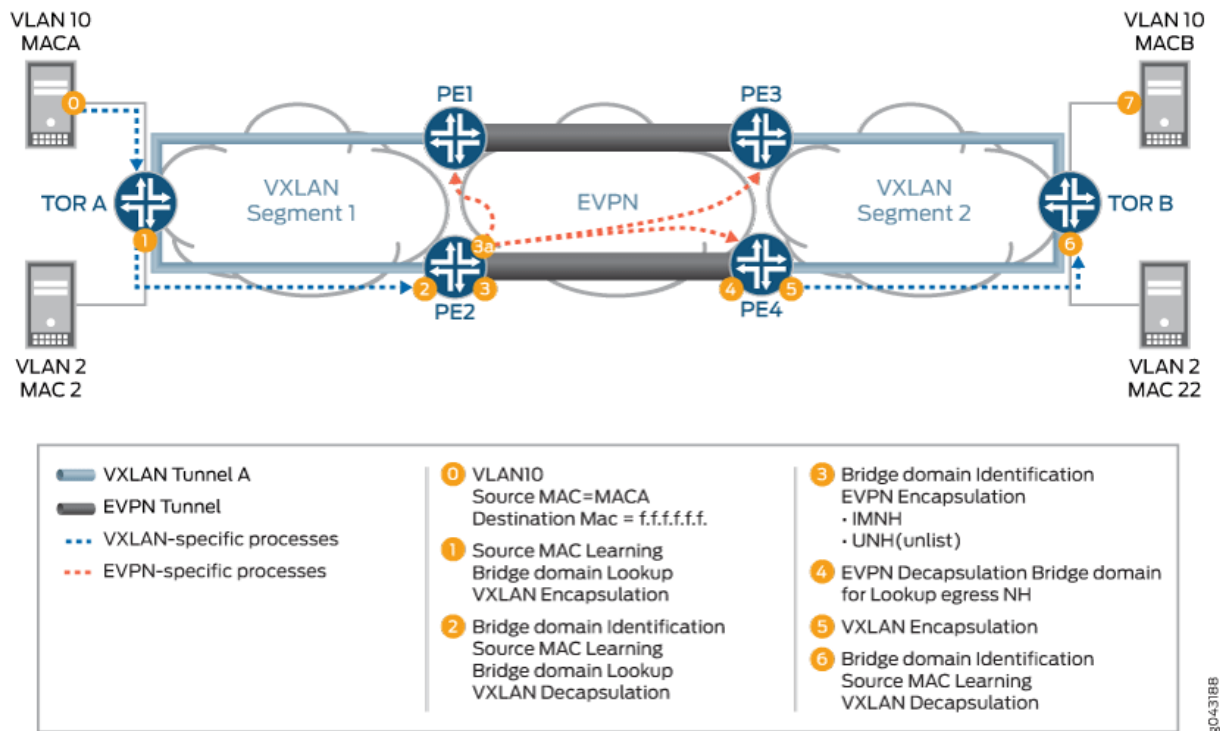
8043187

VXLAN-EVPN Packet Walkthrough

The following sections describe the packet walkthrough for two types of traffic between the VXLAN and EVPN networks:

BUM Traffic Handling

Figure 175: VXLAN-EVPN BUM Traffic Handling



The VXLAN to EVPN BUM traffic from VXLAN segment1 to VXLAN segment2 over the EVPN cloud is handled as follows:

- 0**—On boot up, Server A wants to send traffic to Server B. Because Server A does not have an ARP binding for Server B in its ARP table, Server A builds an ARP broadcast request and sends it.

The contents of the ARP packets are as follows:

- VLAN ID = VLAN 10
- Source MAC= MAC1
- Destination MAC = ff.ff.ff.ff.ff
- Source IP address = IP address of Server A or VM IP address
- Destination IP address = IP address of Server B
- Ether type of packet = 0x0806

A Layer 2 frame is sent to top-of-rack (TOR) switch TOR A, which is VXLAN enabled.

2. **1**—The ARP request (broadcast) frame is received by switch TOR A. TOR A is the originator and terminator of the VXLAN VTEP for VNI 1000. The VTEP for VXLAN 1000 is part of the broadcast domain for Server A VLAN 10.

After receiving the frame, TOR A performs ingress processing, including ingress packet classification. Based on the incoming VLAN in the packet, TOR A classifies the packet into one of the IFL under a given port. The family of this IFL is a bridge family. Based on the IFL bridge family, the bridge domain ID is identified.

After the bridge domain is identified, TOR A learns the incoming frame source MAC so that MAC A becomes reachable through this IFL. Because the frame is a broadcast frame, TOR A needs to send the frame to all the members of the broadcast domain (other than the member on which the frame was received). One of the members of the broadcast domain is the VTEP for VNI 1000. To send the frame on the VXLAN segment, TOR A completes VXLAN BUM next hop processing on the frame. The next hop pushes the VXLAN header.

The contents of the VXLAN header is as follows:

- Source MAC Address = MAC Address or source IP address interface
- Destination MAC address = Multicast MAC address
- Source IP address = 10.10.10.1
- Destination IP Address = Multicast group address (226.0.39.16)
- Source UDP port = Calculated based on the hash on the incoming frame header
- Destination UDP port = 4789 (well known port for VXLAN tunnel)

After building the VXLAN encapsulated frame, TOR A sends the frame to Router PE2.

3. **2**—Router PE2 receives the VXLAN frame and identifies the frame as a VXLAN frame by looking at the well-known destination UDP port. This VXLAN frame's VNI ID is used for bridge domain identification. After router PE2 identifies the bridge domain, PE2 completes MAC learning for the inner source MAC to the outer source IP address (MACA to 10.10.10.1 mapping). After the mapping is done, the VXLAN decapsulation next hop processing removes the VXLAN header to terminate the VXLAN tunnel.
4. **3A**—After MAC learning is done, the learnt source MAC (MAC1 to outer source IP) is sent to the L2ALD. This MAC route is sent by L2ALD to RPD for control plane learning of this MAC through BGP MAC route advertisement to BGP peers. After the BGP peer routers receive the MAC route advertisement, the routers install this MAC reachability (MACA, MPLS LABEL L1) in the bridge-domain table.

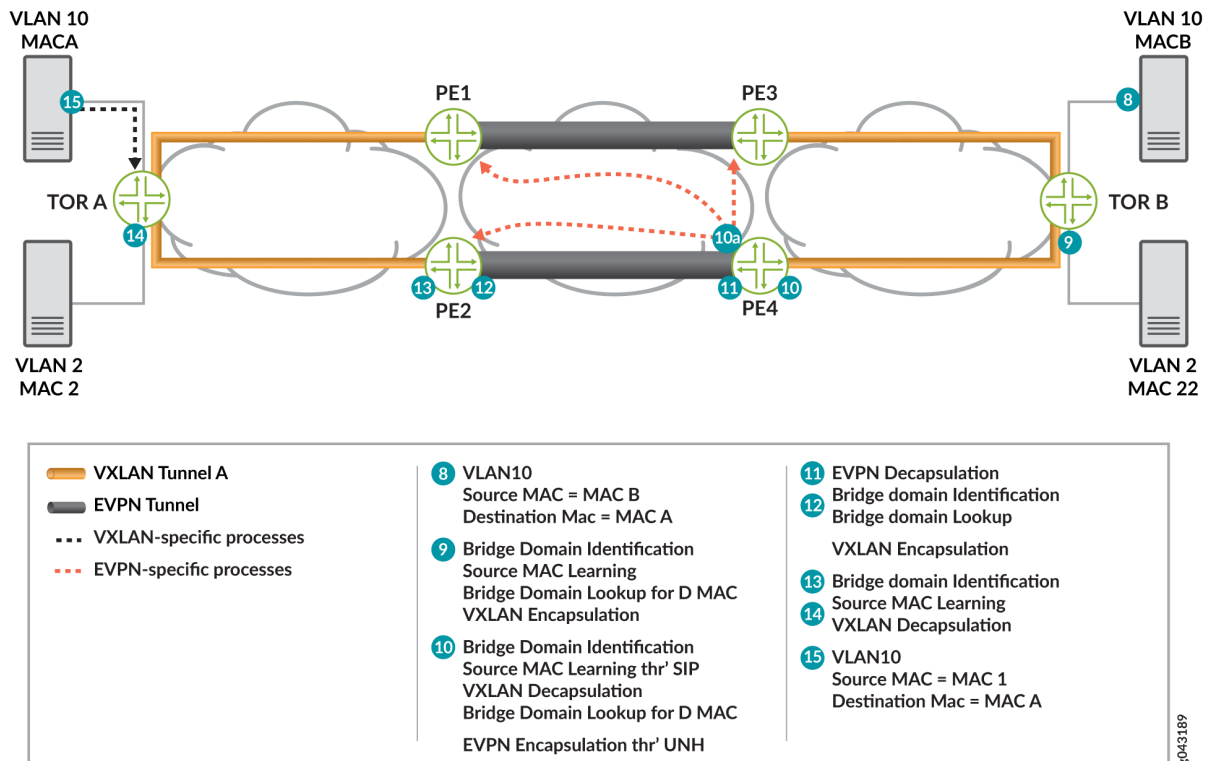
5. 3—The given bridge domain points to the multicast next hop route for forwarding the packet over the EVPN cloud. This next hop pushes the service label (multicast MPLS label associated with VNI per peer ID, bridge domain, label is the per peer ID and VNI ID). The MPLS packet is formed and sent over the MPLS cloud.
6. 4—Router PE4 receives the frame as an MPLS packet. Here, PE4 identifies the bridge domain by looking up the MPLS label L1 in the mpls.0 table. MPLS lookup points to the table next hop for the bridge domain next hop. After the bridge domain is identified and the packet is identified as a broadcast packet, the BUM composite flood next hop is executed. The BUM composite next hop also points to the VXLAN next hop (which is used for building the VXLAN multicast packet).
7. 5—The VXLAN next hop contains information for building the VXLAN header.

The VXLAN header information is as follows:

- Source MAC Address = MAC Address or the source IP address interface
 - Destination MAC address = Multicast MAC Address
 - Source IP address = 11.10.10.1
 - Destination IP Address = Multicast group address (226.0.39.16)
 - Source UDP port = Calculated based on the hash on the incoming frame header
 - Destination UDP port = 4789 (well known port for the VXLAN tunnel)
8. 6—Frame handling for this step is same as Step 1. After the VXLAN header is removed, the frame is forwarded to the CE flood route associated with the broadcast domain, and the packet is forwarded as a Layer 2 frame.
 9. 7—Server B receives an ARP request packet and sends an ARP reply to Server A.

Unicast Traffic Handling

Figure 176: VXLAN-EVPN Unicast Traffic Handling



Assuming that both data and control plane MAC learning has already happened, the VXLAN to EVPN unicast traffic (ARP reply) from Server B is handled as follows:

1. 8—Server B generates an ARP reply.

The contents of the ARP packets are as follows:

- VLAN ID = VLAN 10
- Source MAC = MACB (Server B interface MAC)
- Destination MAC = MACA
- Source IP address = IP address of Server B or VM IP address
- Destination IP address = IP address of Server A

The ARP packet is forwarded to switch TOR B.

2. **9**—After receiving the frame, switch TOR B classifies the incoming frame. The frame is classified in an IFL on the received interface. Based on the IFL family, the bridge domain associated with IFL is identified. On the given bridge domain, TOR B learns the source MAC address. Once TOR B completes the bridge domain destination MAC (MACA) look up, this look up provides the VXLAN unicast next hop. This next hop contains all the information needed to form the VXLAN header.

The content of the next hop that is required to form the packet is as follows:

- Source MAC Address = MAC Address of source IP address interface
- Destination MAC address = MAC address of the next hop
- Source IP address = 11.10.10.2
- Destination IP Address = 11.10.10.1 (as a result of the MAC learning process)
- Source UDP port = Calculated based on the hash on the incoming frame header
- Destination UDP port = 4789 (well known port for the VXLAN tunnel)



NOTE: An earlier version of the VXLAN draft used 8472 as the UDP port.

3. **10**— Router PE receives the VXLAN encapsulated frame⁴. PE4 identifies the frame by completing the lookup using the destination IP address and the destination UDP port. This lookup results in the VXLAN decapsulation. The decapsulation next hop also stores the outer source IP address.

The next lookup is done based on the VNI ID 1000. This lookup results into the bridge domain table.

4. **10A**—Router PE completes the source MAC to source IP address learning and L2ALD receives the MAC learning notification. This MAC is sent to RPD for distribution to other PE routers through BGP-EVPN MAC advertisement route. The BGP control plane distributes this MAC reachability information to all other PE routers.

The destination MAC (MAC1) lookup is done in the bridge domain MAC address table. This lookup results into a unicast next hop (EVPN NH).

5. **11**—The EVPN unicast next hop is executed. This next hop contains an unicast MPLS service label. This label is distributed through the MP-BGP control plane. The downstream peer allocates this MPLS service label. Allocation of this label can be per PE (PE, VLAN) or per MAC address basis. Based on the information in the next hop, the MPLS packet is formed and forwarded on the MPLS network.
6. **12**—Router PE2 receives the frame. The frame is identified as an MPLS packet. An MPLS label lookup is done in the MPLS.0 table. This lookup results in the table next hop and in the bridge domain table.

The destination MAC (MAC1) lookup is done in the bridge domain MAC table. This lookup results in a VXLAN unicast next hop.

7. 13—The VXLAN unicast next hop contains all the information for building the VXLAN encapsulated header. The VXLAN header is imposed on the packet.

The contents of the VXLAN encapsulation next hop header are as follows:

- Source MAC Address = MAC Address of source IP address interface
- Destination MAC address = MAC address of the next hop
- Source IP address = 10.10.10.2
- Destination IP Address = 10.10.10.1 (as a result of the MAC learning process)
- Source UDP port = Calculated based on the hash on the incoming frame header
- Destination UDP port = 4789 (well known port for the VXLAN tunnel)

8. 14—The VXLAN encapsulated frame is received by switch TOR A. TOR A identifies the frame by doing the lookup using the destination IP address and the destination UDP port. This lookup results in the VXLAN decapsulation. The de-encapsulated next hop also stores the outer source IP address.

The next lookup is done based on the VNI ID 1000. This lookup results into the bridge domain table. TOR A completes the source MAC (MAC2) to source IP address (10.10.10.2) learning. TOR A looks up the destination MAC (MAC1) in the bridge domain MAC address table. This lookup results into a unicast next hop that has the information about the egress interface.

9. 15—Server A receives the ARP reply, and Server A and Server B are ready to communicate.

Implementation Overview of VXLAN-EVPN Integration for DCI

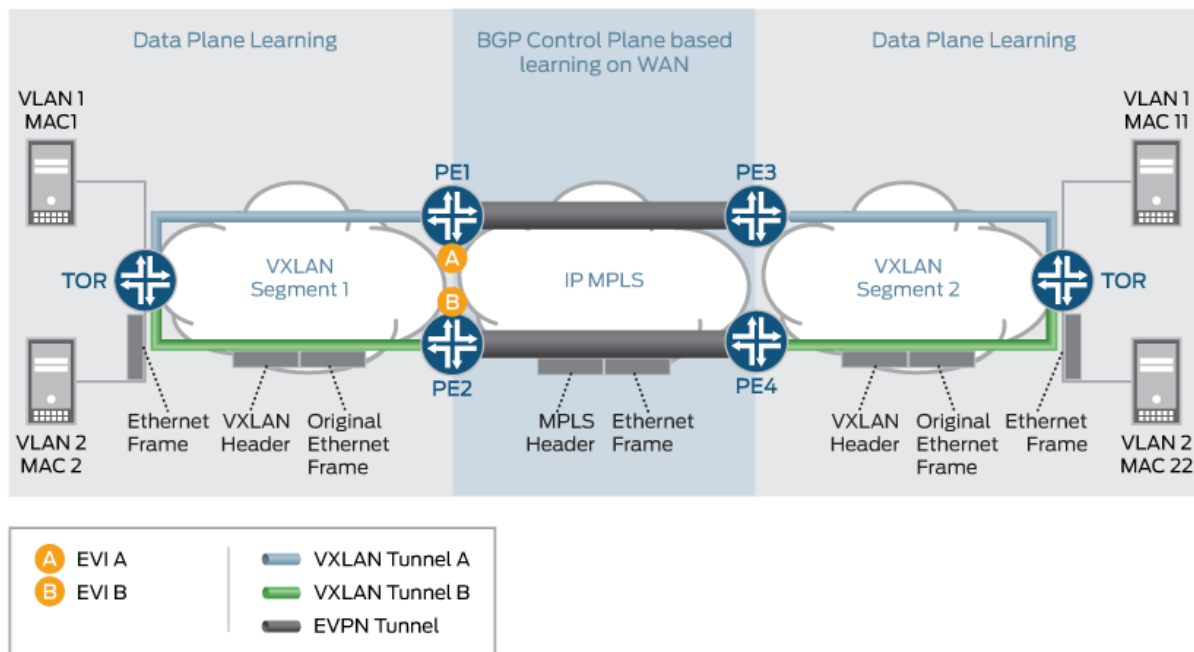
The following sections provide use case scenarios for VXLAN-EVPN integration for DCI.

VNI Base Service Use Case

In the case of the VNI base service, there is one-to-one mapping between a VNI and an EVI. In this case, there is no need to carry the VNI in the MAC advertisement route because the bridge domain ID can be derived from the route-target (RT) associated with this route. The MPLS label allocation is done per EVI basis.

[Figure 177 on page 1732](#) provides an overview for VNI base use case scenarios. The VNI base service is used most commonly for achieving the VNI translation and VNI to VLAN interworking.

Figure 177: VNI Base Service

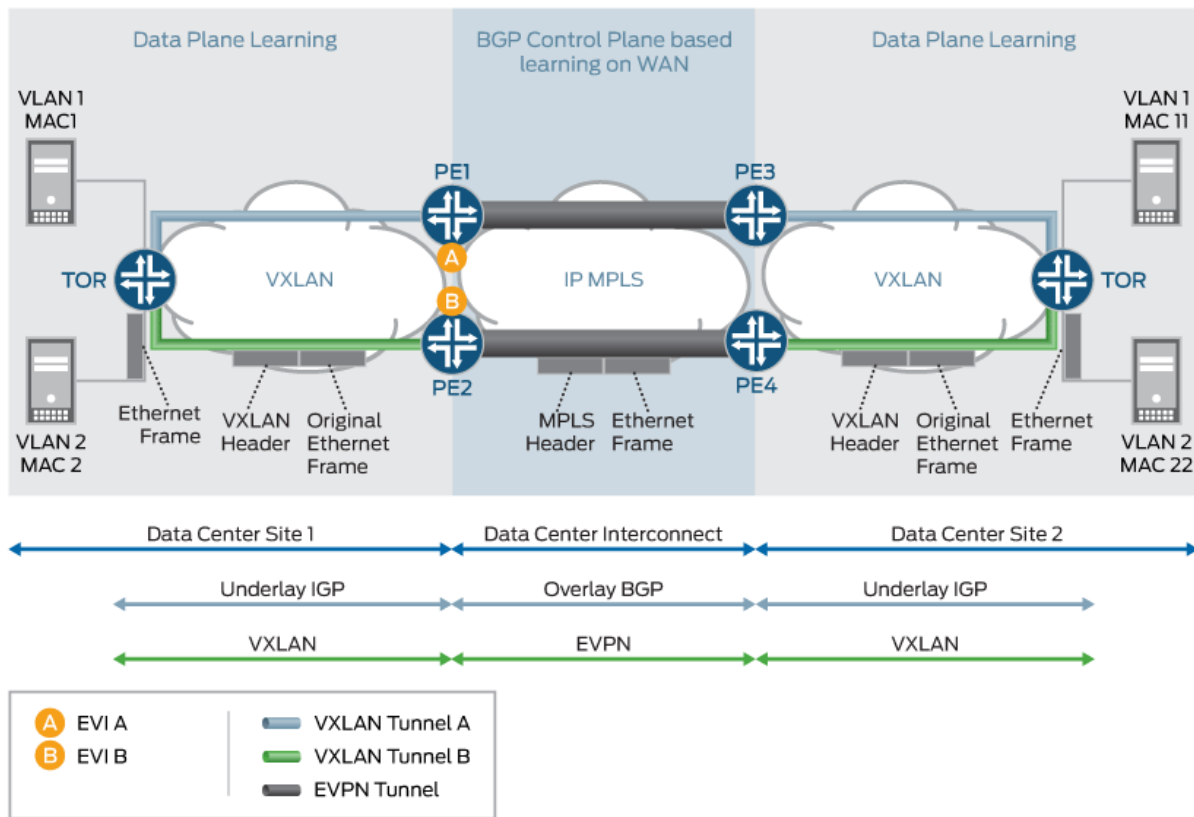


VNI Aware Service Use Case

In the case of VNI aware bundle mode, there are multiple VNIs that can be mapped to the same EVI. The Ethernet tag ID must be set to the VNI ID in the BGP routes advertisements. The MPLS label allocation in this use case should be done per EVI, VNI basis so that the VXLAN can be terminated at the ingress PE router and recreated at the egress PE router.

[Figure 178 on page 1733](#) provides details about the VNI aware service used case.

Figure 178: VNI Aware Service

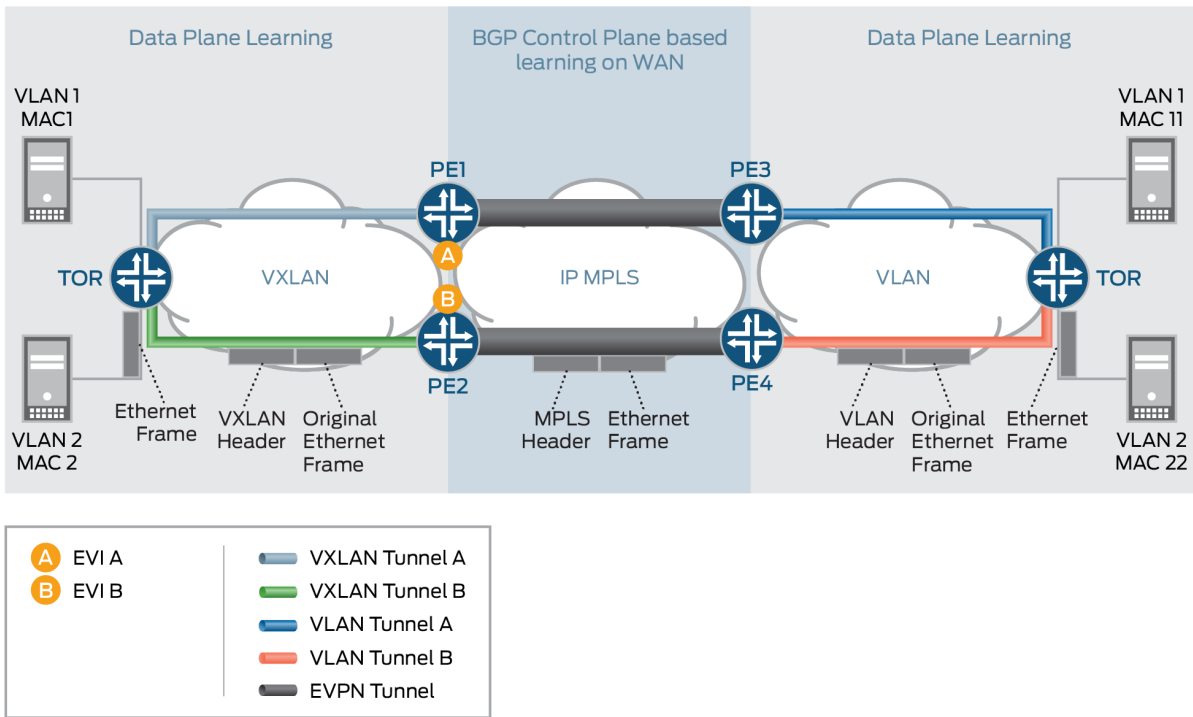


VXLAN-VLAN Interworking Use Case

This use case scenario is required for heterogeneous datacenter sites. In this scenario, the new data center site is a VXLAN-based datacenter site, and the old datacenter sites are based on VLAN. In this scenario, there is a need to have VXLAN interworking with VLAN over EVPN.

Figure 179 on page 1734 provides the detail packet walkthrough for the VXLAN-VLAN interworking use case scenario. There is a need to do the VLAN to VXLAN interworking and vice-versa from the control plane BGP route updates perspective. The label allocation needs to be done on a per EVI-basis.

Figure 179: VXLAN-VLAN Interworking



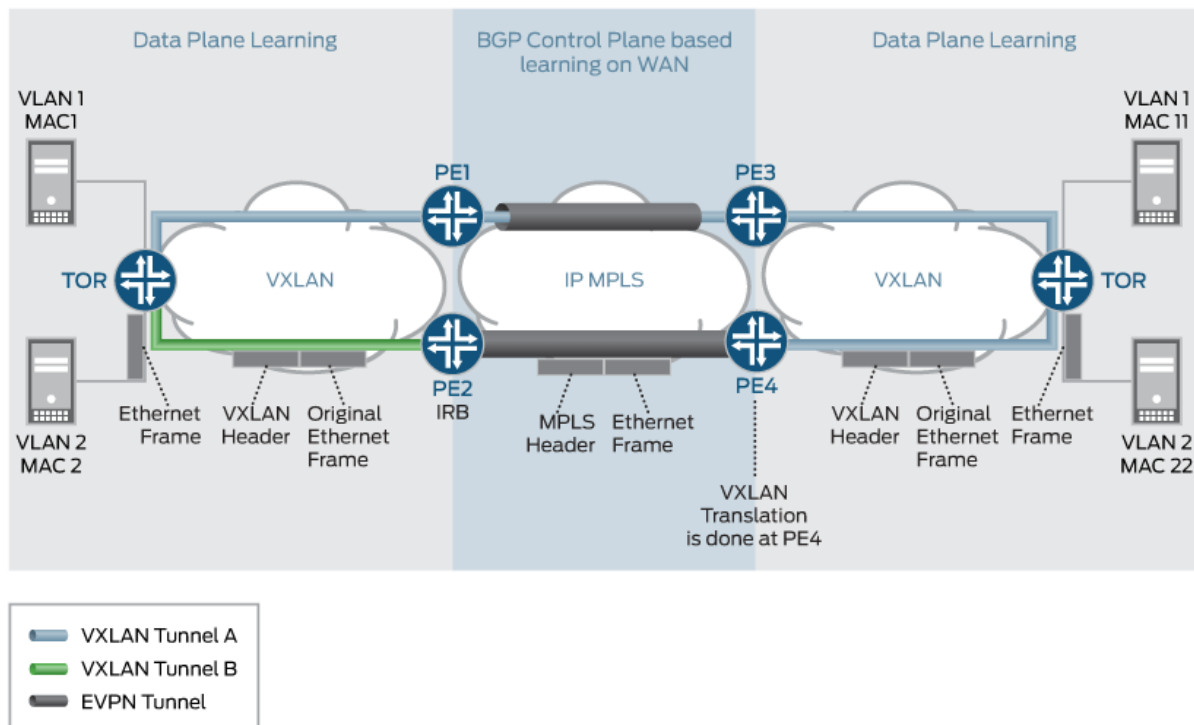
8043194

Inter VXLAN Routing Use Case

In this use case, a VM or host in one subnet (VNI-A) wants to send traffic to a VM or host in a different subnet (VNI-B). In order to provide this communication, the inter VXLAN routing should be supported.

[Figure 180 on page 1735](#) provides the use case scenarios for the inter VXLAN routing use case.

Figure 180: Inter-VXLAN Routing



Redundancy Use Case

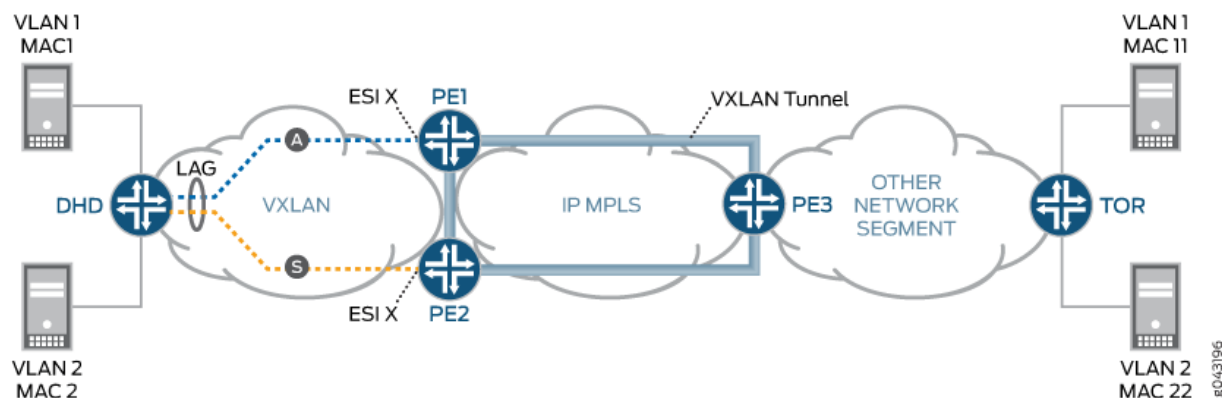
The two types of redundancy use case scenarios include Active-Standby and Active-Active.

Active-Standby Redundancy Use Case

In this use case scenario, the TOR switch (VXLAN originating GW), or the VXLAN network originating the VXLAN tunnel, is dual homed to two PE devices for active-standby redundancy. If the active link or node fails, a backup path takes over.

[Figure 181 on page 1736](#) provides details of the active-standby redundancy use case scenario.

Figure 181: Active-Standby Redundancy



Active-Active Redundancy Use Case

When interconnecting EVPN VXLAN in a data center to EVPN-VXLAN in a WAN using a gateway model on QFX series platforms, you can configure Active-Active redundancy mode on multihomed customer edge devices to allow Layer 2 unicast traffic to be load-balanced across all the multihomed links on and toward the CE device.

Setting the `interconnect-multihoming-peer-gateway` CLI command is required for MAC-VRF and VTEP-Scaling configurations. Note that in some cases, EVPN-VXLAN is supported only in VTEP Scaling mode, where a single VTEP is created for a given peer device that may have multiple routing instances. In this case, it is only possible to have a peer device be represented as a WAN peer (WAN VTEP) or a DC VTEP (a normal VTEP).

For Active-Active redundancy, additional configurations are required in the “interconnect” stanza to enable DCI interconnection. For a default switch (switch-options) configuration, be sure to set the DCI under `global protocols evpn`.

Protocols EVPN Example:

```
evpn-vxlan-dc1
  vtep-source-interface lo0.0;
  instance-type mac-vrf;
  route-distinguisher 101:1;           //DC-RD
  vrf-target target:1:1;              //DC-RT
  protocols {
    evpn {
      encapsulation vxlan;
      extended-vni-list all;
```

```

interconnect {
    vrf-target target:2:2; // WAN RT
    vrf-import <>
    route-distinguisher 101:2; // WAN RD
    interconnected-vni-list all;
    esi {
        00:00:01:02:03:04:05:06:07:08;
        all-active;
    }
}

}

}

}

vlangs {
    bd1 {
        vlan-id 51;
        l3-interface irb.0;
        vxlan {
            vni 51;
            translation-vni <>
        }
    }
}

}

}

global
    protocols {
        evpn {
            interconnect-multihoming-peer-gateways <GW>
        }
    }
}

```

Note: interconnect-multihoming-peer-gateways should be configured to contain a list of all DCI peers on the same DC.

The list can contain up to 64 peer gateway entries. Be sure to configure under the global protocol evpn stanza and not under any mac-vrf setting.

[Example: Active-Active Multihoming](#) provides details for active-active redundancy.

Supported and Unsupported Features for VXLAN DCI Using EVPN

Junos OS supports the following features for VXLAN DCI using EVPN:

- One-to-one mapping of VXLAN tunnel and an EVPN instance. In other words, one-to-one mapping between a VNI and an EVI.
- Many-to-one mapping of VXLAN tunnels over one EVPN instance, where multiple VNIs can be mapped to the same EVI.
- VNI Translation.



NOTE: VNI translation is supported by normalizing a VXLAN tag into a VLAN.

- VXLAN-to-VLAN interworking.
- Inter-VXLAN routing.
- Single active redundancy.
- Active-active redundancy in the PIM BIDIR mode.
- VXLAN tunnel traffic protection using IPSec.
- Graceful Routing Engine switchover.
- ISSU.

Junos OS does not support the following functionality for VXLAN DCI using EVPN:

- VXLAN uses IANA assigned UDP port 4789. Packets destined to the UDP port 4789 are processed only when the VXLAN configuration is enabled. The VXLAN packets are de-encapsulated by the forwarding plane and an inner Layer 2 packet is processed. MAC learnt packets are generated for the control plane processing for newly learnt MAC entries. These entries are throttled using existing infrastructure for MAC learning. VXLAN generates addition learn messages for the remote endpoints. These messages are also throttled using the existing infrastructure for denial of service detection.
- Packets received on the VXLAN tunnel are processed only if the VXLAN identifier in the packet is a known entity to the device. Unknown entities are discarded by the forwarding plane.
- Using configurable firewall filters can be discarded before it reaches the VXLAN processing module in the forwarding plane of the MX series routers.
- Logical systems.

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, Ethernet VPN (EVPN) technology can be used to interconnect Virtual Extensible Local Area Network (VXLAN) networks over an MPLS/IP network to provide data center connectivity.

Example: Configuring VXLAN Data Center Interconnect Using EVPN

IN THIS SECTION

- [Requirements | 1739](#)
- [Overview | 1740](#)
- [Configuration | 1741](#)
- [Verification | 1753](#)

This example shows how to configure Virtual Extensible Local Area Network (VXLAN) data center connectivity using Ethernet VPN (EVPN) to leverage the benefits of EVPN as a data center interconnect (DCI) solution.

Requirements

This example uses the following hardware and software components:

- Two provider edge (PE) devices in different data centers (DCs) acting as VXLAN tunnel endpoints (VTEPs).
- Two customer edge (CE) devices.
- Four host devices connected to each PE and CE device.

Before you begin:

- Configure the device interfaces.
- Configure an IGP, such as OSPF, on all the devices.

- Establish a BGP session between the PE devices.
- Configure MPLS and RSVP on the PE devices.
- Configure PIM on the CE devices and in the routing instance of the PE devices.

Overview

IN THIS SECTION

- [Topology | 1741](#)

VXLAN is a technology that provides intra data center connectivity using a tunneling scheme to stretch Layer 2 connections over an intervening Layer 3 network.

The Ethernet VPN (EVPN) technology, on the other hand, provides a solution for multipoint Layer 2 VPN services with advanced multihoming capabilities, using BGP control plane over MPLS/IP network.

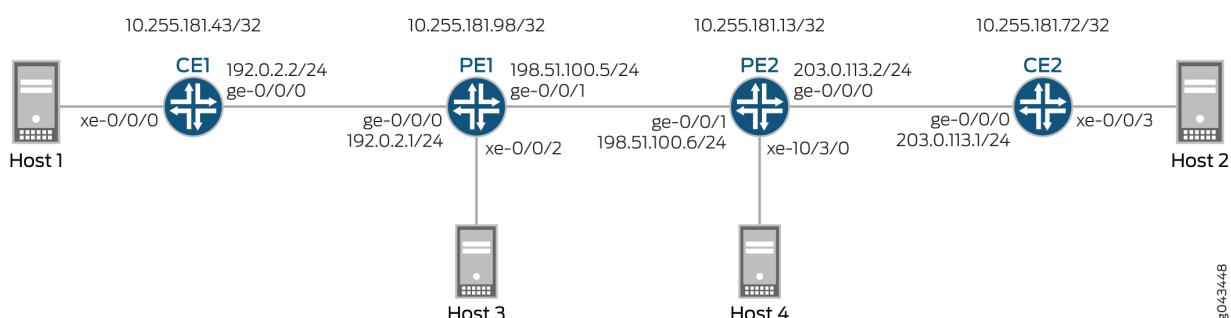
Although several solutions are available for data center connectivity, the integration of EVPN with VXLAN provides an added advantage over the existing MPLS data center interconnect (DCI) technologies.

EVPN provides mechanisms for next generation DCI by adding extended control plane procedures to exchange Layer 2 MAC address and Layer 3 IP address information among the participating Data Center Border Routers (DCBRs). EVPN with its advanced features like active-active redundancy, aliasing, and mass MAC withdrawal helps in addressing the DCI challenges, such as seamless VM mobility and optimal IP routing, thus making it essential to provide VXLAN solutions over EVPN.

[Figure 182 on page 1741](#) illustrates VXLAN data center interconnect using EVPN between devices PE1 and PE2 that are located in different data centers (DC1 and DC2, respectively). Each PE device is connected to one CE device and one host. All the PE and CE devices are configured under VLAN 10, and with the same VXLAN Network Identifier (VNI) of 10. Devices CE1 and PE1 belong to the multicast group of 192.168.1.10, and devices CE2 and PE2 belong to the multicast group of 172.16.1.10.

Topology

Figure 182: VXLAN Data Center Interconnect Using EVPN



Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1741](#)
- [Procedure | 1745](#)
- [Results | 1749](#)

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.

CE1

```
set interfaces xe-0/0/0 vlan-tagging
set interfaces xe-0/0/0 encapsulation flexible-ethernet-services
set interfaces xe-0/0/0 unit 10 encapsulation vlan-bridge
set interfaces xe-0/0/0 unit 10 vlan-id 10
set interfaces ge-0/0/0 unit 0 family inet address 192.0.2.2/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.181.43/32
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
```

```

set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols pim rp local address 10.255.181.43
set protocols pim interface all
set bridge-domains evpn10 vlan-id 10
set bridge-domains evpn10 interface xe-0/0/0.10
set bridge-domains vxlan vni 10
set bridge-domains vxlan multicast-group 172.16.1.10
set bridge-domains vxlan encapsulate-inner-vlan
set bridge-domains vxlan decapsulate-accept-inner-vlan

```

CE2

```

set interfaces xe-0/0/3 vlan-tagging
set interfaces xe-0/0/3 encapsulation flexible-ethernet-services
set interfaces xe-0/0/3 unit 10 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 10 vlan-id 10
set interfaces lo0 unit 0 family inet address 10.255.181.72/32
set protocols ospf area 0 interface ge-0/0/0.0
set protocols ospf area 0 interface lo0.0 passive
set protocols ospf area 0 interface fxp0.0 disable
set protocols pim rp local address 10.255.181.72
set protocols pim interface all
set bridge-domains evpn10 vlan-id 10
set bridge-domains evpn10 interface xe-0/0/3.10
set bridge-domains vxlan vni 10
set bridge-domains vxlan multicast-group 192.168.1.10
set bridge-domains vxlan encapsulate-inner-vlan
set bridge-domains vxlan decapsulate-accept-inner-vlan

```

PE1

```

set interfaces xe-0/0/2 vlan-tagging
set interfaces xe-0/0/2 encapsulation flexible-ethernet-services
set interfaces xe-0/0/2 unit 10 encapsulation vlan-bridge
set interfaces xe-0/0/2 unit 10 vlan-id 10
set interfaces ge-0/0/0 unit 0 family inet address 192.0.2.1/24
set interfaces ge-0/0/1 mtu 1600
set interfaces ge-0/0/1 unit 0 family inet address 198.51.100.5/24
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces lo0 unit 1 family inet address 10.255.181.98/32

```

```

set protocols rsvp interface all
set protocols mpls no-cspf
set protocols mpls label-switched-path to-PE2 to 10.255.181.13
set protocols mpls interface all
set protocols bgp family evpn signaling
set protocols bgp group ibgp type internal
set protocols bgp group ibgp neighbor 10.255.181.13 local-address 10.255.181.98
set protocols ospf area 0 interface ge-0/0/1.0
set protocols ospf area 0 interface fxp0.0 disable
set protocols ospf area 0 interface lo0.0 passive
set routing-instances evpn10 vtep-source-interface lo0.0
set routing-instances evpn10 instance-type evpn
set routing-instances evpn10 vlan-id 10
set routing-instances evpn10 interface xe-0/0/2.10
set routing-instances evpn10 vxlan vni 10
set routing-instances evpn10 vxlan multicast-group 172.16.1.10
set routing-instances evpn10 vxlan encapsulate-inner-vlan
set routing-instances evpn10 vxlan decapsulate-accept-inner-vlan
set routing-instances evpn10 route-distinguisher 10.255.181.98:10
set routing-instances evpn10 vrf-target target:10:10
set routing-instances evpn10 protocols evpn
set routing-instances evpna instance-type vrf
set routing-instances evpna route-distinguisher 10.255.181.98:11
set routing-instances evpna vrf-target target:65000:11
set routing-instances evpna vrf-table-label
set routing-instances vrf instance-type vrf
set routing-instances vrf interface ge-0/0/0.0
set routing-instances vrf interface lo0.0
set routing-instances vrf route-distinguisher 10.255.181.98:100
set routing-instances vrf vrf-target target:100:100
set routing-instances vrf protocols ospf area 0 interface lo0.0 passive
set routing-instances vrf protocols ospf area 0 interface ge-0/0/0.0
set routing-instances vrf protocols pim rp static address 10.255.181.43
set routing-instances vrf protocols pim interface all

```

PE2

```

set interfaces ge-0/0/1 mtu 1600
set interfaces ge-0/0/1 unit 0 family inet address 198.51.100.6/24
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces xe-10/3/0 vlan-tagging
set interfaces xe-10/3/0 encapsulation flexible-ethernet-services

```



```

set interfaces xe-10/3/0 unit 10 encapsulation vlan-bridge
set interfaces xe-10/3/0 unit 10 vlan-id 10
set interfaces lo0 unit 1 family inet address 10.255.181.13/32
set protocols rsvp interface all
set protocols mpls no-cspf
set protocols mpls label-switched-path to-PE1 to 10.255.181.98
set protocols mpls interface all
set protocols bgp family evpn signaling
set protocols bgp group ibgp type internal
set protocols bgp group ibgp neighbor 10.255.181.98 local-address 10.255.181.13
set protocols ospf area 0 interface ge-0/0/1.0
set protocols ospf area 0 interface fxp0.0 disable
set protocols ospf area 0 interface lo0.0 passive
set routing-instances evpn10 vtep-source-interface lo0.0
set routing-instances evpn10 instance-type evpn
set routing-instances evpn10 vlan-id 10
set routing-instances evpn10 interface xe-10/3/0.10
set routing-instances evpn10 vxlan vni 10
set routing-instances evpn10 vxlan multicast-group 192.168.1.10
set routing-instances evpn10 vxlan encapsulate-inner-vlan
set routing-instances evpn10 vxlan decapsulate-accept-inner-vlan
set routing-instances evpn10 route-distinguisher 10.255.181.13:10
set routing-instances evpn10 vrf-target target:10:10
set routing-instances evpn10 protocols evpn
set routing-instances evpna instance-type vrf
set routing-instances evpna route-distinguisher 10.255.181.13:11
set routing-instances evpna vrf-target target:65000:11
set routing-instances evpna vrf-table-label
set routing-instances vrf instance-type vrf
set routing-instances vrf interface xe-10/3/0.0
set routing-instances vrf interface lo0.0
set routing-instances vrf route-distinguisher 10.255.181.13:100
set routing-instances vrf vrf-target target:100:100
set routing-instances vrf protocols ospf area 0 interface lo0.0 passive
set routing-instances vrf protocols ospf area 0 interface xe-10/3/0.0
set routing-instances vrf protocols pim rp static address 10.255.181.72
set routing-instances vrf protocols pim interface all

```

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 Device CE1:



NOTE: Repeat this procedure for Device CE2 after modifying the appropriate interface names, addresses, and other parameters.

1. Configure Device CE1 interfaces.

```
[edit interfaces]
user@CE1# set xe-0/0/0 vlan-tagging
user@CE1# set xe-0/0/0 encapsulation flexible-ethernet-services
user@CE1# set xe-0/0/0 unit 10 encapsulation vlan-bridge
user@CE1# set xe-0/0/0 unit 10 vlan-id 10
user@CE1# set ge-0/0/0 unit 0 family inet address 192.0.2.2/24
user@CE1# set ge-0/0/0 unit 0 family mpls
user@CE1# set lo0 unit 0 family inet address 10.255.181.43/32
```

2. Enable OSPF on Device CE1 interface, excluding the management interface.

```
[edit protocols]
user@CE1# set ospf area 0.0.0.0 interface ge-0/0/0.0
user@CE1# set ospf area 0.0.0.0 interface lo0.0 passive
user@CE1# set ospf area 0.0.0.0 interface fxp0.0 disable
```

3. Enable PIM on all the interfaces of Device CE1.

```
[edit protocols]
user@CE1# set pim rp local address 10.255.181.43
user@CE1# set pim interface all
```

4. Configure an EVPN bridge domain, and assign VLAN ID and interface.

```
[edit bridge-domains]
user@CE1# set evpn10 vlan-id 10
user@CE1# set evpn10 interface xe-0/0/0.10
```

5. Configure a VXLAN bridge domain, assign VXLAN ID, a multicast group address, and encapsulation and decapsulation parameters.

```
[edit bridge-domains]
user@CE1# set vxlan vni 10
user@CE1# set vxlan multicast-group 172.16.1.10
user@CE1# set vxlan encapsulate-inner-vlan
user@CE1# set vxlan decapsulate-accept-inner-vlan
```

Step-by-Step Procedure

To configure Device PE1:



NOTE: Repeat this procedure for Device PE2 after modifying the appropriate interface names, addresses, and other parameters.

1. Configure Device PE1 interfaces.

```
[edit interfaces]
user@PE1# set xe-0/0/2 vlan-tagging
user@PE1# set xe-0/0/2 encapsulation flexible-ethernet-services
user@PE1# set xe-0/0/2 unit 10 encapsulation vlan-bridge
user@PE1# set xe-0/0/2 unit 10 vlan-id 10
user@PE1# set ge-0/0/0 unit 0 family inet address 192.0.2.1/24
user@PE1# set ge-0/0/1 mtu 1600
user@PE1# set ge-0/0/1 unit 0 family inet address 198.51.100.5/24
user@PE1# set ge-0/0/1 unit 0 family mpls
user@PE1# set lo0 unit 1 family inet address 10.255.181.98/32
```

2. Enable MPLS and RSVP on all the interfaces of Device PE1.

```
[edit protocols]
user@PE1# set rsvp interface all
user@PE1# set mpls no-cspf
user@PE1# set mpls interface all
```

3. Configure a label-switched-path from Device PE1 to Device PE2.

```
[edit protocols]
user@PE1# set mpls label-switched-path to-PE2 to 10.255.181.13
```

4. Configure internal BGP peering between Devices PE1 and PE2, and enable EVPN signaling for the BGP session.

```
[edit protocols]
user@PE1# set bgp family evpn signaling
user@PE1# set bgp group ibgp type internal
user@PE1# set bgp group ibgp neighbor 10.255.181.13 local-address 10.255.181.98
```

5. Configure OSPF on Device PE1 interface, excluding the management interface.

```
[edit protocols]
user@PE1# set ospf area 0 interface ge-0/0/1.0
user@PE1# set ospf area 0 interface fxp0.0 disable
user@PE1# set ospf area 0 interface lo0.0 passive
```

6. Configure an EVPN routing instance, assign the VXLAN tunnel endpoint source interface, VLAN ID, assign route distinguisher and VRF target values, and assign Device PE1 interface to the routing instance.

```
[edit routing-instances]
user@PE1# set evpn10 vtep-source-interface lo0.0
user@PE1# set evpn10 instance-type evpn
user@PE1# set evpn10 vlan-id 10
user@PE1# set evpn10 interface xe-0/0/2.10
user@PE1# set evpn10 route-distinguisher 10.255.181.13:10
```

```
user@PE1# set evpn10 vrf-target target:10:10
user@PE1# set evpn10 protocols evpn
```

7. Assign the VXLAN ID, multicast group address, and encapsulation and decapsulation parameters for the EVPN routing instance.

```
[edit routing-instances]
user@PE1# set evpn10 vxlan vni 10
user@PE1# set evpn10 vxlan multicast-group 172.16.1.10
user@PE1# set evpn10 vxlan encapsulate-inner-vlan
user@PE1# set evpn10 vxlan decapsulate-accept-inner-vlan
```

8. Configure the first VPN routing and forwarding (VRF) routing instance, and assign route distinguisher and vrf-target values.

```
[edit routing-instances]
user@PE1# set evpna instance-type vrf
user@PE1# set evpna route-distinguisher 10.255.181.13:11
user@PE1# set evpna vrf-target target:65000:11
user@PE1# set evpna vrf-table-label
```

9. Configure the second VRF routing instance, and assign Device PE1 interfaces, route distinguisher and vrf-target values.

```
[edit routing-instances]
user@PE1# set vrf instance-type vrf
user@PE1# set vrf interface ge-0/0/0.0
user@PE1# set vrf interface lo0.0
user@PE1# set vrf route-distinguisher 10.255.181.13:100
user@PE1# set vrf vrf-target target:100:100
```

10. Configure OSPF and PIM protocols for the second VRF routing instance.

```
[edit routing-instances]
user@PE1# set vrf protocols ospf area 0 interface lo0.0 passive
user@PE1# set vrf protocols ospf area 0 interface ge-0/0/0.0
user@PE1# set vrf protocols pim rp static address 10.255.181.43
user@PE1# set vrf protocols pim interface all
```

Results

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

CE1

```
user@CE1# show interfaces
xe-0/0/0 {
  vlan-tagging;
  encapsulation flexible-ethernet-services;
  unit 10 {
    encapsulation vlan-bridge;
    vlan-id 10;
  }
}
ge-0/0/0 {
  unit 0 {
    family inet {
      address 192.0.2.2/24;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 10.255.181.43/32;
    }
  }
}
user@CE1# show protocols
ospf {
  area 0.0.0.0 {
    interface ge-0/0/0.0;
    interface lo0.0 {
      passive;
    }
    interface fxp0.0 {
      disable;
    }
  }
}
```

```

    }
}
pim {
    rp {
        local {
            address 10.255.181.43;
        }
    }
    interface all;
}
user@CE1# show bridge-domains
evpn10 {
    vlan-id 10;
    interface xe-0/0/0.10;
    vxlan {
        vni 10;
        multicast-group 172.16.1.10;
        encapsulate-inner-vlan;
        decapsulate-accept-inner-vlan;
    }
}

```

PE1

```

user@PE1# show interfaces
xe-0/0/2 {
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 10 {
        encapsulation vlan-bridge;
        vlan-id 10;
    }
}
ge-0/0/0 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
    }
}
ge-0/0/1 {
    mtu 1600;
}

```

```

    unit 0 {
        family inet {
            address 198.51.100.5/24;
        }
        family mpls;
    }
}
lo0 {
    unit 1 {
        family inet {
            address 10.255.181.98/32;
        }
    }
}
user@PE1# show protocols
rsvp {
    interface all;
}
mpls {
    no-cspf;
    label-switched-path to-PE2 {
        to 10.255.181.13;
    }
    interface all;
}
bgp {
    family evpn {
        signaling;
    }
    group ibgp {
        type internal;
        neighbor 10.255.181.13 {
            local-address 10.255.181.98;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-0/0/1.0;
        interface fxp0.0 {
            disable;
        }
        interface lo0.0 {

```



```

        passive;
    }
}
}
user@PE1# show routing-instances
evpn10 {
    vtep-source-interface lo0.0;
    instance-type evpn;
    vlan-id 10;
    interface xe-0/0/2.10;
    vxlan {
        vni 10;
        multicast-group 172.16.1.10;
        encapsulate-inner-vlan;
        decapsulate-accept-inner-vlan;
    }
    route-distinguisher 10.255.181.13:10;
    vrf-target target:10:10;
    protocols {
        evpn;
    }
}
evpna {
    instance-type vrf;
    route-distinguisher 10.255.181.98:11;
    vrf-target target:65000:11;
    vrf-table-label;
}
vrf {
    instance-type vrf;
    interface ge-0/0/0.0;
    interface lo0.0;
    route-distinguisher 10.255.181.98:100;
    vrf-target target:100:100;
    protocols {
        ospf {
            area 0.0.0.0 {
                interface lo0.0 {
                    passive;
                }
                interface ge-0/0/0.0;
            }
        }
    }
}

```

```
pim {  
    rp {  
        static {  
            address 10.255.181.43;  
        }  
    }  
    interface all;  
}  
}
```

Verification

IN THIS SECTION

- [Verifying MAC Learning | 1753](#)
- [Verifying PIM Reachability | 1754](#)
- [Verifying VXLAN Reachability | 1757](#)

Confirm that the configuration is working properly.

Verifying MAC Learning

Purpose

Verify the bridging and EVPN MAC table entries on CE and PE devices.

Action

On Device CE1, determine the bridging MAC table entries.

From operational mode, run the **show bridge mac-table** command.

```
user@CE1> show bridge mac-table
```

```
MAC flags      (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC  
O -OVSDDB MAC, SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)
```

```

Routing instance : default-switch
Bridging domain : evpn10, VLAN : 10
  MAC          MAC      Logical      NH      RTR
  address      flags    interface  Index   ID
  00:00:00:00:00:11  D      xe-0/0/0.10
  00:00:00:00:00:22  D      vtep.32769

```

On Device PE1, determine the EVPN MAC table entries.

From operational mode, run the **show evpn mac-table** command.

```

user@PE1> show evpn mac-table

MAC flags      (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
               O -OVSDDB MAC, SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : evpn10
Bridging domain : __evpn10__, VLAN : 10
  MAC          MAC      Logical      NH      RTR
  address      flags    interface  Index   ID
  00:00:00:00:00:11  D      vtep.32769
  00:00:00:00:00:22  DC

```

Meaning

The bridging and EVPN MAC tables have learned the VLAN configurations.

Verifying PIM Reachability

Purpose

Verify that the PIM configuration is working properly on the CE and PE devices.

Action

On Device CE1, verify PIM configuration.

From operational mode, run the **show pim rps extensive** command.

```

user@CE1> show pim rps extensive
Instance: PIM.master

```

```

address-family INET

RP: 10.255.181.43
Learned via: static configuration
Mode: Sparse
Time Active: 00:06:08
Holdtime: 150
Device Index: 161
Subunit: 32769
Interface: pd-0/2/0.32769
Static RP Override: Off
Group Ranges:
    224.0.0.0/4
Register State for RP:

```

Group	Source	FirstHop	RP Address	State	Timeout
172.16.1.10	203.1.113.11	203.1.113.11	10.255.181.43	Receive	171

```

address-family INET6

```

From operational mode, run the **show pim join extensive** command.

```

user@CE1> show pim join extensive
Instance: PIM.master Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 172.16.1.10
  Source: *
  RP: 10.255.181.43
  Flags: sparse,rptree,wildcard
  Upstream interface: Local
  Upstream neighbor: Local
  Upstream state: Local RP
  Uptime: 00:06:08
  Downstream neighbors:
    Interface: ge-0/0/0.0 (assert winner)
      192.0.2.1 State: Join Flags: SRW Timeout: 201
      Uptime: 00:05:08 Time since last Join: 00:00:08
      Assert Winner: 192.0.2.2 Metric: 0 Pref: 2147483648 Timeout: 82
    Interface: Pseudo-VXLAN
  Number of downstream interfaces: 2

```

```

Group: 172.16.1.10
  Source: 10.255.181.43
  Flags: sparse,spt
  Upstream interface: Local
  Upstream neighbor: Local
  Upstream state: Local Source, Local RP, No Prune to RP
  Keepalive timeout: 338
  Uptime: 00:04:15
  Downstream neighbors:
    Interface: ge-0/0/0.0
      192.0.2.1 State: Join Flags: S Timeout: 201
      Uptime: 00:04:15 Time since last Join: 00:00:08
    Interface: Pseudo-VXLAN
  Number of downstream interfaces: 2

Group: 172.16.1.10
  Source: 203.1.113.11
  Flags: sparse,spt
  Upstream interface: ge-0/0/0.0
  Upstream neighbor: 192.0.2.1 (assert winner)
  Upstream state: Local RP, Join to Source, No Prune to RP
  Keepalive timeout: 338
  Uptime: 00:04:15
  Downstream neighbors:
    Interface: ge-0/0/0.0 (pruned)
      192.0.2.1 State: Prune Flags: SR Timeout: 201
      Uptime: 00:04:15 Time since last Prune: 00:00:08
      Assert Winner: 192.0.2.1 Metric: 0 Pref: 0 Timeout: 179
    Interface: Pseudo-VXLAN
  Number of downstream interfaces: 2

Instance: PIM.master Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

```

Meaning

The device reachability using PIM is working as configured.

Verifying VXLAN Reachability

Purpose

Verify the connectivity between the VTEPs in the different data centers.

Action

From the operational mode, run the **show l2-learning vxlan-tunnel-end-point source**, **show l2-learning vxlan-tunnel-end-point remote**, and **show interfaces vtep** commands.

```
user@PE1> show l2-learning vxlan-tunnel-end-point source
```

Logical System Name	Id	SVTEP-IP	IFL	L3-Idx
<default>	0	203.1.113.11	lo0.0	7
L2-RTT	Bridge Domain		VNID	MC-Group-IP
evpn10	__evpn10__		10	172.16.1.10

```
user@PE2> show l2-learning vxlan-tunnel-end-point source
```

Logical System Name	Id	SVTEP-IP	IFL	L3-Idx
<default>	0	203.1.113.12	lo0.0	7
L2-RTT	Bridge Domain		VNID	MC-Group-IP
evpn10	__evpn10__		10	192.168.1.10

```
user@PE1> show l2-learning vxlan-tunnel-end-point remote
```

Logical System Name	Id	SVTEP-IP	IFL	L3-Idx
<default>	0	203.1.113.11	lo0.0	7
RVTEP-IP	IFL-Idx	NH-Id		
10.255.181.43	2684275660	2684275660		
VNID	MC-Group-IP			
10	172.16.1.10			

```
user@PE2> show l2-learning vxlan-tunnel-end-point remote
```

Logical System Name	Id	SVTEP-IP	IFL	L3-Idx
<default>	0	203.1.113.12	lo0.0	7
RVTEP-IP	IFL-Idx	NH-Id		
10.255.181.98	351	661		

VNID	MC-Group-IP
10	192.168.1.10

```

user@PE1> show interfaces vtep
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 133, SNMP ifIndex: 508
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: 1600, Speed: Unlimited
  Device flags   : Present Running
  Interface flags: SNMP-Traps
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 339) (SNMP ifIndex 560)
    Flags: Up SNMP-Traps Encapsulation: ENET2
    Ethernet segment value: 00:00:00:00:00:00:00:00:00:00, Mode: Single-homed, Multi-homed
status: Forwarding
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 203.1.113.11, L2 Routing Instance:
evpn10, L3 Routing Instance: vrf
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32769 (Index 341) (SNMP ifIndex 567)
    Flags: Up SNMP-Traps Encapsulation: ENET2
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.255.181.43, L2 Routing Instance:
evpn10, L3 Routing Instance: vrf
    Input packets : 143746
    Output packets: 95828
    Protocol bridge, MTU: 1600
    Flags: Trunk-Mode

```

Meaning

The output shows the correct tunnel source IP address (assigned to the loopback interface), VLAN, and multicast group for the VXLAN. Device PE1 is reachable because its IP address (the address assigned to the loopback interface) appears in the output. The output also shows that the VXLAN (VNI 10) and corresponding multicast group are configured correctly on the remote VTEP, Device PE2.

RELATED DOCUMENTATION

| [VXLAN Data Center Interconnect Using EVPN Overview](#) | 1719

Interconnecting EVPN-VXLAN Data Centers Through an EVPN-MPLS WAN

IN THIS CHAPTER

- [EVPN-VXLAN Data Center Interconnect Through EVPN-MPLS WAN Overview | 1760](#)
- [Example: Interconnecting EVPN-VXLAN Data Center Networks Through a WAN Running EVPN-based MPLS | 1771](#)
- [Overview of EVPN-VXLAN Interconnect through EVPN MPLS WAN Using Gateways | 1842](#)

EVPN-VXLAN Data Center Interconnect Through EVPN-MPLS WAN Overview

IN THIS SECTION

- [Interconnection of Data Center Networks Through WAN Overview | 1761](#)
- [Multi-homing on Data Center Gateways | 1764](#)
- [EVPN Designated Forwarder \(DF\) Election | 1764](#)
- [Split Horizon | 1765](#)
- [Aliasing | 1765](#)
- [VLAN-Aware Bundle Service | 1766](#)

You can interconnect different data center networks running Ethernet VPN (EVPN) with Virtual extensible LAN (VXLAN) encapsulation through a WAN running MPLS-based EVPN.

The following sections describe the technology and implementation overview of interconnecting data center networks running EVPN-VXLAN through a WAN running EVPN-MPLS to be used as a data center interconnect (DCI) solution.

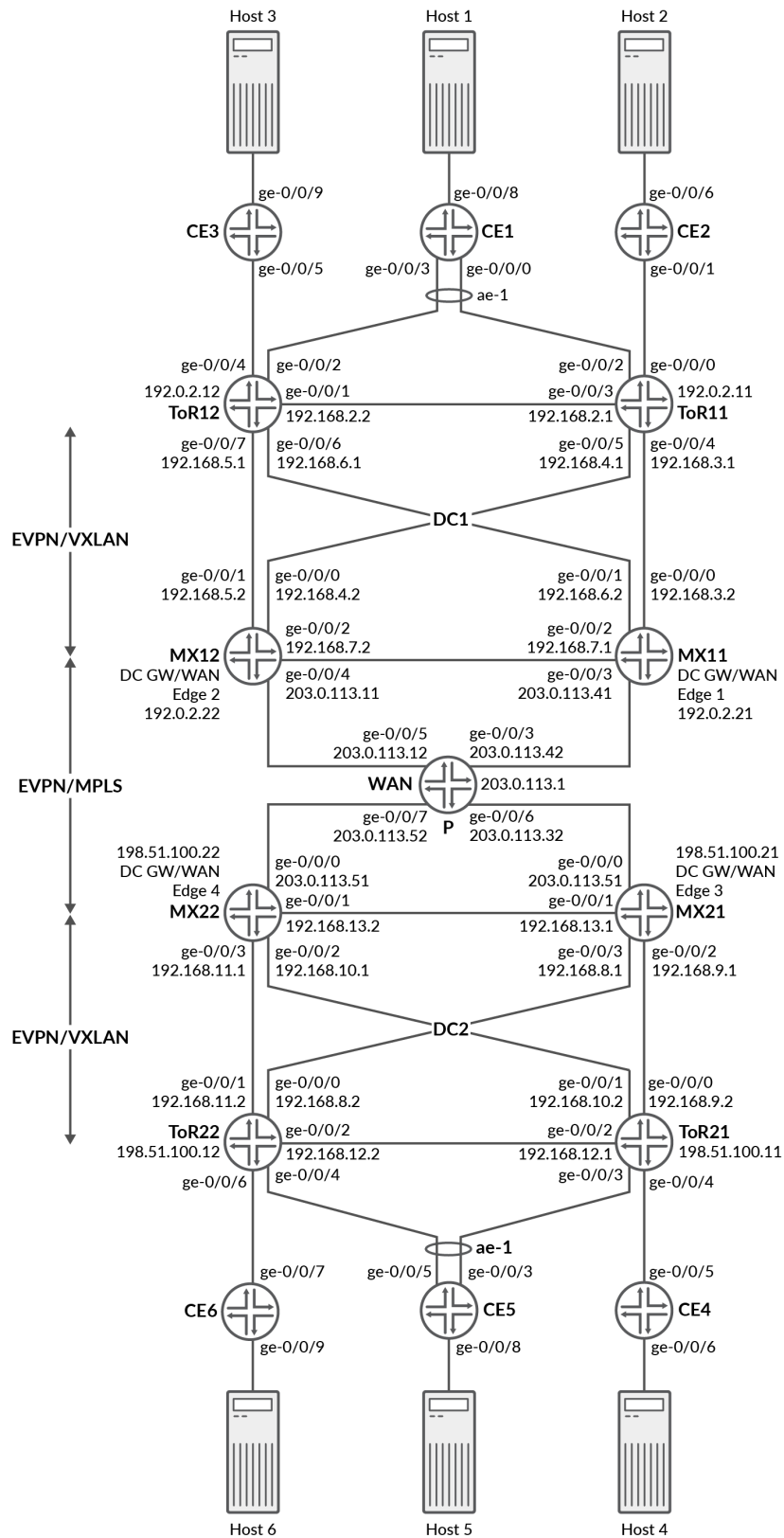
Interconnection of Data Center Networks Through WAN Overview

The following provides a conceptual overview of interconnecting different data center networks running Ethernet VPN (EVPN) with Virtual extensible LAN (VXLAN) encapsulation through a WAN running MPLS-based EVPN using the logical tunnel (It-) interface. You can:

- Connect data center edge routers over a WAN network running MPLS-based EVPN to achieve data center interconnection.
- Interconnect EVPN-VXLAN and EVPN-MPLS using the logical tunnel (It-) interface configured on the data center edge routers.

[Figure 183 on page 1762](#) illustration shows the interconnection of two data center networks (DC1 and DC2) running EVPN-VXLAN encapsulation through a WAN running MPLS-based EVPN:

Figure 183: EVPN-VXLAN Data Center Interconnect Through WAN Running EVPN-MPLS



In this illustration,

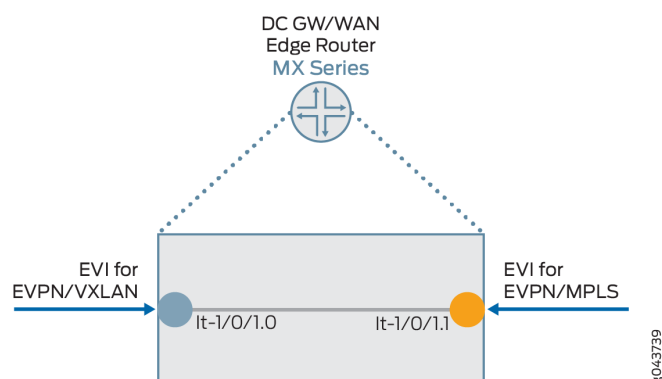
- The following devices are a part of the data center EVPN-VXLAN overlay network 1 (DC1):
 - Customer edge devices (CE1, CE2, and CE3) connected to the data center network.
 - VLAN hosts connected to each CE device.
 - MX routers playing the role of top-of-rack (ToR11 and ToR12) routers.
 - MX routers playing the role of the data center gateway router in the EVPN-VXLAN network and as a WAN edge router running MPLS-based EVPN (MX11 and MX12).
- The following devices are a part of the data center EVPN-VXLAN overlay network 2 (DC2):
 - Customer edge devices (CE4, CE5, and CE6) connected to the data center network.
 - VLAN hosts connected to each CE device.
 - MX routers playing the role of top-of-rack (ToR21 and ToR22) routers.
 - MX routers playing the role of the data center gateway router in the EVPN-VXLAN network and as a WAN edge router running MPLS-based EVPN (MX21 and MX22).

The interconnection of the data center network is realized on the data center gateway router through a pair of logical tunnel (It-) interface.

On the data center gateway router, you need to configure a pair of logical tunnel (It-) interface to interconnect the data center EVPN-VXLAN instance and the WAN MPLS-based EVPN instance: one logical tunnel (It-) interface is configured as the access interface for EVPN-VXLAN network and the other logical tunnel (It-) interface as the access interface for MPLS-based EVPN network as shown in the [Figure 184 on page 1764](#).

The support for active-active multi-homing is provided at the data center gateway routers for interconnection.

Figure 184: Logical Tunnel (It-) Interface of DC GW/WAN Edge Router Configured to Interconnect EVPN-VXLAN and EVPN-MPLS Instances



To configure EVPN-VXLAN and MPLS-based EVPN instances on the logical tunnel (It-) interface of the data center gateway router, see ["Example: Interconnecting EVPN-VXLAN Data Center Networks Through a WAN Running EVPN-based MPLS"](#) on page 1771.

Multi-homing on Data Center Gateways

You can configure redundant data center gateways and active-active multi-homing of EVPN-VXLAN network to a WAN running MPLS-based EVPN and active-active multi-homing of MPLS-based EVPN network to EVPN-VXLAN. This allows redundancy between the interconnection of EVPN-VXLAN network and MPLS-based EVPN WAN network. This also enables load-balancing of unicast traffic among the redundant data center gateways in both directions (from EVPN-VXLAN to EVPN-MPLS, as well as from EVPN-MPLS to EVPN-VXLAN). Broadcast, unknown unicast, and multicast (BUM) traffic is forwarded out of the data center by one of the data center gateways.

EVPN Designated Forwarder (DF) Election

To achieve active-active EVPN-VXLAN to EVPN-MPLS interconnect instance and active-active EVPN-MPLS to EVPN-VXLAN instance, the logical tunnel (It-) interface on the data center gateway router is configured with a non-zero Ethernet Segment Identifier (ESI). The ESI, a 10-octet value that must be unique across the entire network, is configured on a per port basis for the logical tunnel (It-) interface. As per the EVPN multi-homing procedure defined in the RFC7432, the following routes are advertised for an EVPN instance (EVI):

- Advertise an Ethernet segment route
- Advertise an ESI auto-discovery route with a valid split-horizon label and mode set to multi-homing

The standard EVPN DF election procedure described in RFC7432 is considered. The DF election is based on per Ethernet segment for each EVI. The EVPN-VXLAN and EVPN-MPLS run its DF election process independently.

Split Horizon

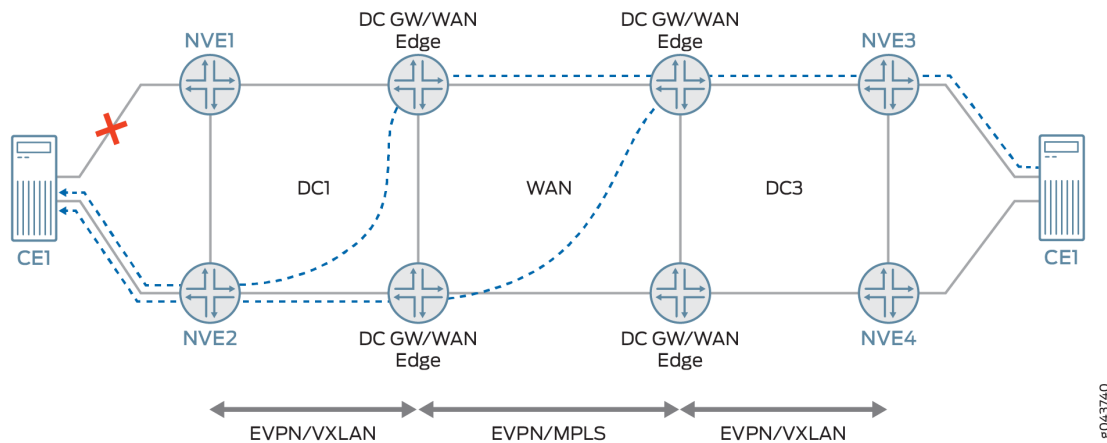
Split horizon prevents the looping of BUM traffic in a network, see RFC7432. For BUM traffic from core to the data center gateway (EVPN PE) direction, DF floods the BUM traffic to the access (It- interface) router and non-DF blocks the BUM traffic. When a DF or non-DF receives the BUM traffic coming from access (It- interface) router, it gets flooded to the core, but DF does not flood BUM traffic received from its non-DF to the access router based on split horizon rules. For a given BUM packet, only one copy is flooded to the access router (It- interface) and then to the EVPN core through one of the data center gateway routers because EVPN is multi-homed to another EVPN network. The DF filter rule from the first EVPN instance guarantees that only one copy of BUM traffic is forwarded from the DF to the It-interface before it re-enters the second EVPN instance.

Aliasing

When redundancy is configured in data center gateways, the traffic is load-balanced among redundant data center gateway routers on a per flow basis. MAC address is learned through the data plane using a pair of logical tunnel (It-) interfaces configured for EVPN-VXLAN instance and EVPN-MPLS instance for data center interconnect. However, the MAC owned by a host is always accessible by all the redundant data center gateways due to the nature of active-active multi-homing and full-mesh of EVPN PEs in both EVPN-VXLAN network and in WAN running EVPN-MPLS. Each EVPN instance on the data center gateway router declares the support of aliasing function for the ESI configured on the logical tunnel (It-) interface by advertising per EVI auto-discovery route. The aliasing functionality support is defined in the RFC7432.

[Figure 185 on page 1766](#) illustrates a link failure between CE1 and NVE1 but CE1 is still reachable by both data center gateway routers within the data center network (DC1).

Figure 185: Load Balancing Among Redundant DC GW/WAN Edge Routers



A link failure between a host and its top-of-rack (TOR) devices does not impact the aliasing functionality declared by the data center gateway router as the data center network itself is active-active to the WAN running EVPN-MPLS. As long as the host is connected to another ToR device in the data center network, the host is still accessible by all the other redundant data center gateway routers, so the aliasing functionality applies.

VLAN-Aware Bundle Service

In Junos OS for MX Series, both EVPN-VXLAN and EVPN-MPLS instances support VLAN-aware bundle service with one or more bridge domains. To connect two EVIs with VLAN-aware bundle service through a pair of logical tunnel (lt-) interface, it needs trunk interface support on the logical tunnel (lt-) interface, as well as trunk interface support for both EVPN-VXLAN and EVPN-MPLS instances. A trunk interface on Junos OS MX Series allows a logical interface to accept packets tagged with any VLAN ID specified in a VLAN ID list.

When the trunk mode is used for the logical tunnel (lt-) interface, the frames going out of the logical tunnel (lt-) interface trunk port from the first EVPN virtual switch are tagged with the appropriate VLAN tag; going through its peer logical tunnel (lt-) interface, the incoming frames to the second virtual switch are inspected and forwarded based on the VLAN tag found within the frame.

The following is a sample configuration to use trunk mode on logical tunnel (lt-) interface to support VLAN-aware bundle service for interconnection of EVPN-VXLAN with a WAN running MPLS-based EVPN:

```
interfaces lt-1/0/10 {
  esi {
    36:36:36:36:36:36:36:36:36:36;
    all-active;
  }
}
```

```

}
unit 3 {
    encapsulation ethernet-bridge;
    peer-unit 4;
    family bridge {
        interface-mode trunk;
        vlan-id-list [ 51 52 ];
    }
}
unit 4 {
    encapsulation ethernet-bridge;
    peer-unit 3;
    family bridge {
        interface-mode trunk;
        vlan-id-list [ 51 52 ];
    }
}
}

```

The following is a sample configuration of trunk port support for EVPN-VXLAN and EVPN-MPLS:

```

routing-instances evpn-mpls {
    vtep-source-interface lo0.0;
    instance-type virtual-switch;
    interface lt-1/0/10.4;
    route-distinguisher 101:2;
    vrf-target target:2:2;
    protocols {
        evpn {
            extended-vlan-list 51-52;
        }
    }
}
bridge-domains {
    bd1 {
        domain-type bridge;
        vlan-id 51;
    }
    bd2 {
        domain-type bridge;
        vlan-id 52;
    }
}
}

```



```

}
routing-instances red {
vtep-source-interface lo0.0;
    instance-type virtual-switch;
    interface lt-1/0/10.4
    route-distinguisher 101:1;
    vrf-target target:1:1;
    protocols {
    evpn {
        encapsulation vxlan;
        extended-vni-list all;
    }
    }
}
bridge-domains {
    bd1 {
        domain-type bridge;
        vlan-id 51;
        routing-interface irb.0;
        vxlan {
            vni 51;
            encapsulate-inner-vlan;
            decapsulate-accept-inner-vlan;
        }
    }
    bd2 {
        domain-type bridge;
        vlan-id 52;
        routing-interface irb.1;
        vxlan {
            vni 52;
            encapsulate-inner-vlan;
            decapsulate-accept-inner-vlan;
        }
    }
}
}
}

```

Data Center Network Design and Considerations

Before designing a data center network, you need to decide whether to use IGP, or iBGP, or eBGP protocols in the data center network for IP underlay. Another important factor to consider is the AS

assignment. The ToR device in the data center network is required to have an AS number that is different than the AS number used in the WAN edge router.

For the overlay network, you need to decide whether to use iBGP, or eBGP, or a combination of both iBGP and eBGP.

Figure 186: Data Center Network Design

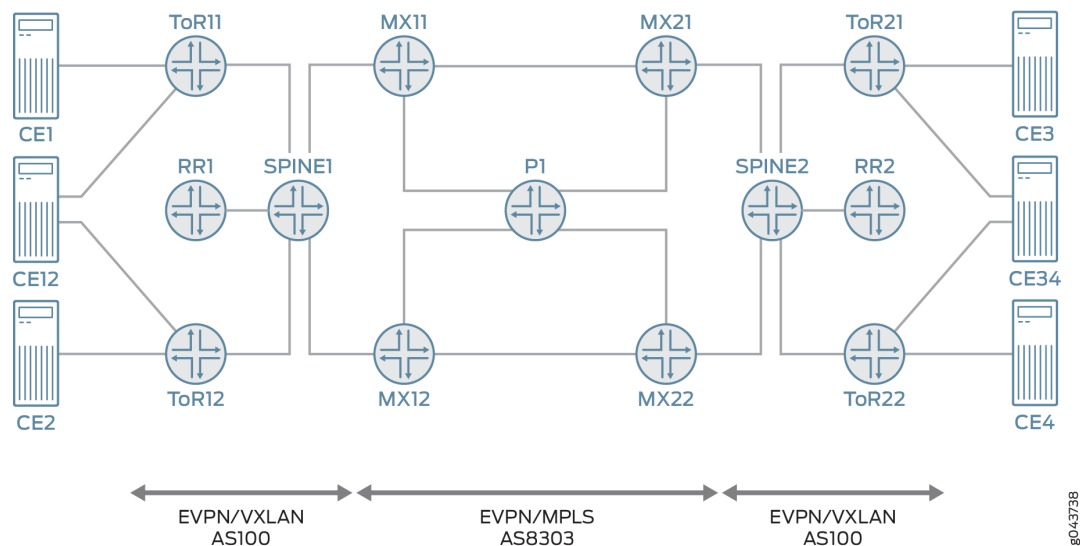


Figure 186 on page 1769 illustrates the MX routers (MX11, MX12, MX21 and MX22) as the data center gateway and WAN edge routers that interconnects EVPN-VXLAN to EVPN-MPLS. Spine switches provide connection for east and west traffic among ToRs so that traffic that does not need to be Layer 3 routed does not go through the MX routers. From a network design perspective, to provide an end-to-end EVPN solution, the following requirements must be met:

Isolate IGP Between EVPN-VXLAN and EVPN-MPLS Segments

When IGP is used in the data center network, you need to isolate the IP network in EVPN-VXLAN from the IP network in the WAN. When IGP is used in the data center, one option is not to run IGP protocol on the interfaces that connects spine switches and MX routers. Instead, an eBGP session with address family inet unicast is used between spine switches and MX routers such that through IGP/eBGP/policy you can leak loopback addresses of ToR and MX routers to each other and still maintain the isolation of IP network in the data center from WAN. In the EVPN-VXLAN segment, IGP is between spine switches and ToRs only. In the EVPN-MPLS segment, IGP is between all the MX routers.

Using iBGP for IP Underlay in the Data Center Network

If the requirement is to not use IGP in the IP underlay in the data center, iBGP with address family inet unicast can be used to replace OSPF between spine switches and ToRs. Between spine switches and data center gateways, you still need to use eBGP for advertising loopback IP.

Using eBGP for the IP Underlay in the Data Center Network

If the requirement is to use eBGP only in the data center, you need to use eBGP with address family inet unicast for the IP underlay. In this case, it is a typical 2 stage CLOS network without spine aggregation layer. Each ToR and data center gateway is assigned a unique AS number. ToR establishes eBGP session with data center gateway routers directly.

Different Autonomous Systems (AS) in EVPN-VXLAN and EVPN-MPLS Network

The following is support for different AS in EVPN-VXLAN and EVPN-MPLS network running iBGP, or eBGP for the IP overlay.

Runing iBGP/eBGP for the Overlay

ToRs and spine switches are in the same AS100 and all MX Series routers are in AS8303. Among ToRs, its EVPN Network Layer Reachability Information (NLRI) is exchanged through iBGP session. A BGP route reflector (RR) is used and each ToR establishes iBGP session with the RR. Data traffic between ToRs that belongs to the same bridge domain goes through spine switch only and it is always 2 hops away. Since the ToRs and spine switches are in the same AS and the MX edge routers are in the different AS, the MX edge router establishes eBGP session to either RR or each ToR directly. By default, route learned from iBGP session (ToRs) are re-advertised to the eBGP (MX routers) and vice versa. BGP next-hop unchanged is enforced when BGP re-advertises EVPN NLRI among iBGP and eBGP session.

Running eBGP only for the Overlay

If the requirement is to run eBGP only in the data center, each ToR is assigned a unique AS number. Each data center gateway router uses a unique AS number on the data center facing side. For the WAN facing side, the same AS number is used, but the AS number would be different from the AS number used for the data center facing side. The AS number may also be reused in each data center.

To prevent an EVPN route in the data center to be advertised to the data center gateway routers in another data center, you must turn on the route constrain in the EVPN-MPLS network. To make BGP route constrain to work, different route target is used for EVPN-VXLAN and EVPN-MPLS network, respectively.

RELATED DOCUMENTATION

[Example: Interconnecting EVPN-VXLAN Data Center Networks Through a WAN Running EVPN-based MPLS | 1771](#)

Example: Interconnecting EVPN-VXLAN Data Center Networks Through a WAN Running EVPN-based MPLS

IN THIS SECTION

- [Requirements | 1771](#)
- [Overview | 1772](#)
- [Configuration | 1775](#)
- [Verification | 1790](#)
- [Appendix 1: Set Commands on All Devices | 1796](#)
- [Appendix 2: Show Configuration Output on DUT | 1826](#)

This example shows how to interconnect EVPN-VXLAN data center networks through a WAN running EVPN-MPLS to leverage the benefits of EVPN as a Data Center Interconnect (DCI) solution.

Requirements

This example uses the following hardware and software components:

- Four Juniper Networks MX Series routers to be configured as data center gateways and WAN edge routers.
- Four Juniper Networks MX Series routers to be configured as top-of-rack (ToR) routers.
- Six customer edge (CE) devices.
- Six host devices connected to each CE device that has the capability to configure multiple VLANs.
- One provider (P) router part of the EVPN-MPLS WAN network.
- Junos OS Release 17.2 or later.

Overview

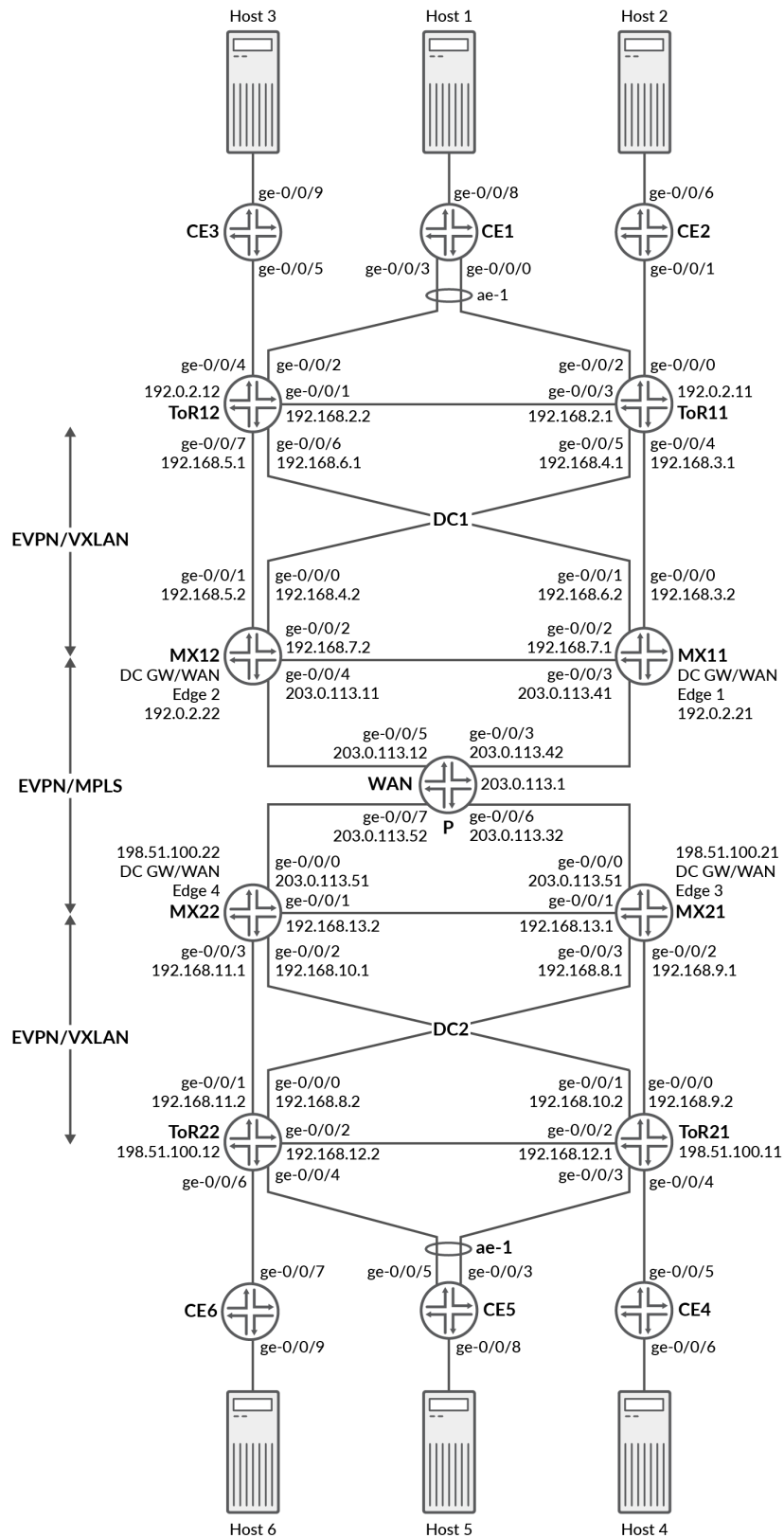
You can interconnect different data center networks running Ethernet VPN (EVPN) with Virtual extensible LAN (VXLAN) encapsulation through a WAN running MPLS-based EVPN using the logical tunnel (It-) interface.

[Figure 187 on page 1773](#) illustrates the interconnection of data center networks running EVPN with VXLAN encapsulation through a WAN running MPLS-based EVPN. For the purposes of this example, the MX Series routers acting as data center gateways and as WAN edge routers are named MX11, MX12, MX21, and MX22. The MX Series routers acting as top-of-rack (ToR) routers are named ToR11, ToR12, ToR21, and ToR22. The customer edge (CE) devices connected to the data center network 1 (DC1) are named CE1, CE2, and CE3. The customer edge (CE) devices connected to the data center network 2 (DC2) are named CE4, CE5, and CE6. The host devices connected to each CE device should be able to configure multiple host VLANs. The WAN provider router is named P.



NOTE: CE devices are part of the logical system of ToR devices.

Figure 187: EVPN-VXLAN Data Center Interconnect Through WAN Running EVPN-MPLS



For the MX Series routers acting as data center gateways and WAN edge routers, configure the following information:

- IRB interfaces, virtual gateway addresses, and loopback logical interfaces.
- External BGP (EBGP) underlay connectivity between gateway and ToR routers, EVPN as the signaling protocol.
- Routing policies to allow specific routes into the virtual-switch tables.
- Routing instances (Layer 3 VRFs) for each virtual network, including a unique *route-distinguisher*, and a *vrf-target* value.
- Virtual-switch instances (Layer 2 MAC-VRFs) for each virtual network, the VTEP source interface (always lo0.0), route distinguisher, and *vrf-import* policy.
- EVPN protocol, encapsulation method, VNI list, and BUM traffic forwarding method for each virtual switch.
- Bridge domain within each virtual switch that maps VNIDs to VLAN IDs, an IRB (Layer 3) interface, and the BUM forwarding method.

For the MX Series routers acting as top-of-rack (ToR) routers, configure the following information:

- Host facing interfaces with VLANs, VLAN IDs, and loopback logical interfaces.
- Link Aggregation Control Protocol (LACP)-enabled link aggregation group (LAG), Ethernet Segment ID (ESI), and all-active mode.
- Multiprotocol external BGP (MP-EBGP) overlays between ToR and gateway routers using EVPN as the signaling protocol.
- EVPN with VXLAN as the encapsulation method, *extended-vni-list*, multicast mode, and route targets for each VNI.
- Vrf import policy, *vtep-source-interface*, route-distinguisher, and vrf import and target information.
- VLANs, with VLAN IDs mapped to globally significant VNIs.



NOTE: You can set the virtual gateway address as the default IPv4 or IPv6 gateway address for end hosts (virtual machines or servers).

Configuration

IN THIS SECTION

- [Configuring ToR11 | 1775](#)
- [Configuring Data Center Gateway and WAN Edge 1 Router \(MX11\) | 1779](#)
- [Configuring WAN Router \(P\) | 1786](#)
- [Configuring CE1 | 1789](#)

Configuring ToR11

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*.

Configure the MX router as ToR11:



NOTE: Follow similar steps for ToR12, ToR21, and ToR22. Refer to [Figure 187 on page 1773](#) for interface name, IP address, and connectivity.

1. Set the system hostname.

```
[edit]
user@ToR11# set system host-name ToR11
```

2. Set the number of aggregated Ethernet interfaces.

```
[edit]
user@ToR11# set chassis aggregated-devices ethernet device-count 1
```


3. Configure the interfaces on the ToR11 device to connect to the MX12, CE-2, CE-1, ToR12, and MX11 devices to enable underlay connectivity.

[edit]

```
user@ToR11# set interfaces ge-0/0/0 unit 0 description "CONNECTED TO CE-2"
user@ToR11# set interfaces ge-0/0/0 unit 0 family bridge interface-mode trunk
user@ToR11# set interfaces ge-0/0/0 unit 0 family bridge vlan-id-list 1-5
user@ToR11# set interfaces ge-0/0/2 description "CONNECTED TO CE-1"
user@ToR11# set interfaces ge-0/0/2 gigether-options 802.3ad ae0
user@ToR11# set interfaces ge-0/0/3 unit 0 description "CONNECTED TO ToR12"
user@ToR11# set interfaces ge-0/0/3 unit 0 family inet address 192.168.2.1/24
user@ToR11# set interfaces ge-0/0/4 unit 0 description "CONNECTED TO MX-11"
user@ToR11# set interfaces ge-0/0/4 unit 0 family inet address 192.168.3.1/24
user@ToR11# set interfaces ge-0/0/5 unit 0 description "CONNECTED TO MX-12"
user@ToR11# set interfaces ge-0/0/5 unit 0 family inet address 192.168.4.1/24
```

4. Configure a Link Aggregation Control Protocol (LACP)-enabled link aggregation group (LAG) interface towards the CE-1 end host device. The ESI value is globally unique across the entire EVPN domain. The all-active configuration enables ToR11 and ToR12 to forward traffic to, and from the CE devices, such that all CE links are actively used.

[edit]

```
user@ToR11# set interfaces ae0 esi 00:11:11:11:11:11:11:11:11
user@ToR11# set interfaces ae0 esi all-active
user@ToR11# set interfaces ae0 aggregated-ether-options lacp active
user@ToR11# set interfaces ae0 aggregated-ether-options lacp periodic fast
user@ToR11# set interfaces ae0 aggregated-ether-options lacp system-id 11:11:11:11:11:11
user@ToR11# set interfaces ae0 unit 0 family bridge interface-mode trunk
user@ToR11# set interfaces ae0 unit 0 family bridge vlan-id-list 1-5
```

5. Configure the loopback interface address and routing options.

[edit]

```
user@ToR11# set interfaces lo0 unit 0 family inet address 192.0.2.11/32
user@ToR11# set routing-options router-id 192.0.2.11
user@ToR11# set routing-options autonomous-system 65100
```

6. Apply the load balancing policy to the forwarding table.

```
[edit]
user@ToR11# set routing-options forwarding-table export evpn-pplb
```

7. Configure routing policy to accept the direct loopback address route.

```
[edit]
user@ToR11# set policy-options policy-statement L0 term 1 from protocol direct
user@ToR11# set policy-options policy-statement L0 term 1 from route-filter 192.0.2.11/32
exact
user@ToR11# set policy-options policy-statement L0 term 1 then accept
```

8. Configure the NO-EXPORT community.

```
[edit]
user@ToR11# set policy-options community NO-EXPORT members no-advertise
user@ToR11# set policy-options community NO-EXPORT members no-export
user@ToR11# set policy-options community NO-EXPORT members no-export-subconfed
```

9. Configure the load balancing and TEST policies.

```
[edit]
user@ToR11# set policy-options policy-statement TEST then community add NO-EXPORT
user@ToR11# set policy-options policy-statement evpn-pplb from protocol evpn
user@ToR11# set policy-options policy-statement evpn-pplb then load-balance per-packet
```

10. Configure EVPN routing instances for each virtual network. Define the VTEP source interface, route distinguisher (used to identify and advertise EVPN routes), and vrf-target (exports and tags all routes for that local VRF using the defined route target). Configure the EVPN protocol, encapsulation method, VNI list, and BUM traffic forwarding method. Finally, configure a bridge domain for each virtual router that maps VNIDs to VLAN IDs, and identify the BUM forwarding method.

```
[edit]
user@ToR11# set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
user@ToR11# set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
user@ToR11# set routing-instances EVPN-VXLAN-1 protocols evpn default-gateway no-gateway-
```

```

community
user@ToR11# set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5
user@ToR11# set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 routing-interface irb.1
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 routing-interface irb.2
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 routing-interface irb.3
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 routing-interface irb.4
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 routing-interface irb.5
user@ToR11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
user@ToR11# set routing-instances EVPN-VXLAN-1 interface ge-0/0/0.0
user@ToR11# set routing-instances EVPN-VXLAN-1 interface ae0.0
user@ToR11# set routing-instances EVPN-VXLAN-1 route-distinguisher 192.0.2.21:1
user@ToR11# set routing-instances EVPN-VXLAN-1 vrf-target target:1:1
user@ToR11# set routing-instances VRF instance-type vrf
user@ToR11# set routing-instances VRF interface irb.1
user@ToR11# set routing-instances VRF interface irb.2
user@ToR11# set routing-instances VRF interface irb.3
user@ToR11# set routing-instances VRF interface irb.4
user@ToR11# set routing-instances VRF interface irb.5
user@ToR11# set routing-instances VRF route-distinguisher 1:1
user@ToR11# set routing-instances VRF vrf-target target:10:10

```

Configuring Data Center Gateway and WAN Edge 1 Router (MX11)

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*.

Configure an MX Series router as the data center gateway and WAN edge router and name it as MX11:



NOTE: Follow similar steps for MX12, MX21, and MX22. Refer to [Figure 187 on page 1773](#) for interface name, IP address, and connectivity.

1. Set the system hostname.

```
[edit]
user@MX11# set system host-name MX11
```

2. Configure the interfaces on the MX11 router (DC GW/WAN Edge1) to enable the underlay connectivity to the MX12, ToR11, ToR12, and P devices, which is the EVPN-VXLAN part of DC1 network.

```
[edit]
user@MX11# set interfaces ge-0/0/0 unit 0 description "CONNECTED TO ToR11"
user@MX11# set interfaces ge-0/0/0 unit 0 family inet address 192.168.3.2/24
user@MX11# set interfaces ge-0/0/1 unit 0 description "CONNECTED TO ToR12"
user@MX11# set interfaces ge-0/0/1 unit 0 family inet address 192.168.6.2/24
user@MX11# set interfaces ge-0/0/2 unit 0 description "CONNECTED TO MX12"
user@MX11# set interfaces ge-0/0/2 unit 0 family inet address 192.168.7.1/24
user@MX11# set interfaces ge-0/0/3 unit 0 description "CONNECTED TO P"
user@MX11# set interfaces ge-0/0/3 unit 0 family inet address 203.0.113.41/24
user@MX11# set interfaces ge-0/0/3 unit 0 family mpls
```

3. Configure external BGP (EBGP) underlay connectivity between the gateway routers (MX11 and MX12) and ToR (ToR11 and ToR12).

```
[edit]
user@MX11# set protocols bgp group MX12 type external
user@MX11# set protocols bgp group MX12 local-address 192.168.7.1
user@MX11# set protocols bgp group MX12 export TEST
```

```

user@MX11# set protocols bgp group MX12 export L0
user@MX11# set protocols bgp group MX12 peer-as 65500
user@MX11# set protocols bgp group MX12 local-as 65400
user@MX11# set protocols bgp group MX12 neighbor 192.168.7.2 family inet unicast
user@MX11# set protocols bgp group ToR11 type external
user@MX11# set protocols bgp group ToR11 local-address 192.168.3.2
user@MX11# set protocols bgp group ToR11 import TEST
user@MX11# set protocols bgp group ToR11 export TEST
user@MX11# set protocols bgp group ToR11 export L0
user@MX11# set protocols bgp group ToR11 peer-as 65100
user@MX11# set protocols bgp group ToR11 local-as 65400
user@MX11# set protocols bgp group ToR11 neighbor 192.168.3.1 family inet unicast
user@MX11# set protocols bgp group ToR12 type external
user@MX11# set protocols bgp group ToR12 local-address 192.168.6.2
user@MX11# set protocols bgp group ToR12 export TEST
user@MX11# set protocols bgp group ToR12 export L0
user@MX11# set protocols bgp group ToR12 peer-as 65200
user@MX11# set protocols bgp group ToR12 local-as 65400
user@MX11# set protocols bgp group ToR12 neighbor 192.168.6.1 family inet unicast

```

4. Configure a multiprotocol external BGP (MP-EBGP) overlay connectivity between the gateway routers (MX11 and MX12) and ToR (ToR11 and ToR12) and set EVPN as the signaling protocol.

```

[edit]
user@MX11# set protocols bgp group MX12-EVPN type external
user@MX11# set protocols bgp group MX12-EVPN multihop ttl 2
user@MX11# set protocols bgp group MX12-EVPN multihop no-nexthop-change
user@MX11# set protocols bgp group MX12-EVPN local-address 192.0.2.21
user@MX11# set protocols bgp group MX12-EVPN export TEST
user@MX11# set protocols bgp group MX12-EVPN peer-as 65500
user@MX11# set protocols bgp group MX12-EVPN local-as 65400
user@MX11# set protocols bgp group MX12-EVPN neighbor 192.0.2.22 family evpn signaling
user@MX11# set protocols bgp group ToR11-EVPN type external
user@MX11# set protocols bgp group ToR11-EVPN multihop ttl 2
user@MX11# set protocols bgp group ToR11-EVPN multihop no-nexthop-change
user@MX11# set protocols bgp group ToR11-EVPN local-address 192.0.2.21
user@MX11# set protocols bgp group ToR11-EVPN export TEST
user@MX11# set protocols bgp group ToR11-EVPN peer-as 65100
user@MX11# set protocols bgp group ToR11-EVPN local-as 65400
user@MX11# set protocols bgp group ToR11-EVPN neighbor 192.0.2.11 family evpn signaling
user@MX11# set protocols bgp group ToR12-EVPN type external
user@MX11# set protocols bgp group ToR12-EVPN multihop ttl 2

```

```

user@MX11# set protocols bgp group ToR12-EVPN multihop no-nexthop-change
user@MX11# set protocols bgp group ToR12-EVPN local-address 192.0.2.21
user@MX11# set protocols bgp group ToR12-EVPN export TEST
user@MX11# set protocols bgp group ToR12-EVPN peer-as 65200
user@MX11# set protocols bgp group ToR12-EVPN local-as 65400
user@MX11# set protocols bgp group ToR12-EVPN neighbor 192.0.2.12 family evpn signaling

```

5. Configure integrated routing and bridging (IRB) interfaces that advertise the MAC and IP routes (MAC+IP type 2 routes) for hosts in the topology. The IRB configuration is the gateway for the VLANs on the hosts.

Step-by-Step Procedure

- a. The following is the IRB gateway configuration for the VLAN-1 on MX11 (which is the host part of VLAN-1):

```

[edit]
user@MX11# set interfaces irb unit 1 proxy-macip-advertisement
user@MX11# set interfaces irb unit 1 virtual-gateway-esi 00:11:aa:aa:aa:aa:aa:aa:aa:aa
user@MX11# set interfaces irb unit 1 virtual-gateway-esi all-active
user@MX11# set interfaces irb unit 1 family inet address 10.11.1.12/24 virtual-gateway-
address 10.11.1.10

```

- b. The following is the IRB gateway configuration for the VLAN-2 on MX11 (which is the host part of VLAN-2):

```

[edit]
user@MX11# set interfaces irb unit 2 proxy-macip-advertisement
user@MX11# set interfaces irb unit 2 virtual-gateway-esi 00:11:bb:bb:bb:bb:bb:bb:bb:bb
user@MX11# set interfaces irb unit 2 virtual-gateway-esi all-active
user@MX11# set interfaces irb unit 2 family inet address 10.12.1.12/24 virtual-gateway-
address 10.12.1.10

```

- c. The following is the IRB gateway configuration for the VLAN-3 on MX11 (which is the host part of VLAN-3):

```

[edit]
user@MX11# set interfaces irb unit 3 proxy-macip-advertisement
user@MX11# set interfaces irb unit 3 virtual-gateway-esi 00:11:cc:cc:cc:cc:cc:cc:cc:cc
user@MX11# set interfaces irb unit 3 virtual-gateway-esi all-active

```

```
user@MX11# set interfaces irb unit 3 family inet address 10.13.1.12/24 virtual-gateway-
address 10.13.1.10
```

- d. The following is the IRB gateway configuration for the VLAN-4 on MX11 (which is the host part of VLAN-4):

```
[edit]
user@MX11# set interfaces irb unit 4 proxy-macip-advertisement
user@MX11# set interfaces irb unit 4 virtual-gateway-esi 00:11:dd:dd:dd:dd:dd:dd:dd
user@MX11# set interfaces irb unit 4 virtual-gateway-esi all-active
user@MX11# set interfaces irb unit 4 family inet address 10.14.1.12/24 virtual-gateway-
address 10.14.1.10
```

- e. The following is the IRB gateway configuration for the VLAN-5 on MX11 (which is the host part of VLAN-5):

```
[edit]
user@MX11# set interfaces irb unit 5 proxy-macip-advertisement
user@MX11# set interfaces irb unit 5 virtual-gateway-esi 00:11:ee:ee:ee:ee:ee:ee:ee
user@MX11# set interfaces irb unit 5 virtual-gateway-esi all-active
user@MX11# set interfaces irb unit 5 family inet address 10.15.1.12/24 virtual-gateway-
address 10.15.1.10
```

6. Configure routing policy to accept the direct loopback address route.

```
[edit]
user@MX11# set policy-options policy-statement L0 from protocol direct
user@MX11# set policy-options policy-statement L0 from route-filter 192.0.2.21/32 exact
user@MX11# set policy-options policy-statement L0 then accept
```

7. Configure the NO-EXPORT community.

```
[edit]
user@MX11# set policy-options community NO-EXPORT members no-advertise
user@MX11# set policy-options community NO-EXPORT members no-export
user@MX11# set policy-options community NO-EXPORT members no-export-subconfed
```

8. Configure the load balancing and TEST policies.

```
[edit]
user@MX11# set policy-options policy-statement TEST then community add NO-EXPORT
user@MX11# set policy-options policy-statement evpn-pplb from protocol evpn
user@MX11# set policy-options policy-statement evpn-pplb then load-balance per-packet
```

9. Configure an ESI value on the logical tunnel interface. Use the same ESI value on all other gateway/WAN edge routers in the DC1 network.

```
[edit]
user@MX11# set interfaces lt-5/1/0 esi 00:22:22:22:22:22:22:22:22
```

10. Configure active-active multihoming on the logical tunnel interface by including the all-active statement.

```
[edit]
user@MX11# set interfaces lt-5/1/0 esi all-active
```

11. Configure a pair of logical tunnel (lt-) interfaces on the MX11 gateway router to interconnect the EVPN-VXLAN instance of the data center network with the MPLS-based EVPN instance of the WAN. One logical tunnel (lt-) interface is configured as the access interface for EVPN-VXLAN and the other logical tunnel (lt-) interface is configured as the access interface for MPLS-based EVPN.

```
[edit]
user@MX11# set interfaces lt-5/1/0 unit 0 description TO-EVPN-VXLAN
user@MX11# set interfaces lt-5/1/0 unit 0 encapsulation ethernet-bridge
user@MX11# set interfaces lt-5/1/0 unit 0 peer-unit 1
user@MX11# set interfaces lt-5/1/0 unit 0 family bridge interface-mode trunk
user@MX11# set interfaces lt-5/1/0 unit 0 family bridge vlan-id-list 1-5
user@MX11# set interfaces lt-5/1/0 unit 1 description TO-EVPN-MPLS
user@MX11# set interfaces lt-5/1/0 unit 1 encapsulation ethernet-bridge
user@MX11# set interfaces lt-5/1/0 unit 1 peer-unit 0
user@MX11# set interfaces lt-5/1/0 unit 1 family bridge interface-mode trunk
user@MX11# set interfaces lt-5/1/0 unit 1 family bridge vlan-id-list 1-5
```


12. Configure the loopback interface address and routing options.

```
[edit]
user@MX11# set interfaces lo0 unit 0 family inet address 192.0.2.21/32
user@MX11# set interfaces lo0 unit 0 family mpls
user@MX11# set routing-options router-id 192.0.2.21
user@MX11# set routing-options autonomous-system 65300
```

13. Apply the load balancing policy to the forwarding table.

```
[edit]
user@MX11# set routing-options forwarding-table export evpn-pplb
```

14. Enable MPLS, BGP, and OSPF protocols on the core interfaces. Create MPLS LSPs and specify the address of the other gateway and WAN edge routers (MX12, P, MX21, MX22).

```
[edit]
user@MX11# set protocols mpls label-switched-path MX11-T0-MX12 to 192.0.2.22
user@MX11# set protocols mpls label-switched-path MX11-T0-P to 203.0.113.1
user@MX11# set protocols mpls label-switched-path MX11-T0-MX21 to 198.51.100.21
user@MX11# set protocols mpls label-switched-path MX11-T0-MX22 to 198.51.100.22
user@MX11# set protocols mpls interface all
user@MX11# set protocols mpls interface fxp0.0 disable
user@MX11# set protocols bgp local-address 192.0.2.21
user@MX11# set protocols bgp local-as 65300
user@MX11# set protocols bgp group INT type internal
user@MX11# set protocols bgp group INT local-address 192.0.2.21
user@MX11# set protocols bgp group INT family evpn signaling
user@MX11# set protocols bgp group INT export TEST
user@MX11# set protocols bgp group INT neighbor 203.0.113.1
user@MX11# set protocols ospf traffic-engineering
user@MX11# set protocols ospf area 0.0.0.0 interface ge-0/0/3.0
user@MX11# set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

15. Configure EVPN-based MPLS routing instances on the MX11 router for each virtual network. Define the route distinguisher (used to identify and advertise EVPN-MPLS routes) and vrf-target

(exports and tags all routes for that local VRF using the defined route target). Configure a bridge domain for each virtual router that maps VLAN IDs.

```
[edit]
user@MX11# set routing-instances EVPN-MPLS-1 instance-type virtual-switch
user@MX11# set routing-instances EVPN-MPLS-1 interface lt-5/1/0.0
user@MX11# set routing-instances EVPN-MPLS-1 route-distinguisher 192.0.2.21:100
user@MX11# set routing-instances EVPN-MPLS-1 vrf-target target:1:2
user@MX11# set routing-instances EVPN-MPLS-1 protocols evpn extended-vlan-list 1-5
user@MX11# set routing-instances EVPN-MPLS-1 protocols evpn default-gateway no-gateway-
community
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-1 domain-type bridge
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-1 vlan-id 1
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-2 domain-type bridge
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-2 vlan-id 2
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-3 domain-type bridge
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-3 vlan-id 3
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-4 domain-type bridge
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-4 vlan-id 4
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-5 domain-type bridge
user@MX11# set routing-instances EVPN-MPLS-1 bridge-domains BD-5 vlan-id 5
```

16. Configure EVPN-VXLAN routing instances on the MX11 router for each virtual network. Define the VTEP source interface, route distinguisher (used to identify and advertise EVPN routes), and vrf-target (exports and tags all routes for that local VRF using the defined route target). Configure the EVPN protocol, encapsulation method, VNI list, and BUM traffic forwarding method. Finally, configure a bridge domain for each virtual router that maps VNIDs to VLAN IDs, and identify the BUM forwarding method.

```
[edit]
user@MX11# set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
user@MX11# set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
user@MX11# set routing-instances EVPN-VXLAN-1 interface lt-5/1/0.1
user@MX11# set routing-instances EVPN-VXLAN-1 route-distinguisher 192.0.2.21:1
user@MX11# set routing-instances EVPN-VXLAN-1 vrf-target target:1:1
user@MX11# set routing-instances EVPN-VXLAN-1 protocols evpn traceoptions file MX11-EVPN-
VXLAN-1.log
user@MX11# set routing-instances EVPN-VXLAN-1 protocols evpn traceoptions file size 10m
user@MX11# set routing-instances EVPN-VXLAN-1 protocols evpn traceoptions flag all
user@MX11# set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
user@MX11# set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5
```

```

user@MX11# set routing-instances EVPN-VXLAN-1 protocols evpn default-gateway no-gateway-
community
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 routing-interface irb.1
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 routing-interface irb.2
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 routing-interface irb.3
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 routing-interface irb.4
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 routing-interface irb.5
user@MX11# set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
user@MX11# set routing-instances VRF instance-type vrf
user@MX11# set routing-instances VRF interface irb.1
user@MX11# set routing-instances VRF interface irb.2
user@MX11# set routing-instances VRF interface irb.3
user@MX11# set routing-instances VRF interface

```

Configuring WAN Router (P)

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*.

Configure an MX Series router as the WAN edge router and name it as P:

1. Set the system hostname.

```
[edit]
user@P# set system host-name P
```

2. Configure the interfaces on the P router (WAN) to interconnect different data center networks running Ethernet VPN (EVPN) with Virtual extensible LAN (VXLAN) encapsulation through a WAN running MPLS-based EVPN.

```
[edit]
user@P# set interfaces ge-0/0/3 unit 0 description "CONNECTED TO MX11"
user@P# set interfaces ge-0/0/3 unit 0 family inet address 203.0.113.42/24
user@P# set interfaces ge-0/0/3 unit 0 family mpls
user@P# set interfaces ge-0/0/5 unit 0 description "CONNECTED TO MX12"
user@P# set interfaces ge-0/0/5 unit 0 family inet address 203.0.113.12/24
user@P# set interfaces ge-0/0/5 unit 0 family mpls
user@P# set interfaces ge-0/0/6 unit 0 description "CONNECTED TO MX21"
user@P# set interfaces ge-0/0/6 unit 0 family inet address 203.0.113.32/24
user@P# set interfaces ge-0/0/6 unit 0 family mpls
user@P# set interfaces ge-0/0/7 unit 0 description "CONNECTED TO MX22"
user@P# set interfaces ge-0/0/7 unit 0 family inet address 203.0.113.52/24
user@P# set interfaces ge-0/0/7 unit 0 family mpls
```

3. Configure the community.

```
[edit]
user@P# set policy-options community RT-CORE members target:1:2
user@P# set policy-options community RT-DC1 members target:1:1
user@P# set policy-options community RT-DC2 members target:1:3
```

4. Configure policies.

```
[edit]
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 1 from protocol
bgp
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 1 from
community RT-CORE
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 1 then accept
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 2 from protocol
```

```

bgp
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 2 from
community RT-DC1
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 2 then reject
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 3 from protocol
bgp
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 3 from
community RT-DC2
user@P# set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 3 then reject

```

5. Configure the loopback interface address and routing options.

```

[edit]
user@P# interfaces lo0 unit 86 family inet address 203.0.113.1/32
user@P# interfaces lo0 unit 86 family mpls
user@P# routing-options router-id 203.0.113.1
user@P# routing-options autonomous-system 65300

```

6. Enable MPLS, BGP, and OSPF protocols on the core interfaces. Create MPLS LSPs between P and other gateway and WAN edge routers (MX11, MX12, P, MX22).

```

[edit]
user@P# set protocols mpls label-switched-path P-T0-MX11 from 203.0.113.1
user@P# set protocols mpls label-switched-path P-T0-MX11 to 192.0.2.21
user@P# set protocols mpls label-switched-path P-T0-MX12 to 192.0.2.22
user@P# set protocols mpls label-switched-path P-T0-MX21 to 198.51.100.21
user@P# set protocols mpls label-switched-path P-T0-MX22 to 198.51.100.22
user@P# set protocols mpls interface all
user@P# set protocols bgp group INT type internal
user@P# set protocols bgp group INT import BLOCK-VXLAN-ROUTES-FROM-CORE
user@P# set protocols bgp group INT family evpn signaling
user@P# set protocols bgp group INT cluster 203.0.113.1
user@P# set protocols bgp group INT neighbor 192.0.2.21
user@P# set protocols bgp group INT neighbor 192.0.2.22
user@P# set protocols bgp group INT neighbor 198.51.100.21
user@P# set protocols bgp group INT neighbor 198.51.100.22
user@P# set protocols bgp local-address 203.0.113.1
user@P# set protocols bgp local-as 65300
user@P# set protocols ospf traffic-engineering

```

```
user@P# set protocols ospf area 0.0.0.0 interface all
user@P# set protocols ospf area 0.0.0.0 interface lo0.86
```

Configuring CE1

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*.

Configure the MX router CE1:



NOTE: Follow similar steps for CE2, CE3, CE4, CE5, and CE6. Refer to [Figure 187 on page 1773](#) for interface name, IP address, and connectivity.

1. Set the system hostname.

```
[edit]
user@CE1# set system host-name CE1
```

2. Configure the interfaces on the CE1 device to connect to the host and TOR12. Enables forwarding traffic between host and TOR12.

```
[edit]
user@CE1# set interfaces ge-0/0/8 unit 0 description "CONNECTED TO Host 1"
user@CE1# set interfaces ge-0/0/8 unit 0 family bridge interface-mode trunk
user@CE1# set interfaces ge-0/0/8 unit 0 family bridge vlan-id-list 1-5
user@CE1# set interfaces ae1 unit 0 description "CONNECTED TO ToR12"
user@CE1# set interfaces ae1 unit 0 family bridge interface-mode trunk
user@CE1# set interfaces ae1 unit 0 family bridge vlan-id-list 1-5
```

3. Define bridge domains and associate it with a VLAN ID.

```
[edit]
user@CE1# set bridge-domains BD-1 domain-type bridge
user@CE1# set bridge-domains BD-1 vlan-id 1
user@CE1# set bridge-domains BD-2 domain-type bridge
user@CE1# set bridge-domains BD-2 vlan-id 2
```

```

user@CE1# set bridge-domains BD-3 domain-type bridge
user@CE1# set bridge-domains BD-3 vlan-id 3
user@CE1# set bridge-domains BD-4 domain-type bridge
user@CE1# set bridge-domains BD-4 vlan-id 4
user@CE1# set bridge-domains BD-5 domain-type bridge
user@CE1# set bridge-domains BD-5 vlan-id 5

```

Verification

IN THIS SECTION

- [Verify EVPN-VXLAN and EVPN-MPLS EVI stitching using Logical Tunnel interface | 1790](#)
- [Verify flooding in bridge domains | 1792](#)
- [Verify MAC learning | 1794](#)
- [Verify MAC Address Forwarding Table | 1795](#)

After you configure both the underlay and EVPN overlay we recommend that you verify that the configurations work as you intended.

Verify EVPN-VXLAN and EVPN-MPLS EVI stitching using Logical Tunnel interface

Purpose

Confirm the EVPN-VXLAN and EVPN-MPLS EVPN instances (EVIs) are configured for stitching using logical tunnel interfaces.

Action

On any gateway device, issue the `show routing-instances evpn-vxlan-instance` and the `show routing-instances evpn-mpls-instance` commands to see the logical tunnel interfaces assigned to the respective EVIs. Issue the `show interfaces lt-interface` command to verify the EVIs are stitched using the assigned LT interface.



NOTE: The output is truncated for brevity

```

user@MX11> show routing-instances EVPN-VXLAN-1
...

```

```
interface lt-5/1/0.1;
route-distinguisher 192.0.2.21:1;
vrf-target target:1:1;
```

```
user@MX11> show routing-instances EVPN-MPLS-1
...
interface lt-5/1/0.0;
route-distinguisher 192.0.2.21:100;
vrf-target target:1:2;
```

```
user@MX11> show interfaces lt-5/1/0
esi {
    00:22:22:22:22:22:22:22:22:22;
    all-active;
}
unit 0 {
    description T0-EVPN-VXLAN;
    encapsulation ethernet-bridge;
    peer-unit 1;
    family bridge {
        interface-mode trunk;
        vlan-id-list 1-5;
    }
}
unit 1 {
    description T0-EVPN-MPLS;
    encapsulation ethernet-bridge;
    peer-unit 0;
    family bridge {
        interface-mode trunk;
        vlan-id-list 1-5;
    }
}
```

Meaning

The EVPN-VXLAN-1 and the EVPN-MPLS-1 EVIs are stitched using lt-5/1/0 interface.

Verify flooding in bridge domains

Purpose

Verify bridge domain flood information for both EVPN-MPLS and EVPN-VXLAN instances any gateway device.

Action

On any gateway device (MX11), issue the show bridge flood extensive instance *instance-name* command.



NOTE: The output is truncated for brevity

```
user@MX11> show bridge flood extensive instance EVPN-MPLS-1
```

```
Name: EVPN-MPLS-1
```

```
CEs: 1
```

```
VEs: 15
```

```
Bridging domain: BD-1
```

```
  EVPN extended: Yes
```

```
  Flood route prefix: 0x3001a/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: 779
```

```
    Flooding to:
```

Name	Type	NhType	Index
__all_ces__	Group	comp	759

```
      Composition: split-horizon
```

```
      Flooding to:
```

Name	Type	NhType	Index
lt-5/1/0.0	MH-CE	ucst	736

```
      ESI: 00:22:22:22:22:22:22:22:22:22
```

```
Flood route prefix: 0x30010/51
```

```
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: 812
Flooding to:
  Name          Type      NhType      Index
  __all_ces__   Group    comp        759
    Composition: split-horizon
    Flooding to:
      Name          Type      NhType      Index
      1t-5/1/0.0   MH-CE    ucst        736
      ESI: 00:22:22:22:22:22:22:22:22:22

Flooding to:
  Name          Type      NhType      Index
  __ves__       Group    comp        763
    Composition: flood-to-all
Component flood-nh(s) (for flooding to EVPN core):
  Index      Peer          NH-Type
  891        192.0.2.22      comp (IM/SH)
  892        198.51.100.21   comp (IM/SH)
  746        198.51.100.22   comp (IM/SH)

```

```

user@MX11> show bridge flood extensive instance EVPN-VXLAN-1Name: EVPN-VXLAN-1
Name: EVPN-VXLAN-1
CEs: 1
VEs: 3
Bridging domain: BD-1
Flood route prefix: 0x3000f/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: 845
Flooding to:
  Name          Type      NhType      Index
  __all_ces__   Group    comp        797
    Composition: split-horizon
    Flooding to:
      Name          Type      NhType      Index
      1t-5/1/0.0   CE        ucst        751

Flood route prefix: 0x30015/51

```

```

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: 842
  Flooding to:
    Name          Type          NhType          Index
    __all_ces__   Group          comp          797
    Composition: split-horizon
    Flooding to:
      Name          Type          NhType          Index
      1t-5/1/0.0    CE          ucst          751
    Flooding to:
      Name          Type          NhType          Index
      __ves__       Group          comp          700
      Composition: flood-to-all
      Flooding to:
        Name          Type          NhType          Index          RVTEP-IP
        vtep.32769    CORE_FACING    venh          687          192.0.2.11
        vtep.32770    CORE_FACING    venh          729          192.0.2.12
        vtep.32771    CORE_FACING    venh          748          192.0.2.22

```

Meaning

The output shows that EVPN-MPLS-1 is actively participating in BUM traffic flooding for the associated bridge domains. It uses composite next-hop groups to flood traffic to other gateway devices in the MPLS core, which is MX12 (192.0.2.22), MX21 (198.51.100.21), and MX22 (198.51.100.22). Meanwhile, the EVPN-MPLS instance EVPN-MPLS-1 has bridge domains configured for EVPN extension with active endpoints to neighboring gateway devices.

Verify MAC learning

Purpose

Verify the MAC learning between gateway devices.

Action

On any leaf device (TOR11), issue the `show evpn database instance routing-instance-name l2-domain-id 1` command to verify the learned MAC addresses.

```
user@TOR11> show evpn database instance EVPN-VXLAN-1 l2-domain-id 1
Instance: EVPN-VXLAN-1
VLAN  DomainId  MAC address      Active source      Timestamp      IP address
  1      00:00:5e:00:01:01  00:11:aa:aa:aa:aa:aa:aa:aa  Jul 30 01:02:11  10.11.1.10
  1      2c:6b:f5:c2:ff:f0  DRP                    vtep.32769        Jul 30 01:02:11
192.0.2.21
```

Meaning

The host MAC/IP binding is learned from a remote VTEP (192.0.2.21 which is MX11), showing active host participation in the EVPN fabric.

Verify MAC Address Forwarding Table

Purpose

Verify MAC learning and flooding behavior in the bridging environment.

Action

On any gateway device, issue the `show bridge mac-table mac-address instance evpn-instance`.

```
user@MX11> show bridge mac-table 2c:6b:f5:c2:ff:f0 instance EVPN-VXLAN-1
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 : EVPN-VXLAN-1
Bridging domain : BD-1, VLAN : 1
MAC              MAC              GBP      Logical      Active
```

address	flags	TAG	interface	source
2c:6b:f5:c2:ff:f0	DR		lt-5/1/0.1	192.0.2.22

```
user@MX11> show bridge mac-table 2c:6b:f5:c2:ff:f0 instance EVPN-MPLS-1
MAC flags      (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
               0 -OVSDB MAC, SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P -Pinned
               MAC)
```

Routing instance : EVPN-MPLS-1

Bridging domain : **BD-1, VLAN : 1**

MAC	MAC	GBP	Logical	NH	RTR
address	flags	TAG	interface	Index	ID
2c:6b:f5:c2:ff:f0	DC			1048581	1048581

Meaning

For the EVPN-VXLAN-1 instance, the MAC address **2c:6b:f5:c2:ff:f0** is dynamically learned (D flag) and associated with Bridge Domain **BD-1 (VLAN 1)** in the EVPN instance EVPN-VXLAN-1. It was learned from a **lt-5/1/0.1** logical tunnel interface, with the remote source IP **192.0.2.22 (MX12)**, indicating that this MAC belongs to a host connected to a remote device in the EVPN fabric.

For the EVPN-MPLS-1 instance, the MAC address **2c:6b:f5:c2:ff:f0** is present in Bridge Domain **BD-1 (VLAN 1)** under the EVPN instance EVPN-MPLS-1. It is marked with flags **D (Dynamically learned)** and **C (Control MAC)**, indicating that it was learned via the EVPN control plane rather than data plane snooping. The MAC is associated with a next-hop index **1048581**, which maps to a remote router, confirming that this MAC belongs to a host reachable via EVPN/MPLS from a remote site.

Appendix 1: Set Commands on All Devices

Set command output on all devices.

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.

ToR11

```
set system host-name ToR11
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 unit 0 description "CONNECTED TO CE-2"
set interfaces ge-0/0/0 unit 0 family bridge interface-mode trunk
```

```

set interfaces ge-0/0/0 unit 0 family bridge vlan-id-list 1-5
set interfaces ge-0/0/2 description "CONNECTED TO CE-1"
set interfaces ge-0/0/2 gigether-options 802.3ad ae0
set interfaces ge-0/0/3 unit 0 description "CONNECTED TO ToR12"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.2.1/24
set interfaces ge-0/0/4 unit 0 description "CONNECTED TO MX-11"
set interfaces ge-0/0/4 unit 0 family inet address 192.168.3.1/24
set interfaces ge-0/0/5 unit 0 description "CONNECTED TO MX-12"
set interfaces ge-0/0/5 unit 0 family inet address 192.168.4.1/24
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 11:11:11:11:11:11
set interfaces ae0 unit 0 family bridge interface-mode trunk
set interfaces ae0 unit 0 family bridge vlan-id-list 1-5
set interfaces lo0 unit 0 family inet address 192.0.2.11/32
set policy-options policy-statement L0 term 1 from protocol direct
set policy-options policy-statement L0 term 1 from route-filter 192.0.2.11/32 exact
set policy-options policy-statement L0 term 1 then accept
set policy-options policy-statement TEST then community add NO-EXPORT
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set policy-options community NO-EXPORT members no-advertise
set policy-options community NO-EXPORT members no-export
set policy-options community NO-EXPORT members no-export-subconfed
set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-VXLAN-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5
set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 routing-interface irb.1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 routing-interface irb.2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 routing-interface irb.3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3

```

```

set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 routing-interface irb.4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 routing-interface irb.5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
set routing-instances EVPN-VXLAN-1 interface ge-0/0/0.0
set routing-instances EVPN-VXLAN-1 interface ae0.0
set routing-instances EVPN-VXLAN-1 route-distinguisher 192.0.2.21:1
set routing-instances EVPN-VXLAN-1 vrf-target target:1:1
set routing-instances VRF instance-type vrf
set routing-instances VRF interface irb.1
set routing-instances VRF interface irb.2
set routing-instances VRF interface irb.3
set routing-instances VRF interface irb.4
set routing-instances VRF interface irb.5
set routing-instances VRF route-distinguisher 1:1
set routing-instances VRF vrf-target target:10:10
set routing-options router-id 192.0.2.11
set routing-options autonomous-system 65100
set routing-options forwarding-table export evpn-pplb

```

ToR12

```

set system host-name ToR12

set chassis aggregated-devices ethernet device-count 2
set interfaces ge-0/0/0 description "CONNECTED TO ToR11"
set interfaces ge-0/0/0 gigether-options 802.3ad ae1
set interfaces ge-0/0/1 unit 0 description "CONNECTED TO ToR11"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.2.2/24
set interfaces ge-0/0/2 description "CONNECTED TO CE-1"
set interfaces ge-0/0/2 gigether-options 802.3ad ae0
set interfaces ge-0/0/3 description "CONNECTED TO ToR12"
set interfaces ge-0/0/3 gigether-options 802.3ad ae1
set interfaces ge-0/0/4 unit 0 description "CONNECTED TO CE-3"
set interfaces ge-0/0/4 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/4 unit 0 family bridge vlan-id-list 1-5
set interfaces ge-0/0/6 unit 0 description "CONNECTED TO MX11"

```

```

set interfaces ge-0/0/6 unit 0 family inet address 192.168.6.1/24
set interfaces ge-0/0/7 unit 0 description "CONNECTED TO MX12"
set interfaces ge-0/0/7 unit 0 family inet address 192.168.5.1/24
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp system-id 11:11:11:11:11:11
set interfaces ae0 unit 0 family bridge interface-mode trunk
set interfaces ae0 unit 0 family bridge vlan-id-list 1-5
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces lo0 unit 0 family inet address 192.0.2.12/32
set policy-options policy-statement L0 term 1 from protocol direct
set policy-options policy-statement L0 term 1 from route-filter 192.0.2.12/32 exact
set policy-options policy-statement L0 term 1 then accept
set policy-options policy-statement TEST then community add NO-EXPORT
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set policy-options community NO-EXPORT members no-advertise
set policy-options community NO-EXPORT members no-export
set policy-options community NO-EXPORT members no-export-subconfed
set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5
set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
set routing-instances EVPN-VXLAN-1 interface ge-0/0/4.0
set routing-instances EVPN-VXLAN-1 interface ae0.0
set routing-instances EVPN-VXLAN-1 route-distinguisher 192.0.2.12:1
set routing-instances EVPN-VXLAN-1 vrf-target target:1:1

```



```
set routing-options router-id 192.0.2.12
set routing-options autonomous-system 65200
set routing-options forwarding-table export evpn-pplb
set protocols bgp group MX11 type external
set protocols bgp group MX11 local-address 192.168.6.1
set protocols bgp group MX11 export L0
set protocols bgp group MX11 export TEST
set protocols bgp group MX11 peer-as 65400
set protocols bgp group MX11 local-as 65200
set protocols bgp group MX11 neighbor 192.168.6.2 family inet unicast
set protocols bgp group MX12 type external
set protocols bgp group MX12 local-address 192.168.5.1
set protocols bgp group MX12 export L0
set protocols bgp group MX12 export TEST
set protocols bgp group MX12 peer-as 65500
set protocols bgp group MX12 local-as 65200
set protocols bgp group MX12 neighbor 192.168.5.2 family inet unicast
set protocols bgp group ToR11 type external
set protocols bgp group ToR11 local-address 192.168.2.2
set protocols bgp group ToR11 export L0
set protocols bgp group ToR11 export TEST
set protocols bgp group ToR11 peer-as 65100
set protocols bgp group ToR11 local-as 65200
set protocols bgp group ToR11 neighbor 192.168.2.1 family inet unicast
set protocols bgp group MX11-EVPN type external
set protocols bgp group MX11-EVPN multihop ttl 2
set protocols bgp group MX11-EVPN multihop no-nexthop-change
set protocols bgp group MX11-EVPN local-address 192.0.2.12
set protocols bgp group MX11-EVPN export TEST
set protocols bgp group MX11-EVPN peer-as 65400
set protocols bgp group MX11-EVPN local-as 65200
set protocols bgp group MX11-EVPN neighbor 192.0.2.21 family evpn signaling
set protocols bgp group MX12-EVPN type external
set protocols bgp group MX12-EVPN multihop ttl 2
set protocols bgp group MX12-EVPN multihop no-nexthop-change
set protocols bgp group MX12-EVPN local-address 192.0.2.12
set protocols bgp group MX12-EVPN export TEST
set protocols bgp group MX12-EVPN peer-as 65500
set protocols bgp group MX12-EVPN local-as 65200
set protocols bgp group MX12-EVPN neighbor 192.0.2.22 family evpn signaling
set protocols bgp group ToR11-EVPN type external
set protocols bgp group ToR11-EVPN multihop ttl 2
set protocols bgp group ToR11-EVPN multihop no-nexthop-change
```

```

set protocols bgp group ToR11-EVPN local-address 192.0.2.12
set protocols bgp group ToR11-EVPN export TEST
set protocols bgp group ToR11-EVPN peer-as 65100
set protocols bgp group ToR11-EVPN local-as 65200
set protocols bgp group ToR11-EVPN neighbor 192.0.2.11 family evpn signaling
set protocols bgp group ToR12-EVPN export TEST
set protocols bgp local-as 65200

```

Data Center Gateway and WAN Edge 1 Router (MX11)

```

set system host-name MX11
set interfaces ge-0/0/0 unit 0 description "CONNECTED TO ToR11"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.3.2/24
set interfaces ge-0/0/1 unit 0 description "CONNECTED TO ToR12"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.6.2/24
set interfaces ge-0/0/2 unit 0 description "CONNECTED TO MX12"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.7.1/24
set interfaces ge-0/0/3 unit 0 description "CONNECTED TO P"
set interfaces ge-0/0/3 unit 0 family inet address 203.0.113.41/24
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces lt-5/1/0 esi 00:22:22:22:22:22:22:22:22:22
set interfaces lt-5/1/0 esi 00:22:22:22:22:22:22:22:22:22
set interfaces lt-5/1/0 esi all-active
set interfaces lt-5/1/0 unit 0 description TO-EVPN-VXLAN
set interfaces lt-5/1/0 unit 0 encapsulation ethernet-bridge
set interfaces lt-5/1/0 unit 0 peer-unit 1
set interfaces lt-5/1/0 unit 0 family bridge interface-mode trunk
set interfaces lt-5/1/0 unit 0 family bridge vlan-id-list 1-5
set interfaces lt-5/1/0 unit 1 description TO-EVPN-MPLS
set interfaces lt-5/1/0 unit 1 encapsulation ethernet-bridge
set interfaces lt-5/1/0 unit 1 peer-unit 0
set interfaces lt-5/1/0 unit 1 family bridge interface-mode trunk
set interfaces lt-5/1/0 unit 1 family bridge vlan-id-list 1-5set interfaces irb unit 1 proxy-
macip-advertisement
set interfaces irb unit 1 virtual-gateway-esi 00:11:aa:aa:aa:aa:aa:aa:aa:aa
set interfaces irb unit 1 virtual-gateway-esi all-active
set interfaces irb unit 1 family inet address 10.11.1.12/24 virtual-gateway-address 10.11.1.10
set interfaces irb unit 2 proxy-macip-advertisement
set interfaces irb unit 2 virtual-gateway-esi 00:11:bb:bb:bb:bb:bb:bb:bb:bb
set interfaces irb unit 2 virtual-gateway-esi all-active
set interfaces irb unit 2 family inet address 10.12.1.12/24 virtual-gateway-address 10.12.1.10

```

```

set interfaces irb unit 3 proxy-macip-advertisement
set interfaces irb unit 3 virtual-gateway-esi 00:11:cc:cc:cc:cc:cc:cc:cc
set interfaces irb unit 3 virtual-gateway-esi all-active
set interfaces irb unit 3 family inet address 10.13.1.12/24 virtual-gateway-address 10.13.1.10
set interfaces irb unit 4 proxy-macip-advertisement
set interfaces irb unit 4 virtual-gateway-esi 00:11:dd:dd:dd:dd:dd:dd:dd
set interfaces irb unit 4 virtual-gateway-esi all-active
set interfaces irb unit 4 family inet address 10.14.1.12/24 virtual-gateway-address 10.14.1.10
set interfaces irb unit 5 proxy-macip-advertisement
set interfaces irb unit 5 virtual-gateway-esi 00:11:ee:ee:ee:ee:ee:ee:ee
set interfaces irb unit 5 virtual-gateway-esi all-active
set interfaces irb unit 5 family inet address 10.15.1.12/24 virtual-gateway-address 10.15.1.10
set interfaces lo0 unit 0 family inet address 192.0.2.21/32
set interfaces lo0 unit 0 family mpls
set policy-options policy-statement L0 from protocol direct
set policy-options policy-statement L0 from route-filter 192.0.2.21/32 exact
set policy-options policy-statement L0 then accept
set policy-options policy-statement TEST then community add NO-EXPORT
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set policy-options community NO-EXPORT members no-advertise
set policy-options community NO-EXPORT members no-export
set policy-options community NO-EXPORT members no-export-subconfed
set routing-instances EVPN-MPLS-1 instance-type virtual-switch
set routing-instances EVPN-MPLS-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-MPLS-1 protocols evpn extended-vlan-list 1-5
set routing-instances EVPN-MPLS-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-MPLS-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-MPLS-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-MPLS-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-MPLS-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-MPLS-1 interface lt-5/1/0.0
set routing-instances EVPN-MPLS-1 route-distinguisher 192.0.2.21:100
set routing-instances EVPN-MPLS-1 vrf-target target:1:2
set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-VXLAN-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5

```

```

set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 routing-interface irb.1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 routing-interface irb.2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 routing-interface irb.3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 routing-interface irb.4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 routing-interface irb.5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
set routing-instances EVPN-VXLAN-1 interface lt-5/1/0.1
set routing-instances EVPN-VXLAN-1 route-distinguisher 192.0.2.21:1
set routing-instances EVPN-VXLAN-1 vrf-target target:1:1
set routing-instances VRF instance-type vrf
set routing-instances VRF interface irb.1
set routing-instances VRF interface irb.2
set routing-instances VRF interface irb.3
set routing-instances VRF interface
set protocols bgp group INT type internal
set protocols bgp group INT local-address 192.0.2.21
set protocols bgp group INT family evpn signaling
set protocols bgp group INT export TEST
set protocols bgp group INT neighbor 203.0.113.1
set protocols bgp group MX12 type external
set protocols bgp group MX12 local-address 192.168.7.1
set protocols bgp group MX12 export TEST
set protocols bgp group MX12 export L0
set protocols bgp group MX12 peer-as 65500
set protocols bgp group MX12 local-as 65400
set protocols bgp group MX12 neighbor 192.168.7.2 family inet unicast
set protocols bgp group ToR11 type external
set protocols bgp group ToR11 local-address 192.168.3.2

```

```
set protocols bgp group ToR11 import TEST
set protocols bgp group ToR11 export TEST
set protocols bgp group ToR11 export L0
set protocols bgp group ToR11 peer-as 65100
set protocols bgp group ToR11 local-as 65400
set protocols bgp group ToR11 neighbor 192.168.3.1 family inet unicast
set protocols bgp group ToR12 type external
set protocols bgp group ToR12 local-address 192.168.6.2
set protocols bgp group ToR12 export TEST
set protocols bgp group ToR12 export L0
set protocols bgp group ToR12 peer-as 65200
set protocols bgp group ToR12 local-as 65400
set protocols bgp group ToR12 neighbor 192.168.6.1 family inet unicast
set protocols bgp group MX12-EVPN type external
set protocols bgp group MX12-EVPN multihop ttl 2
set protocols bgp group MX12-EVPN multihop no-nexthop-change
set protocols bgp group MX12-EVPN local-address 192.0.2.21
set protocols bgp group MX12-EVPN export TEST
set protocols bgp group MX12-EVPN peer-as 65500
set protocols bgp group MX12-EVPN local-as 65400
set protocols bgp group MX12-EVPN neighbor 192.0.2.22 family evpn signaling
set protocols bgp group ToR11-EVPN type external
set protocols bgp group ToR11-EVPN multihop ttl 2
set protocols bgp group ToR11-EVPN multihop no-nexthop-change
set protocols bgp group ToR11-EVPN local-address 192.0.2.21
set protocols bgp group ToR11-EVPN export TEST
set protocols bgp group ToR11-EVPN peer-as 65100
set protocols bgp group ToR11-EVPN local-as 65400
set protocols bgp group ToR11-EVPN neighbor 192.0.2.11 family evpn signaling
set protocols bgp group ToR12-EVPN type external
set protocols bgp group ToR12-EVPN multihop ttl 2
set protocols bgp group ToR12-EVPN multihop no-nexthop-change
set protocols bgp group ToR12-EVPN local-address 192.0.2.21
set protocols bgp group ToR12-EVPN export TEST
set protocols bgp group ToR12-EVPN peer-as 65200
set protocols bgp group ToR12-EVPN local-as 65400
set protocols bgp group ToR12-EVPN neighbor 192.0.2.12 family evpn signaling
set protocols bgp local-address 192.0.2.21
set protocols bgp local-as 65300
set protocols mpls label-switched-path MX11-T0-MX12 to 192.0.2.22
set protocols mpls label-switched-path MX11-T0-P to 203.0.113.1
set protocols mpls label-switched-path MX11-T0-MX21 to 198.51.100.21
set protocols mpls label-switched-path MX11-T0-MX22 to 198.51.100.22
```

```

set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-0/0/3.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable

```

Data Center Gateway and WAN Edge 2 Router (MX12)

```

set system host-name MX12
set chassis fpc 1 pic 0 tunnel-services
set interfaces ge-0/0/0 unit 0 description "CONNECTED TO ToR11"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.4.2/24
set interfaces ge-0/0/1 unit 0 description "CONNECTED TO ToR12"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.5.2/24
set interfaces ge-0/0/2 unit 0 description "CONNECTED TO MX11"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.7.2/24
set interfaces ge-0/0/4 unit 0 description "CONNECTED TO P"
set interfaces ge-0/0/4 unit 0 family inet address 203.0.113.11/24
set interfaces ge-0/0/4 unit 0 family mpls
set interfaces lt-1/0/0 esi 00:22:22:22:22:22:22:22:22:22
set interfaces lt-1/0/0 esi all-active
set interfaces lt-1/0/0 unit 0 description TO-EVPN-VXLAN
set interfaces lt-1/0/0 unit 0 encapsulation ethernet-bridge
set interfaces lt-1/0/0 unit 0 peer-unit 1
set interfaces lt-1/0/0 unit 0 family bridge interface-mode trunk
set interfaces lt-1/0/0 unit 0 family bridge vlan-id-list 1-5
set interfaces lt-1/0/0 unit 1 description TO-EVPN-MPLS
set interfaces lt-1/0/0 unit 1 encapsulation ethernet-bridge
set interfaces lt-1/0/0 unit 1 peer-unit 0
set interfaces lt-1/0/0 unit 1 family bridge interface-mode trunk
set interfaces lt-1/0/0 unit 1 family bridge vlan-id-list 1-5
set interfaces irb unit 1 proxy-macip-advertisement
set interfaces irb unit 1 virtual-gateway-esi 00:11:aa:aa:aa:aa:aa:aa:aa:aa
set interfaces irb unit 1 virtual-gateway-esi all-active
set interfaces irb unit 1 family inet address 10.11.1.13/24 virtual-gateway-address 10.11.1.10
set interfaces irb unit 2 proxy-macip-advertisement
set interfaces irb unit 2 virtual-gateway-esi 00:11:bb:bb:bb:bb:bb:bb:bb:bb
set interfaces irb unit 2 virtual-gateway-esi all-active
set interfaces irb unit 2 family inet address 10.12.1.13/24 virtual-gateway-address 10.12.1.10

```

```

set interfaces irb unit 3 proxy-macip-advertisement
set interfaces irb unit 3 virtual-gateway-esi 00:11:cc:cc:cc:cc:cc:cc:cc
set interfaces irb unit 3 virtual-gateway-esi all-active
set interfaces irb unit 3 family inet address 10.13.1.13/24 virtual-gateway-address 10.13.1.10
set interfaces irb unit 4 proxy-macip-advertisement
set interfaces irb unit 4 virtual-gateway-esi 00:11:dd:dd:dd:dd:dd:dd:dd
set interfaces irb unit 4 virtual-gateway-esi all-active
set interfaces irb unit 4 family inet address 10.14.1.13/24 virtual-gateway-address 10.14.1.10
set interfaces irb unit 5 proxy-macip-advertisement
set interfaces irb unit 5 virtual-gateway-esi 00:11:ee:ee:ee:ee:ee:ee:ee
set interfaces irb unit 5 virtual-gateway-esi all-active
set interfaces irb unit 5 family inet address 10.15.1.13/24 virtual-gateway-address 10.15.1.10
set interfaces lo0 unit 0 family inet address 192.0.2.22/32
set interfaces lo0 unit 0 family mpls
set policy-options policy-statement L0 from protocol direct
set policy-options policy-statement L0 from route-filter 192.0.2.22/32 exact
set policy-options policy-statement L0 then accept
set policy-options policy-statement TEST from protocol bgp
set policy-options policy-statement TEST
set routing-instances EVPN-MPLS-1 instance-type virtual-switch
set routing-instances EVPN-MPLS-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-MPLS-1 protocols evpn extended-vlan-list 1-5
set routing-instances EVPN-MPLS-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-MPLS-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-MPLS-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-MPLS-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-MPLS-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-MPLS-1 interface lt-1/0/0.0
set routing-instances EVPN-MPLS-1 route-distinguisher 192.0.2.22:100
set routing-instances EVPN-MPLS-1 vrf-target target:1:2
set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-VXLAN-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5
set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 routing-interface irb.1

```

```

set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 routing-interface irb.2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 routing-interface irb.3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 routing-interface irb.4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 routing-interface irb.5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
set routing-instances EVPN-VXLAN-1 interface lt-1/0/0.1
set routing-instances EVPN-VXLAN-1 route-distinguisher 192.0.2.22:1
set routing-instances EVPN-VXLAN-1 vrf-target target:1:1
set routing-instances VRF instance-type vrf
set routing-instances VRF interface irb.1
set routing-instances VRF interface irb.2
set routing-instances VRF interface irb.3
set routing-instances VRF interface irb.4
set routing-instances VRF interface irb.5
set routing-instances VRF route-distinguisher 1:1
set routing-instances VRF vrf-target target:10:10
set routing-options router-id 192.0.2.22
set routing-options autonomous-system 65300
set routing-options forwarding-table export evpn-pplb
set protocols bgp group INT type internal
set protocols bgp group INT family evpn signaling
set protocols bgp group INT export TEST
set protocols bgp group INT neighbor 203.0.113.1
set protocols bgp group MX11 type external
set protocols bgp group MX11 local-address 192.168.7.2
set protocols bgp group MX11 export TEST
set protocols bgp group MX11 export L0
set protocols bgp group MX11 peer-as 65400
set protocols bgp group MX11 local-as 65500
set protocols bgp group MX11 neighbor 192.168.7.1 family inet unicast
set protocols bgp group ToR11 type external

```



```
set protocols bgp group ToR11 local-address 192.168.4.2
set protocols bgp group ToR11 export TEST
set protocols bgp group ToR11 export L0
set protocols bgp group ToR11 peer-as 65100
set protocols bgp group ToR11 local-as 65500
set protocols bgp group ToR11 neighbor 192.168.4.1 family inet unicast
set protocols bgp group ToR12 type external
set protocols bgp group ToR12 local-address 192.168.5.2
set protocols bgp group ToR12 export TEST
set protocols bgp group ToR12 export L0
set protocols bgp group ToR12 peer-as 65200
set protocols bgp group ToR12 local-as 65500
set protocols bgp group ToR12 neighbor 192.168.5.1 family inet unicast
set protocols bgp group MX11-EVPN type external
set protocols bgp group MX11-EVPN multihop ttl 2
set protocols bgp group MX11-EVPN multihop no-nexthop-change
set protocols bgp group MX11-EVPN local-address 192.0.2.22
set protocols bgp group MX11-EVPN export TEST
set protocols bgp group MX11-EVPN peer-as 65400
set protocols bgp group MX11-EVPN local-as 65500
set protocols bgp group MX11-EVPN neighbor 192.0.2.21 family evpn signaling
set protocols bgp group ToR11-EVPN type external
set protocols bgp group ToR11-EVPN multihop ttl 2
set protocols bgp group ToR11-EVPN multihop no-nexthop-change
set protocols bgp group ToR11-EVPN local-address 192.0.2.22
set protocols bgp group ToR11-EVPN export TEST
set protocols bgp group ToR11-EVPN peer-as 65100
set protocols bgp group ToR11-EVPN local-as 65500
set protocols bgp group ToR11-EVPN neighbor 192.0.2.11 family evpn signaling
set protocols bgp group ToR12-EVPN type external
set protocols bgp group ToR12-EVPN multihop ttl 2
set protocols bgp group ToR12-EVPN multihop no-nexthop-change
set protocols bgp group ToR12-EVPN local-address 192.0.2.22
set protocols bgp group ToR12-EVPN export TEST
set protocols bgp group ToR12-EVPN peer-as 65200
set protocols bgp group ToR12-EVPN local-as 65500
set protocols bgp group ToR12-EVPN neighbor 192.0.2.12 family evpn signaling
set protocols bgp local-address 192.0.2.22
set protocols bgp local-as 65300
set protocols mpls label-switched-path MX12-T0-MX11 to 192.0.2.21
set protocols mpls label-switched-path MX12-T0-P to 203.0.113.1
set protocols mpls label-switched-path MX12-T0-MX21 to 198.51.100.21
set protocols mpls label-switched-path MX12-T0-MX22 to 198.51.100.22
```

```

set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-0

```

Data Center Gateway and WAN Edge 3 Router (MX21)

```

set system host-name MX21
set interfaces ge-0/0/0 unit 0 description "CONNECTED TO P"
set interfaces ge-0/0/0 unit 0 family inet address 203.0.113.31/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 unit 0 description "CONNECTED TO MX22"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.13.1/24
set interfaces ge-0/0/2 unit 0 description "CONNECTED TO ToR21"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.9.1/24
set interfaces ge-0/0/3 unit 0 description "CONNECTED TO ToR22"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.8.1/24
set interfaces lt-5/0/0 esi 00:33:33:33:33:33:33:33:33:33
set interfaces lt-5/0/0 esi all-active
set interfaces lt-5/0/0 unit 0 description TO-EVPN-VXLAN
set interfaces lt-5/0/0 unit 0 encapsulation ethernet-bridge
set interfaces lt-5/0/0 unit 0 peer-unit 1
set interfaces lt-5/0/0 unit 0 family bridge interface-mode trunk
set interfaces lt-5/0/0 unit 0 family bridge vlan-id-list 1-5
set interfaces lt-5/0/0 unit 1 description TO-EVPN-MPLS
set interfaces lt-5/0/0 unit 1 encapsulation ethernet-bridge
set interfaces lt-5/0/0 unit 1 peer-unit 0
set interfaces lt-5/0/0 unit 1 family bridge interface-mode trunk
set interfaces lt-5/0/0 unit 1 family bridge vlan-id-list 1-5
set interfaces irb unit 1 proxy-macip-advertisement
set interfaces irb unit 1 virtual-gateway-esi 00:22:aa:aa:aa:aa:aa:aa:aa:aa
set interfaces irb unit 1 virtual-gateway-esi all-active
set interfaces irb unit 1 family inet address 10.11.1.14/24 virtual-gateway-address 10.11.1.11
set interfaces irb unit 2 proxy-macip-advertisement
set interfaces irb unit 2 virtual-gateway-esi 00:22:bb:bb:bb:bb:bb:bb:bb:bb
set interfaces irb unit 2 virtual-gateway-esi all-active
set interfaces irb unit 2 family inet address 10.12.1.14/24 virtual-gateway-address 10.12.1.11
set interfaces irb unit 3 proxy-macip-advertisement
set interfaces irb unit 3 virtual-gateway-esi 00:22:cc:cc:cc:cc:cc:cc:cc:cc
set interfaces irb unit 3 virtual-gateway-esi all-active
set interfaces irb unit 3 family inet address 10.13.1.14/24 virtual-gateway-address 10.13.1.11

```

```

set interfaces irb unit 4 proxy-macip-advertisement
set interfaces irb unit 4 virtual-gateway-esi 00:22:dd:dd:dd:dd:dd:dd:dd
set interfaces irb unit 4 virtual-gateway-esi all-active
set interfaces irb unit 4 family inet address 10.14.1.14/24 virtual-gateway-address 10.14.1.11
set interfaces irb unit 5 proxy-macip-advertisement
set interfaces irb unit 5 virtual-gateway-esi 00:22:ee:ee:ee:ee:ee:ee:ee
set interfaces irb unit 5 virtual-gateway-esi all-active
set interfaces irb unit 5 family inet address 10.15.1.14/24 virtual-gateway-address 10.15.1.11
set interfaces lo0 unit 0 family inet address 198.51.100.21/32
set interfaces lo0 unit 0 family mpls
set policy-options policy-statement L0 from protocol direct
set policy-options policy-statement L0 from route-filter 198.51.100.21/32 exact
set policy-options policy-statement L0 then accept
set policy-options policy-statement TEST then community add NO-EXPORT
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set policy-options community NO-EXPORT members no-advertise
set policy-options community NO-EXPORT members no-export
set policy-options community NO-EXPORT members no-export-subconfed
set routing-instances EVPN-MPLS-1 instance-type virtual-switch
set routing-instances EVPN-MPLS-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-MPLS-1 protocols evpn extended-vlan-list 1-5
set routing-instances EVPN-MPLS-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-MPLS-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-MPLS-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-MPLS-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-MPLS-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-MPLS-1 interface lt-5/0/0.0
set routing-instances EVPN-MPLS-1 route-distinguisher 198.51.100.21:100
set routing-instances EVPN-MPLS-1 vrf-target target:1:2
set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-VXLAN-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5
set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 routing-interface irb.1

```

```

set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 routing-interface irb.2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 routing-interface irb.3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 routing-interface irb.4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 routing-interface irb.5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
set routing-instances EVPN-VXLAN-1 interface lt-5/0/0.1
set routing-instances EVPN-VXLAN-1 route-distinguisher 198.51.100.21:1
set routing-instances EVPN-VXLAN-1 vrf-target target:1:3
set routing-instances VRF instance-type vrf
set routing-instances VRF interface irb.1
set routing-instances VRF interface irb.2
set routing-instances VRF interface irb.3
set routing-instances VRF interface irb.4
set routing-instances VRF interface irb.5
set routing-instances VRF route-distinguisher 1:1
set routing-instances VRF vrf-target target:10:10
set routing-options router-id 198.51.100.21
set routing-options autonomous-system 65300
set routing-options forwarding-table export evpn-pplb
set protocols bgp group INT type internal
set protocols bgp group INT local-address 198.51.100.21
set protocols bgp group INT family evpn signaling
set protocols bgp group INT export TEST
set protocols bgp group INT neighbor 203.0.113.1
set protocols bgp group MX22 type external
set protocols bgp group MX22 local-address 192.168.13.1
set protocols bgp group MX22 export TEST
set protocols bgp group MX22 export L0
set protocols bgp group MX22 peer-as 64900
set protocols bgp group MX22 local-as 64800
set protocols bgp group MX22 neighbor 192.168.13.2 family inet unicast

```

```
set protocols bgp group ToR21 type external
set protocols bgp group ToR21 local-address 192.168.9.1
set protocols bgp group ToR21 export TEST
set protocols bgp group ToR21 export L0
set protocols bgp group ToR21 peer-as 64600
set protocols bgp group ToR21 local-as 64800
set protocols bgp group ToR21 neighbor 192.168.9.2 family inet unicast
set protocols bgp group ToR22 type external
set protocols bgp group ToR22 local-address 192.168.8.1
set protocols bgp group ToR22 export TEST
set protocols bgp group ToR22 export L0
set protocols bgp group ToR22 peer-as 64700
set protocols bgp group ToR22 local-as 64800
set protocols bgp group ToR22 neighbor 192.168.8.2 family inet unicast
set protocols bgp group MX22-EVPN type external
set protocols bgp group MX22-EVPN multihop ttl 2
set protocols bgp group MX22-EVPN multihop no-nexthop-change
set protocols bgp group MX22-EVPN local-address 198.51.100.21
set protocols bgp group MX22-EVPN peer-as 64900
set protocols bgp group MX22-EVPN local-as 64800
set protocols bgp group MX22-EVPN neighbor 198.51.100.22 family evpn signaling
set protocols bgp group ToR21-EVPN type external
set protocols bgp group ToR21-EVPN multihop ttl 2
set protocols bgp group ToR21-EVPN multihop no-nexthop-change
set protocols bgp group ToR21-EVPN local-address 198.51.100.21
set protocols bgp group ToR21-EVPN peer-as 64600
set protocols bgp group ToR21-EVPN local-as 64800
set protocols bgp group ToR21-EVPN neighbor 198.51.100.11 family evpn signaling
set protocols bgp group ToR22-EVPN type external
set protocols bgp group ToR22-EVPN multihop ttl 2
set protocols bgp group ToR22-EVPN multihop no-nexthop-change
set protocols bgp group ToR22-EVPN local-address 198.51.100.21
set protocols bgp group ToR22-EVPN peer-as 64700
set protocols bgp group ToR22-EVPN local-as 64800
set protocols bgp group ToR22-EVPN neighbor 198.51.100.12 family evpn signaling
set protocols bgp local-address 198.51.100.21
set protocols bgp export TEST
set protocols bgp local-as 65300
set protocols mpls label-switched-path MX21-T0-MX11 to 192.0.2.21
set protocols mpls label-switched-path MX21-T0-MX12 to 192.0.2.22
set protocols mpls label-switched-path MX21-T0-P to 203.0.113.1
set protocols mpls label-switched-path MX21-T0-MX22 to 198.51.100.22
set protocols mpls interface all
```

```

set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0

```

Data Center Gateway and WAN Edge 4 Router (MX22)

```

set system host-name MX22
set interfaces ge-0/0/0 unit 0 description "CONNECTED TO P"
set interfaces ge-0/0/0 unit 0 family inet address 203.0.113.51/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 unit 0 description "CONNECTED TO MX21"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.13.2/24
set interfaces ge-0/0/2 unit 0 description "CONNECTED TO ToR21"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.10.1/24
set interfaces ge-0/0/3 unit 0 description "CONNECTED TO ToR22"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.11.1/24
set interfaces lt-1/0/0 esi 00:33:33:33:33:33:33:33:33:33
set interfaces lt-1/0/0 esi all-active
set interfaces lt-1/0/0 unit 0 description TO-EVPN-VXLAN
set interfaces lt-1/0/0 unit 0 encapsulation ethernet-bridge
set interfaces lt-1/0/0 unit 0 peer-unit 1
set interfaces lt-1/0/0 unit 0 family bridge interface-mode trunk
set interfaces lt-1/0/0 unit 0 family bridge vlan-id-list 1-5
set interfaces lt-1/0/0 unit 1 description TO-EVPN-MPLS
set interfaces lt-1/0/0 unit 1 encapsulation ethernet-bridge
set interfaces lt-1/0/0 unit 1 peer-unit 0
set interfaces lt-1/0/0 unit 1 family bridge interface-mode trunk
set interfaces lt-1/0/0 unit 1 family bridge vlan-id-list 1-5
set interfaces irb unit 1 proxy-macip-advertisement
set interfaces irb unit 1 virtual-gateway-esi 00:22:aa:aa:aa:aa:aa:aa:aa:aa
set interfaces irb unit 1 virtual-gateway-esi all-active
set interfaces irb unit 1 family inet address 10.11.1.15/24 virtual-gateway-address 10.11.1.11
set interfaces irb unit 2 proxy-macip-advertisement
set interfaces irb unit 2 virtual-gateway-esi 00:22:bb:bb:bb:bb:bb:bb:bb:bb
set interfaces irb unit 2 virtual-gateway-esi all-active
set interfaces irb unit 2 family inet address 10.12.1.15/24 virtual-gateway-address 10.12.1.11
set interfaces irb unit 3 proxy-macip-advertisement
set interfaces irb unit 3 virtual-gateway-esi 00:22:cc:cc:cc:cc:cc:cc:cc:cc
set interfaces irb unit 3 virtual-gateway-esi all-active
set interfaces irb unit 3 family inet address 10.13.1.15/24 virtual-gateway-address 10.13.1.11

```

```

set interfaces irb unit 4 proxy-macip-advertisement
set interfaces irb unit 4 virtual-gateway-esi 00:22:dd:dd:dd:dd:dd:dd:dd
set interfaces irb unit 4 virtual-gateway-esi all-active
set interfaces irb unit 4 family inet address 10.14.1.15/24 virtual-gateway-address 10.14.1.11
set interfaces irb unit 5 proxy-macip-advertisement
set interfaces irb unit 5 virtual-gateway-esi 00:22:ee:ee:ee:ee:ee:ee:ee
set interfaces irb unit 5 virtual-gateway-esi all-active
set interfaces irb unit 5 family inet address 10.15.1.15/24 virtual-gateway-address 10.15.1.11
set interfaces lo0 unit 0 family inet address 198.51.100.22/32
set interfaces lo0 unit 0 family mpls
set policy-options policy-statement L0 from protocol direct
set policy-options policy-statement L0 from route-filter 198.51.100.22/32 exact
set policy-options policy-statement L0 then accept
set policy-options policy-statement TEST then community add NO-EXPORT
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set policy-options community NO-EXPORT members no-advertise
set policy-options community NO-EXPORT members no-export
set policy-options community NO-EXPORT members no-export-subconfed
set routing-instances EVPN-MPLS-1 instance-type virtual-switch
set routing-instances EVPN-MPLS-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-MPLS-1 protocols evpn extended-vlan-list 1-5
set routing-instances EVPN-MPLS-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-MPLS-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-MPLS-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-MPLS-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-MPLS-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-MPLS-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-MPLS-1 interface lt-1/0/0.0
set routing-instances EVPN-MPLS-1 route-distinguisher 198.51.100.22:100
set routing-instances EVPN-MPLS-1 vrf-target target:1:2
set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-VXLAN-1 protocols evpn default-gateway no-gateway-community
set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5
set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 routing-interface irb.1

```

```

set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 routing-interface irb.2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 routing-interface irb.3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 routing-interface irb.4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 routing-interface irb.5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
set routing-instances EVPN-VXLAN-1 interface lt-1/0/0.1
set routing-instances EVPN-VXLAN-1 route-distinguisher 198.51.100.22:1
set routing-instances EVPN-VXLAN-1 vrf-target target:1:3
set routing-instances VRF instance-type vrf
set routing-instances VRF interface irb.1
set routing-instances VRF interface irb.2
set routing-instances VRF interface irb.3
set routing-instances VRF interface irb.4
set routing-instances VRF interface irb.5
set routing-instances VRF route-distinguisher 1:1
set routing-instances VRF vrf-target target:10:10
set routing-options router-id 198.51.100.22
set routing-options autonomous-system 65300
set routing-options forwarding-table export evpn-pplb
set protocols bgp group INT type internal
set protocols bgp group INT family evpn signaling
set protocols bgp group INT export TEST
set protocols bgp group INT neighbor 203.0.113.1
set protocols bgp group MX21 type external
set protocols bgp group MX21 local-address 192.168.13.2
set protocols bgp group MX21 export TEST
set protocols bgp group MX21 export L0
set protocols bgp group MX21 peer-as 64800
set protocols bgp group MX21 local-as 64900
set protocols bgp group MX21 neighbor 192.168.13.1 family inet unicast
set protocols bgp group ToR21 type external

```



```
set protocols bgp group ToR21 local-address 192.168.10.1
set protocols bgp group ToR21 export TEST
set protocols bgp group ToR21 export LO
set protocols bgp group ToR21 peer-as 64600
set protocols bgp group ToR21 local-as 64900
set protocols bgp group ToR21 neighbor 192.168.10.2 family inet unicast
set protocols bgp group ToR22 type external
set protocols bgp group ToR22 local-address 192.168.11.1
set protocols bgp group ToR22 export TEST
set protocols bgp group ToR22 export LO
set protocols bgp group ToR22 peer-as 64700
set protocols bgp group ToR22 local-as 64900
set protocols bgp group ToR22 neighbor 192.168.11.2 family inet unicast
set protocols bgp group MX21-EVPN type external
set protocols bgp group MX21-EVPN multihop ttl 2
set protocols bgp group MX21-EVPN multihop no-nexthop-change
set protocols bgp group MX21-EVPN local-address 198.51.100.22
set protocols bgp group MX21-EVPN peer-as 64800
set protocols bgp group MX21-EVPN local-as 64900
set protocols bgp group MX21-EVPN neighbor 198.51.100.21 family evpn signaling
set protocols bgp group ToR21-EVPN type external
set protocols bgp group ToR21-EVPN multihop ttl 2
set protocols bgp group ToR21-EVPN multihop no-nexthop-change
set protocols bgp group ToR21-EVPN local-address 198.51.100.22
set protocols bgp group ToR21-EVPN peer-as 64600
set protocols bgp group ToR21-EVPN local-as 64900
set protocols bgp group ToR21-EVPN neighbor 198.51.100.11 family evpn signaling
set protocols bgp group ToR22-EVPN type external
set protocols bgp group ToR22-EVPN multihop ttl 2
set protocols bgp group ToR22-EVPN multihop no-nexthop-change
set protocols bgp group ToR22-EVPN local-address 198.51.100.22
set protocols bgp group ToR22-EVPN peer-as 64700
set protocols bgp group ToR22-EVPN local-as 64900
set protocols bgp group ToR22-EVPN neighbor 198.51.100.12 family evpn signaling
set protocols bgp local-address 198.51.100.22
set protocols bgp export TEST
set protocols bgp local-as 65300
set protocols mpls label-switched-path MX22-T0-MX11 to 192.0.2.21
set protocols mpls label-switched-path MX22-T0-MX12 to 192.0.2.22
set protocols mpls label-switched-path MX22-T0-P to 203.0.113.1
set protocols mpls label-switched-path MX22-T0-MX21 to 198.51.100.21
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
```

```

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-0/0/0.0
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable

```

ToR21

```

set system host-name ToR21
set chassis aggregated-devices ethernet device-count 1
set interfaces ge-0/0/0 unit 0 description "CONNECTED TO MX21"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.9.2/24
set interfaces ge-0/0/1 unit 0 description "CONNECTED TO MX22"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.10.2/24
set interfaces ge-0/0/2 unit 0 description "CONNECTED TO ToR22"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.12.1/24
set interfaces ge-0/0/3 description "CONNECTED TO CE-5"
set interfaces ge-0/0/3 gigether-options 802.3ad ae0
set interfaces ge-0/0/4 unit 0 description "CONNECTED TO CE-4"
set interfaces ge-0/0/4 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/4 unit 0 family bridge vlan-id-list 1-5
set interfaces ae0 esi 00:44:44:44:44:44:44:44:44:44
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 22:22:22:22:22:22
set interfaces ae0 unit 0 family bridge interface-mode trunk
set interfaces ae0 unit 0 family bridge vlan-id-list 1-5
set interfaces lo0 unit 0 family inet address 198.51.100.11/32
set policy-options policy-statement L0 term 1 from protocol direct
set policy-options policy-statement L0 term 1 from route-filter 198.51.100.11/32 exact
set policy-options policy-statement L0 term 1 then accept
set policy-options policy-statement TEST then community add NO-EXPORT
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set policy-options community NO-EXPORT members no-advertise
set policy-options community NO-EXPORT members no-export
set policy-options community NO-EXPORT members no-export-subconfed
set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5

```

```

set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
set routing-instances EVPN-VXLAN-1 interface ge-0/0/4.0
set routing-instances EVPN-VXLAN-1 interface ae0.0
set routing-instances EVPN-VXLAN-1 route-distinguisher 198.51.100.11:1
set routing-instances EVPN-VXLAN-1 vrf-target target:1:3
set routing-options router-id 198.51.100.11
set routing-options autonomous-system 64600
set routing-options forwarding-table export evpn-pplb
set protocols bgp group MX21 type external
set protocols bgp group MX21 local-address 192.168.9.2
set protocols bgp group MX21 export L0
set protocols bgp group MX21 export TEST
set protocols bgp group MX21 peer-as 64800
set protocols bgp group MX21 local-as 64600
set protocols bgp group MX21 neighbor 192.168.9.1 family inet unicast
set protocols bgp group MX22 type external
set protocols bgp group MX22 local-address 192.168.10.2
set protocols bgp group MX22 export L0
set protocols bgp group MX22 export TEST
set protocols bgp group MX22 peer-as 64900
set protocols bgp group MX22 local-as 64600
set protocols bgp group MX22 neighbor 192.168.10.1 family inet unicast
set protocols bgp group ToR22 type external
set protocols bgp group ToR22 local-address 192.168.12.1
set protocols bgp group ToR22 export L0
set protocols bgp group ToR22 export TEST
set protocols bgp group ToR22 peer-as 64700
set protocols bgp group ToR22 local-as 64600

```

```

set protocols bgp group ToR22 neighbor 192.168.12.2 family inet unicast
set protocols bgp group MX21-EVPN type external
set protocols bgp group MX21-EVPN multihop ttl 2
set protocols bgp group MX21-EVPN multihop no-nexthop-change
set protocols bgp group MX21-EVPN local-address 198.51.100.11
set protocols bgp group MX21-EVPN peer-as 64800
set protocols bgp group MX21-EVPN local-as 64600
set protocols bgp group MX21-EVPN neighbor 198.51.100.21 family evpn signaling
set protocols bgp group MX22-EVPN type external
set protocols bgp group MX22-EVPN multihop ttl 2
set protocols bgp group MX22-EVPN multihop no-nexthop-change
set protocols bgp group MX22-EVPN local-address 198.51.100.11
set protocols bgp group MX22-EVPN peer-as 64900
set protocols bgp group MX22-EVPN local-as 64600
set protocols bgp group MX22-EVPN neighbor 198.51.100.22 family evpn signaling
set protocols bgp group ToR22-EVPN type external
set protocols bgp group ToR22-EVPN multihop ttl 2
set protocols bgp group ToR22-EVPN multihop no-nexthop-change
set protocols bgp group ToR22-EVPN local-address 198.51.100.11
set protocols bgp group ToR22-EVPN peer-as 64700
set protocols bgp group ToR22-EVPN local-as 64600
set protocols bgp group ToR22-EVPN neighbor 198.51.100.12 family evpn signaling
set protocols bgp export TEST
set protocols bgp local-as 64600

```

ToR22

```

set system host-name ToR22
set chassis aggregated-devices ethernet device-count 2
set interfaces ge-0/0/0 unit 0 description "CONNECTED TO MX21"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.8.2/24
set interfaces ge-0/0/1 unit 0 description "CONNECTED TO MX22"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.11.2/24
set interfaces ge-0/0/2 unit 0 description "CONNECTED TO ToR21"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.12.2/24
set interfaces ge-0/0/3 description "CONNECTED TO ToR21"
set interfaces ge-0/0/3 gigether-options 802.3ad ae1
set interfaces ge-0/0/4 description "CONNECTED TO CE-5"
set interfaces ge-0/0/4 gigether-options 802.3ad ae0
set interfaces ge-0/0/5 description "CONNECTED TO ToR22"
set interfaces ge-0/0/5 gigether-options 802.3ad ae1

```

```

set interfaces ge-0/0/6 unit 0 description "CONNECTED TO CE-6"
set interfaces ge-0/0/6 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/6 unit 0 family bridge vlan-id-list 1-5
set interfaces ae0 esi 00:44:44:44:44:44:44:44:44:44
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 22:22:22:22:22:22
set interfaces ae0 unit 0 family bridge interface-mode trunk
set interfaces ae0 unit 0 family bridge vlan-id-list 1-5
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 aggregated-ether-options lacp system-id 22:22:22:22:22:22
set interfaces lo0 unit 0 family inet address 198.51.100.12/32
set policy-options policy-statement L0 term 1 from protocol direct
set policy-options policy-statement L0 term 1 from route-filter 198.51.100.12/32 exact
set policy-options policy-statement L0 term 1 then accept
set policy-options policy-statement TEST then community add NO-EXPORT
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set policy-options community NO-EXPORT members no-advertise
set policy-options community NO-EXPORT members no-export
set policy-options community NO-EXPORT members no-export-subconfed
set routing-instances EVPN-VXLAN-1 instance-type virtual-switch
set routing-instances EVPN-VXLAN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-VXLAN-1 protocols evpn extended-vni-list 1-5
set routing-instances EVPN-VXLAN-1 vtep-source-interface lo0.0
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vlan-id 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-1 vxlan vni 1
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vlan-id 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-2 vxlan vni 2
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vlan-id 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-3 vxlan vni 3
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vlan-id 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-4 vxlan vni 4
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 domain-type bridge
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vlan-id 5
set routing-instances EVPN-VXLAN-1 bridge-domains BD-5 vxlan vni 5
set routing-instances EVPN-VXLAN-1 interface ge-0/0/6.0

```

```
set routing-instances EVPN-VXLAN-1 interface ae0.0
set routing-instances EVPN-VXLAN-1 route-distinguisher 198.51.100.12:1
set routing-instances EVPN-VXLAN-1 vrf-target target:1:3
set routing-options router-id 198.51.100.12
set routing-options autonomous-system 64700
set routing-options forwarding-table export evpn-pplb
set protocols bgp group MX21 type external
set protocols bgp group MX21 local-address 192.168.8.2
set protocols bgp group MX21 export L0
set protocols bgp group MX21 export TEST
set protocols bgp group MX21 peer-as 64800
set protocols bgp group MX21 local-as 64700
set protocols bgp group MX21 neighbor 192.168.8.1 family inet unicast
set protocols bgp group MX22 type external
set protocols bgp group MX22 local-address 192.168.11.2
set protocols bgp group MX22 export L0
set protocols bgp group MX22 export TEST
set protocols bgp group MX22 peer-as 64900
set protocols bgp group MX22 local-as 64700
set protocols bgp group MX22 neighbor 192.168.11.1 family inet unicast
set protocols bgp group ToR21 type external
set protocols bgp group ToR21 local-address 192.168.12.2
set protocols bgp group ToR21 export L0
set protocols bgp group ToR21 export TEST
set protocols bgp group ToR21 peer-as 64600
set protocols bgp group ToR21 local-as 64700
set protocols bgp group ToR21 neighbor 192.168.12.1 family inet unicast
set protocols bgp group MX21-EVPN type external
set protocols bgp group MX21-EVPN multihop ttl 2
set protocols bgp group MX21-EVPN multihop no-nexthop-change
set protocols bgp group MX21-EVPN local-address 198.51.100.12
set protocols bgp group MX21-EVPN peer-as 64800
set protocols bgp group MX21-EVPN local-as 64700
set protocols bgp group MX21-EVPN neighbor 198.51.100.21 family evpn signaling
set protocols bgp group MX22-EVPN type external
set protocols bgp group MX22-EVPN multihop ttl 2
set protocols bgp group MX22-EVPN multihop no-nexthop-change
set protocols bgp group MX22-EVPN local-address 198.51.100.12
set protocols bgp group MX22-EVPN peer-as 64900
set protocols bgp group MX22-EVPN local-as 64700
set protocols bgp group MX22-EVPN neighbor 198.51.100.22 family evpn signaling
set protocols bgp group ToR21-EVPN type external
set protocols bgp group ToR21-EVPN multihop ttl 2
```

```

set protocols bgp group ToR21-EVPN multihop no-nexthop-change
set protocols bgp group ToR21-EVPN local-address 198.51.100.12
set protocols bgp group ToR21-EVPN peer-as 64600
set protocols bgp group ToR21-EVPN local-as 64700
set protocols bgp group ToR21-EVPN neighbor 198.51.100.11 family evpn signaling
set protocols bgp export TEST
set protocols bgp local-as 64700

```

P

```

set interfaces ge-0/0/3 unit 0 description "CONNECTED TO MX11"
set interfaces ge-0/0/3 unit 0 family inet address 203.0.113.42/24
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces ge-0/0/5 unit 0 description "CONNECTED TO MX12"
set interfaces ge-0/0/5 unit 0 family inet address 203.0.113.12/24
set interfaces ge-0/0/5 unit 0 family mpls
set interfaces ge-0/0/6 unit 0 description "CONNECTED TO MX21"
set interfaces ge-0/0/6 unit 0 family inet address 203.0.113.32/24
set interfaces ge-0/0/6 unit 0 family mpls
set interfaces ge-0/0/7 unit 0 description "CONNECTED TO MX22"
set interfaces ge-0/0/7 unit 0 family inet address 203.0.113.52/24
set interfaces ge-0/0/7 unit 0 family mpls
set interfaces lo0 unit 86 family inet address 203.0.113.1/32
set interfaces lo0 unit 86 family mpls
set protocols bgp group INT type internal
set protocols bgp group INT import BLOCK-VXLAN-ROUTES-FROM-CORE
set protocols bgp group INT family evpn signaling
set protocols bgp group INT cluster 203.0.113.1
set protocols bgp group INT neighbor 192.0.2.21
set protocols bgp group INT neighbor 192.0.2.22
set protocols bgp group INT neighbor 198.51.100.21
set protocols bgp group INT neighbor 198.51.100.22
set protocols bgp local-address 203.0.113.1
set protocols bgp local-as 65300
set protocols mpls label-switched-path P-T0-MX11 from 203.0.113.1
set protocols mpls label-switched-path P-T0-MX11 to 192.0.2.21
set protocols mpls label-switched-path P-T0-MX12 to 192.0.2.22
set protocols mpls label-switched-path P-T0-MX21 to 198.51.100.21
set protocols mpls label-switched-path P-T0-MX22 to 198.51.100.22
set protocols mpls interface all
set protocols ospf traffic-engineering

```

```

set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.86
set protocols rsvp interface all
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 1 from protocol bgp
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 1 from community RT-CORE
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 1 then accept
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 2 from protocol bgp
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 2 from community RT-DC1
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 2 then reject
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 3 from protocol bgp
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 3 from community RT-DC2
set policy-options policy-statement BLOCK-VXLAN-ROUTES-FROM-CORE term 3 then reject
set policy-options community RT-CORE members target:1:2
set policy-options community RT-DC1 members target:1:1
set policy-options community RT-DC2 members target:1:3
set routing-options router-id 203.0.113.1
set routing-options autonomous-system 65300

```

CE1

```

set system host-name CE1
set interfaces ge-0/0/8 unit 0 description "CONNECTED TO Host 1"
set interfaces ge-0/0/8 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/8 unit 0 family bridge vlan-id-list 1-5
set interfaces ae1 unit 0 description "CONNECTED TO ToR12"
set interfaces ae1 unit 0 family bridge interface-mode trunk
set interfaces ae1 unit 0 family bridge vlan-id-list 1-5
set bridge-domains BD-1 domain-type bridge
set bridge-domains BD-1 vlan-id 1
set bridge-domains BD-2 domain-type bridge
set bridge-domains BD-2 vlan-id 2
set bridge-domains BD-3 domain-type bridge
set bridge-domains BD-3 vlan-id 3
set bridge-domains BD-4 domain-type bridge
set bridge-domains BD-4 vlan-id 4
set bridge-domains BD-5 domain-type bridge
set bridge-domains BD-5 vlan-id 5

```


CE2

```
set system host-name CE2
set interfaces ge-0/0/1 unit 0 description "CONNECTED TO ToR11"
set interfaces ge-0/0/1 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/1 unit 0 family bridge vlan-id-list 1-5
set interfaces ge-0/0/6 unit 0 description "CONNECTED TO Host-2"
set interfaces ge-0/0/6 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/6 unit 0 family bridge vlan-id-list 1-5
set bridge-domains BD-1 domain-type bridge
set bridge-domains BD-1 vlan-id 1
set bridge-domains BD-2 domain-type bridge
set bridge-domains BD-2 vlan-id 2
set bridge-domains BD-3 domain-type bridge
set bridge-domains BD-3 vlan-id 3
set bridge-domains BD-4 domain-type bridge
set bridge-domains BD-4 vlan-id 4
set bridge-domains BD-5 domain-type bridge
set bridge-domains BD-5 vlan-id 5
```

CE3

```
set system host-name CE3
set interfaces ge-0/0/5 unit 0 description "CONNECTED TO ToR12"
set interfaces ge-0/0/5 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/5 unit 0 family bridge vlan-id-list 1-5
set interfaces ge-0/0/9 unit 0 description "CONNECTED TO Host 3"
set interfaces ge-0/0/9 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/9 unit 0 family bridge vlan-id-list 1-5
set bridge-domains BD-1 domain-type bridge
set bridge-domains BD-1 vlan-id 1
set bridge-domains BD-2 domain-type bridge
set bridge-domains BD-2 vlan-id 2
set bridge-domains BD-3 domain-type bridge
set bridge-domains BD-3 vlan-id 3
set bridge-domains BD-4 domain-type bridge
set bridge-domains BD-4 vlan-id 4
set bridge-domains BD-5 domain-type bridge
set bridge-domains BD-5 vlan-id 5
```

CE4

```
set system host-name CE4
set interfaces ge-0/0/5 unit 0 description "CONNECTED TO ToR21"
set interfaces ge-0/0/5 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/5 unit 0 family bridge vlan-id-list 1-5
set interfaces ge-0/0/6 unit 0 description "CONNECTED TO Host 4"
set interfaces ge-0/0/6 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/6 unit 0 family bridge vlan-id-list 1-5
set bridge-domains BD-1 domain-type bridge
set bridge-domains BD-1 vlan-id 1
set bridge-domains BD-2 domain-type bridge
set bridge-domains BD-2 vlan-id 2
set bridge-domains BD-3 domain-type bridge
set bridge-domains BD-3 vlan-id 3
set bridge-domains BD-4 domain-type bridge
set bridge-domains BD-4 vlan-id 4
set bridge-domains BD-5 domain-type bridge
set bridge-domains BD-5 vlan-id 5
```

CE5

```
set system host-name CE5
set interfaces ge-0/0/8 unit 0 description "CONNECTED TO Host 5"
set interfaces ge-0/0/8 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/8 unit 0 family bridge vlan-id-list 1-5
set interfaces ae1 unit 0 description "CONNECTED TO ToR21"
set interfaces ae1 unit 0 family bridge interface-mode trunk
set interfaces ae1 unit 0 family bridge vlan-id-list 1-5
set bridge-domains BD-1 domain-type bridge
set bridge-domains BD-1 vlan-id 1
set bridge-domains BD-2 domain-type bridge
set bridge-domains BD-2 vlan-id 2
set bridge-domains BD-3 domain-type bridge
set bridge-domains BD-3 vlan-id 3
set bridge-domains BD-4 domain-type bridge
set bridge-domains BD-4 vlan-id 4
set bridge-domains BD-5 domain-type bridge
set bridge-domains BD-5 vlan-id 5
```

CE6

```

set system host-name CE6
set interfaces ge-0/0/7 unit 0 description "CONNECTED TO ToR22"
set interfaces ge-0/0/7 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/7 unit 0 family bridge vlan-id-list 1-5
set interfaces ge-0/0/9 unit 0 description "CONNECTED TO Host 6"
set interfaces ge-0/0/9 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/9 unit 0 family bridge vlan-id-list 1-5
set bridge-domains BD-1 domain-type bridge
set bridge-domains BD-1 vlan-id 1
set bridge-domains BD-2 domain-type bridge
set bridge-domains BD-2 vlan-id 2
set bridge-domains BD-3 domain-type bridge
set bridge-domains BD-3 vlan-id 3
set bridge-domains BD-4 domain-type bridge
set bridge-domains BD-4 vlan-id 4
set bridge-domains BD-5 domain-type bridge
set bridge-domains BD-5 vlan-id 5

```

Appendix 2: Show Configuration Output on DUT

Show command output on the DUT (MX11 and ToR11)

From configuration mode, confirm your configuration by entering the **show configuration** command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

MX11

```

user@MX11#show configurationinterfaces {
  ge-0/0/0 {
    unit 0 {
      description "CONNECTED TO ToR11";
      family inet {
        address 192.168.3.2/24;
      }
    }
  }
  ge-0/0/1 {
    unit 0 {

```

```

        description "CONNECTED TO ToR12";
        family inet {
            address 192.168.6.2/24;
        }
    }
}
ge-0/0/2 {
    unit 0 {
        description "CONNECTED TO MX12";
        family inet {
            address 192.168.7.1/24;
        }
    }
}
ge-0/0/3 {
    unit 0 {
        description "CONNECTED TO P";
        family inet {
            address 203.0.113.41/24;
        }
        family mpls;
    }
}
lt-5/1/0 {
    esi {
        00:22:22:22:22:22:22:22:22:22;
        all-active;
    }
    unit 0 {
        description TO-EVPN-VXLAN;
        encapsulation ethernet-bridge;
        peer-unit 1;
        family bridge {
            interface-mode trunk;
            vlan-id-list 1-5;
        }
    }
    unit 1 {
        description TO-EVPN-MPLS;
        encapsulation ethernet-bridge;
        peer-unit 0;
        family bridge {
            interface-mode trunk;

```

```

        vlan-id-list 1-5;
    }
}
}
irb {
    unit 1 {
        proxy-macip-advertisement;
        virtual-gateway-esi {
            00:11:aa:aa:aa:aa:aa:aa:aa:aa;
            all-active;
        }
        family inet {
            address 10.11.1.12/24 {
                virtual-gateway-address 10.11.1.10;
            }
        }
    }
    unit 2 {
        proxy-macip-advertisement;
        virtual-gateway-esi {
            00:11:bb:bb:bb:bb:bb:bb:bb:bb;
            all-active;
        }
        family inet {
            address 10.12.1.12/24 {
                virtual-gateway-address 10.12.1.10;
            }
        }
    }
    unit 3 {
        proxy-macip-advertisement;
        virtual-gateway-esi {
            00:11:cc:cc:cc:cc:cc:cc:cc:cc;
            all-active;
        }
        family inet {
            address 10.13.1.12/24 {
                virtual-gateway-address 10.13.1.10;
            }
        }
    }
    unit 4 {
        proxy-macip-advertisement;
    }
}

```

```

        virtual-gateway-esi {
            00:11:dd:dd:dd:dd:dd:dd:dd;
            all-active;
        }
        family inet {
            address 10.14.1.12/24 {
                virtual-gateway-address 10.14.1.10;
            }
        }
    }
    unit 5 {
        proxy-macip-advertisement;
        virtual-gateway-esi {
            00:11:ee:ee:ee:ee:ee:ee:ee;
            all-active;
        }
        family inet {
            address 10.15.1.12/24 {
                virtual-gateway-address 10.15.1.10;
            }
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.21/32;
        }
        family mpls;
    }
}
policy-options {
    policy-statement LO {
        from {
            protocol direct;
            route-filter 192.0.2.21/32 exact;
        }
        then accept;
    }
    policy-statement TEST {
        then {
            community add NO-EXPORT;
        }
    }
}

```

```

    }
}
policy-statement evpn-pplb {
    from protocol evpn;
    then {
        load-balance per-packet;
    }
}
community NO-EXPORT members [ no-advertise no-export no-export-subconfed ];
}
routing-instances {
    EVPN-MPLS-1 {
        instance-type virtual-switch;
        protocols {
            evpn {
                default-gateway no-gateway-community;
                extended-vlan-list 1-5;
            }
        }
    }
    bridge-domains {
        BD-1 {
            domain-type bridge;
            vlan-id 1;
        }
        BD-2 {
            domain-type bridge;
            vlan-id 2;
        }
        BD-3 {
            domain-type bridge;
            vlan-id 3;
        }
        BD-4 {
            domain-type bridge;
            vlan-id 4;
        }
        BD-5 {
            domain-type bridge;
            vlan-id 5;
        }
    }
}
interface lt-5/1/0.0;
route-distinguisher 192.0.2.21:100;

```

```

    vrf-target target:1:2;
}
EVPN-VXLAN-1 {
    instance-type virtual-switch;
    protocols {
        evpn {
            encapsulation vxlan;
            default-gateway no-gateway-community;
            extended-vni-list 1-5;
        }
    }
    vtep-source-interface lo0.0;
    bridge-domains {
        BD-1 {
            domain-type bridge;
            vlan-id 1;
            routing-interface irb.1;
            vxlan {
                vni 1;
            }
        }
        BD-2 {
            domain-type bridge;
            vlan-id 2;
            routing-interface irb.2;
            vxlan {
                vni 2;
            }
        }
        BD-3 {
            domain-type bridge;
            vlan-id 3;
            routing-interface irb.3;
            vxlan {
                vni 3;
            }
        }
        BD-4 {
            domain-type bridge;
            vlan-id 4;
            routing-interface irb.4;
            vxlan {
                vni 4;
            }
        }
    }
}

```



```

        }
    }
    BD-5 {
        domain-type bridge;
        vlan-id 5;
        routing-interface irb.5;
        vxlan {
            vni 5;
        }
    }
}
interface lt-5/1/0.1;
route-distinguisher 192.0.2.21:1;
vrf-target target:1:1;
}
VRF {
    instance-type vrf;
    interface irb.1;
    interface irb.2;
    interface irb.3;
    interface irb.4;
    interface irb.5;
    route-distinguisher 1:1;
    vrf-target target:10:10;
}
}
routing-options {
    router-id 192.0.2.21;
    autonomous-system 65300;
    forwarding-table {
        export evpn-pplb;
    }
}
protocols {
    bgp {
        group INT {
            type internal;
            local-address 192.0.2.21;
            family evpn {
                signaling;
            }
            export TEST;
            neighbor 203.0.113.1;
        }
    }
}

```

```

}
group MX12 {
    type external;
    local-address 192.168.7.1;
    export [ TEST L0 ];
    peer-as 65500;
    local-as 65400;
    neighbor 192.168.7.2 {
        family inet {
            unicast;
        }
    }
}
group ToR11 {
    type external;
    local-address 192.168.3.2;
    import TEST;
    export [ TEST L0 ];
    peer-as 65100;
    local-as 65400;
    neighbor 192.168.3.1 {
        family inet {
            unicast;
        }
    }
}
group ToR12 {
    type external;
    local-address 192.168.6.2;
    export [ TEST L0 ];
    peer-as 65200;
    local-as 65400;
    neighbor 192.168.6.1 {
        family inet {
            unicast;
        }
    }
}
group MX12-EVPN {
    type external;
    multihop {
        ttl 2;
        no-nexthop-change;
    }
}

```

```

    }
    local-address 192.0.2.21;
    export TEST;
    peer-as 65500;
    local-as 65400;
    neighbor 192.0.2.22 {
        family evpn {
            signaling;
        }
    }
}

group ToR11-EVPN {
    type external;
    multihop {
        ttl 2;
        no-nexthop-change;
    }
    local-address 192.0.2.21;
    export TEST;
    peer-as 65100;
    local-as 65400;
    neighbor 192.0.2.11 {
        family evpn {
            signaling;
        }
    }
}

group ToR12-EVPN {
    type external;
    multihop {
        ttl 2;
        no-nexthop-change;
    }
    local-address 192.0.2.21;
    export TEST;
    peer-as 65200;
    local-as 65400;
    neighbor 192.0.2.12 {
        family evpn {
            signaling;
        }
    }
}

```

```

        local-address 192.0.2.21;
        local-as 65300;
    }
    mpls {
        label-switched-path MX11-T0-MX12 {
            to 192.0.2.22;
        }
        label-switched-path MX11-T0-P {
            to 203.0.113.1;
        }
        label-switched-path MX11-T0-MX21 {
            to 198.51.100.21;
            no-cspf;
        }
        label-switched-path MX11-T0-MX22 {
            to 198.51.100.22;
            no-cspf;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface ge-0/0/3.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}

```

ToR11

```
user@ToR11#show configuration
interfaces {
  ge-0/0/0 {
    unit 0 {
      description "CONNECTED TO CE-2";
      family bridge {
        interface-mode trunk;
        vlan-id-list 1-5;
      }
    }
  }
  ge-0/0/2 {
    description "CONNECTED TO CE-1";
    gigether-options {
      802.3ad ae0;
    }
  }
  ge-0/0/3 {
    unit 0 {
      description "CONNECTED TO ToR12";
      family inet {
        address 192.168.2.1/24;
      }
    }
  }
  ge-0/0/4 {
    unit 0 {
      description "CONNECTED TO MX-11";
      family inet {
        address 192.168.3.1/24;
      }
    }
  }
  ge-0/0/5 {
    unit 0 {
      description "CONNECTED TO MX-12";
      family inet {
        address 192.168.4.1/24;
      }
    }
  }
}
```

```

}
ae0 {
    esi {
        00:11:11:11:11:11:11:11:11:11;
        all-active;
    }
    aggregated-ether-options {
        lacp {
            active;
            periodic fast;
            system-id 11:11:11:11:11:11;
        }
    }
    unit 0 {
        family bridge {
            interface-mode trunk;
            vlan-id-list 1-5;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.11/32;
        }
    }
}
}
policy-options {
    policy-statement LO {
        term 1 {
            from {
                protocol direct;
                route-filter 192.0.2.11/32 exact;
            }
            then accept;
        }
    }
    policy-statement TEST {
        then {
            community add NO-EXPORT;
        }
    }
}

```

```

policy-statement evpn-pplb {
    from protocol evpn;
    then {
        load-balance per-packet;
    }
}
community NO-EXPORT members [ no-advertise no-export no-export-subconfed ];
}
routing-instances {
    EVPN-VXLAN-1 {
        instance-type virtual-switch;
        protocols {
            evpn {
                encapsulation vxlan;
                default-gateway no-gateway-community;
                extended-vni-list 1-5;
            }
        }
        vtep-source-interface lo0.0;
        bridge-domains {
            BD-1 {
                domain-type bridge;
                vlan-id 1;
                routing-interface irb.1;
                vxlan {
                    vni 1;
                }
            }
            BD-2 {
                domain-type bridge;
                vlan-id 2;
                routing-interface irb.2;
                vxlan {
                    vni 2;
                }
            }
            BD-3 {
                domain-type bridge;
                vlan-id 3;
                routing-interface irb.3;
                vxlan {
                    vni 3;
                }
            }
        }
    }
}

```

```

    }
    BD-4 {
        domain-type bridge;
        vlan-id 4;
        routing-interface irb.4;
        vxlan {
            vni 4;
        }
    }
    BD-5 {
        domain-type bridge;
        vlan-id 5;
        routing-interface irb.5;
        vxlan {
            vni 5;
        }
    }
}
interface ge-0/0/0.0;
interface ae0.0;
route-distinguisher 192.0.2.21:1;
vrf-target target:1:1;
}
VRF {
    instance-type vrf;
    interface irb.1; ## 'irb.1' is not defined
    interface irb.2; ## 'irb.2' is not defined
    interface irb.3; ## 'irb.3' is not defined
    interface irb.4; ## 'irb.4' is not defined
    interface irb.5; ## 'irb.5' is not defined
    route-distinguisher 1:1;
    vrf-target target:10:10;
}
}
routing-options {
    router-id 192.0.2.11;
    autonomous-system 65100;
    forwarding-table {
        export evpn-pplb;
    }
}
}
protocols {
    bgp {

```



```

group MX11 {
    type external;
    local-address 192.168.3.1;
    export [ LO TEST ];
    peer-as 65400;
    neighbor 192.168.3.2 {
        family inet {
            unicast;
        }
    }
}

group MX12 {
    type external;
    local-address 192.168.4.1;
    export [ LO TEST ];
    peer-as 65500;
    neighbor 192.168.4.2 {
        family inet {
            unicast;
        }
    }
}

group ToR12 {
    type external;
    local-address 192.168.2.1;
    export [ LO TEST ];
    peer-as 65200;
    local-as 65100;
    neighbor 192.168.2.2 {
        family inet {
            unicast;
        }
    }
}

group MX11-EVPN {
    type external;
    multihop {
        ttl 2;
        no-nexthop-change;
    }
    local-address 192.0.2.11;
    export TEST;
    peer-as 65400;
}

```

```

        local-as 65100;
        neighbor 192.0.2.21 {
            family evpn {
                signaling;
            }
        }
    }
group MX12-EVPN {
    type external;
    multihop {
        ttl 2;
        no-nexthop-change;
    }
    local-address 192.0.2.11;
    export TEST;
    peer-as 65500;
    local-as 65100;
    neighbor 192.0.2.22 {
        family evpn {
            signaling;
        }
    }
}
group ToR12-EVPN {
    type external;
    multihop {
        ttl 2;
        no-nexthop-change;
    }
    local-address 192.0.2.11;
    export TEST;
    peer-as 65200;
    local-as 65100;
    neighbor 192.0.2.12 {
        family evpn {
            signaling;
        }
    }
}
local-as 65100;
}
}

```

RELATED DOCUMENTATION

[EVPN-VXLAN Data Center Interconnect Through EVPN-MPLS WAN Overview | 1760](#)

Overview of EVPN-VXLAN Interconnect through EVPN MPLS WAN Using Gateways

IN THIS SECTION

- [Overview of EVPN-VXLAN Data Center to Data Center Stitching Through an EVPN-MPLS Fabric | 1842](#)
- [EVPN-VXLAN Gateway Interconnect Model Configuration Sample | 1843](#)

Overview of EVPN-VXLAN Data Center to Data Center Stitching Through an EVPN-MPLS Fabric

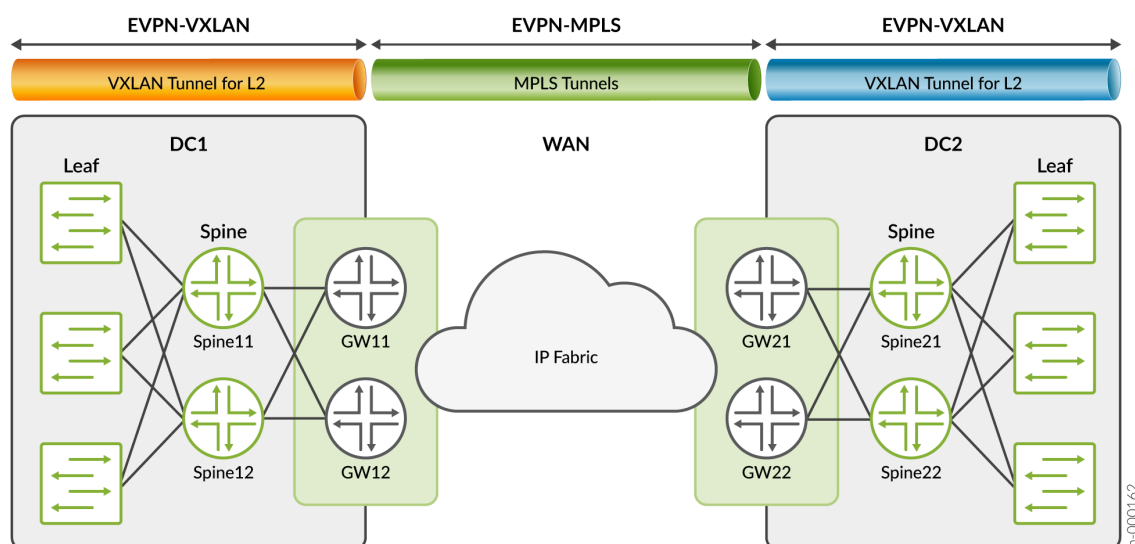
Configure seamless stitching between an EVPN-VXLAN data center, through an EVPN-MPLS fabric, to another EVPN-VXLAN data center, for interconnecting unicast and BUM traffic using WAN gateways with gateway-redundancy multihoming support. This feature is VLAN-based, and includes VLAN-aware bundle and VLAN bundle support using a VLAN list you configure. It is supported for interconnected MX platform bridge domains for both AFT-based and Ukern-based line cards.

This feature adds these parameters to the *interconnect* configuration statement:

```
interconnected-vlan-list [vlan-list];
encapsulation mpls;
```

See "[EVPN-VXLAN Gateway Interconnect Model Configuration Sample](#)" on page 1843 for a sample configuration.

Figure 188: EVPN-VXLAN Data Center to EVPN-MPLS to EVPN-VXLAN Data Center



Seamless stitching of EVPN-VXLAN to EVPN-MPLS is achieved by configuring both encapsulations, one for EVPN-VXLAN encapsulation going to the data center (DC) side, and EVPN-MPLS encapsulation on the WAN (DCI) side (under the *interconnect* stanza).

BUM traffic coming from the data center is flooded to DCI peers by pushing the IM label received from remote peers. BUM traffic coming from the WAN side is flooded to all DC VTEPs after popping the IM label.

One of the multihoming gateway nodes is elected as the designated forward (DF) based on I-ESI (interconnect and DF gateway will forward BUM traffic in either direction). To avoid the traffic duplication, the non-DF gateway drops the BUM traffic coming from the DCI side or the DC side. When there are multihoming CEs attached to gateway nodes, the current local bias filters handle split horizon functionality.

The unicast traffic from the DC side will be load balanced to remote DCI peers, and unicast traffic from the WAN is load balanced to the DC VTEPs within the data center.

EVPN-VXLAN Gateway Interconnect Model Configuration Sample

This sample configuration uses the *interconnected-vlan-list* statement and the *encapsulation (protocols evpn)* statement (with the *mpls* option) to define which VLANs to interconnect from the EVPN-VXLAN domain to the EVPN-MPLS domain.

```
interconnect {
  vrf-target target:2:2;
```

```
route-distinguisher 100:120;  
esi 00:0a:0b:0c:0d:0a:0b:0c:0d:0a; {  
    all-active;  
}  
interconnected-vlan-list [ 51 52 ];  
encapsulation mpls;  
}
```

RELATED DOCUMENTATION

| [Implementing EVPN-VXLAN for Data Centers](#)

Extending a Junos Fusion Enterprise Using EVPN-MPLS

IN THIS CHAPTER

- [Understanding EVPN-MPLS Interworking with Junos Fusion Enterprise and MC-LAG | 1845](#)
- [Example: EVPN-MPLS Interworking With Junos Fusion Enterprise | 1851](#)
- [Example: EVPN-MPLS Interworking With an MC-LAG Topology | 1871](#)

Understanding EVPN-MPLS Interworking with Junos Fusion Enterprise and MC-LAG

IN THIS SECTION

- [Benefits of Using EVPN-MPLS with Junos Fusion Enterprise and MC-LAG | 1848](#)
- [BUM Traffic Handling | 1848](#)
- [Split Horizon | 1849](#)
- [MAC Learning | 1850](#)
- [Handling Down Link Between Cascade and Uplink Ports in Junos Fusion Enterprise | 1850](#)
- [Layer 3 Gateway Support | 1851](#)

You can use Ethernet VPN (EVPN) to extend a Junos Fusion Enterprise or multichassis link aggregation group (MC-LAG) network over an MPLS network to a data center or campus network. With the introduction of this feature, you can now interconnect dispersed campus and data center sites to form a single Layer 2 virtual bridge.

[Figure 189 on page 1846](#) shows a Junos Fusion Enterprise topology with two EX9200 switches that serve as aggregation devices (PE2 and PE3) to which the satellite devices are multihomed. The two

aggregation devices use an interchassis link (ICL) and the Inter-Chassis Control Protocol (ICCP) protocol from MC-LAG to connect and maintain the Junos Fusion Enterprise topology. PE1 in the EVPN-MPLS environment interworks with PE2 and PE3 in the Junos Fusion Enterprise with MC-LAG.

Figure 189: EVPN-MPLS Interworking with Junos Fusion Enterprise

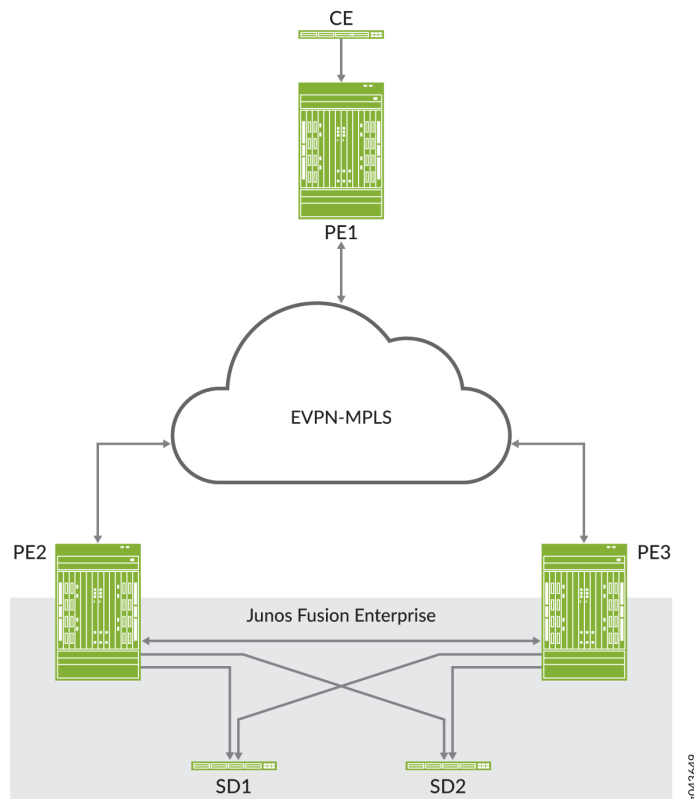
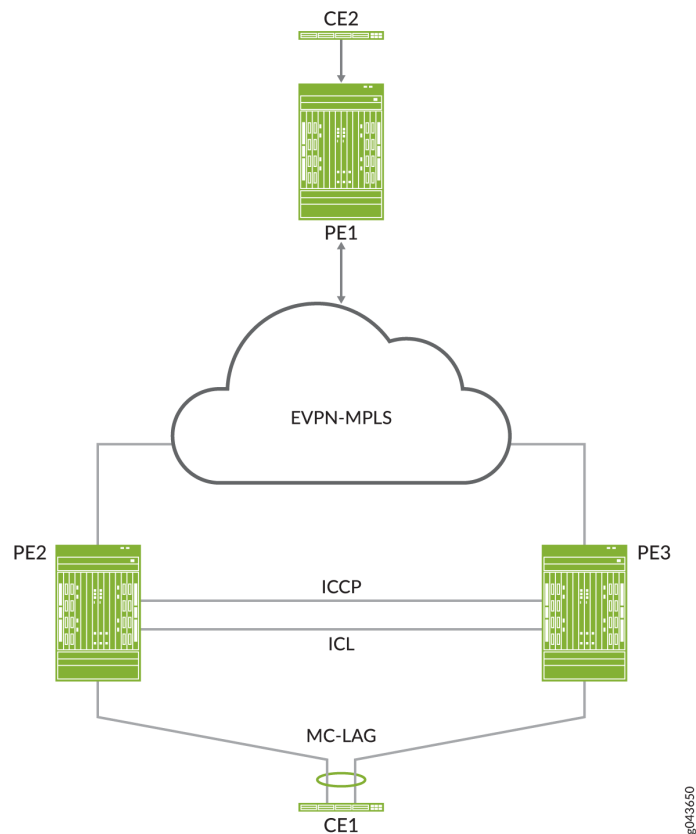


Figure 190 on page 1847 shows an MC-LAG topology in which customer edge (CE) device CE1 is multihomed to PE2 and PE3. PE2 and PE3 use an ICL and the ICCP protocol from MC-LAG to connect and maintain the topology. PE1 in the EVPN-MPLS environment interworks with PE2 and PE3 in the MC-LAG environment.

Figure 190: EVPN-MPLS Interworking with MC-LAG



Throughout this topic, [Figure 189 on page 1846](#) and [Figure 190 on page 1847](#) serve as references to illustrate various scenarios and points.

The use cases depicted in [Figure 189 on page 1846](#) and [Figure 190 on page 1847](#) require the configuration of both EVPN multihoming in active-active mode and MC-LAG on PE2 and PE3. EVPN with multihoming active-active and MC-LAG have their own forwarding logic for handling traffic, in particular, broadcast, unknown unicast, and multicast (BUM) traffic. At times, the forwarding logic for EVPN with multihoming active-active and MC-LAG contradict each other and causes issues. This topic describes the issues and how the EVPN-MPLS interworking feature resolves these issues.



NOTE: Other than the EVPN-MPLS interworking-specific implementations described in this topic, EVPN-MPLS, Junos Fusion Enterprise, and MC-LAG offer the same functionality and function the same as the standalone features.

Benefits of Using EVPN-MPLS with Junos Fusion Enterprise and MC-LAG

Use EVPN-MPLS with Junos Fusion Enterprise and MC-LAG to interconnect dispersed campus and data center sites to form a single Layer 2 virtual bridge.

BUM Traffic Handling

In the use cases shown in [Figure 189 on page 1846](#) and [Figure 190 on page 1847](#), PE1, PE2, and PE3 are EVPN peers, and PE2 and PE3 are MC-LAG peers. Both sets of peers exchange control information and forward traffic to each other, which causes issues. [Table 91 on page 1848](#) outlines the issues that arise, and how Juniper Networks resolves the issues in its implementation of the EVPN-MPLS interworking feature.

Table 91: BUM Traffic: Issues and Resolutions

BUM Traffic Direction	EVPN Interworking with Junos Fusion Enterprise and MC-LAG Logic	Issue	Juniper Networks Implementation Approach
North bound (PE2 receives BUM packet from a locally attached single- or dual-homed interfaces).	PE2 floods BUM packet to the following: <ul style="list-style-type: none"> • All locally attached interfaces, including the ICL, for a particular broadcast domain. • All remote EVPN peers for which PE2 has received inclusive multicast routes. 	Between PE2 and PE3, there are two BUM forwarding paths—the MC-LAG ICL and an EVPN-MPLS path. The multiple forwarding paths result in packet duplication and loops.	<ul style="list-style-type: none"> • BUM traffic is forwarded on the ICL only. • Incoming traffic from the EVPN core is not forwarded on the ICL. • Incoming traffic from the ICL is not forwarded to the EVPN core.
South bound (PE1 forwards BUM packet to PE2 and PE3).	PE2 and PE3 both receive a copy of the BUM packet and flood the packet out of all of their local interfaces, including the ICL.	PE2 and PE3 both forward the BUM packet out of the ICL, which results in packet duplication and loops.	

Split Horizon

In the use cases shown in [Figure 189 on page 1846](#) and [Figure 190 on page 1847](#), split horizon prevents multiple copies of a BUM packet from being forwarded to a CE device (satellite device). However, the EVPN-MPLS and MC-LAG split horizon implementations contradict each other, which causes an issue. [Table 92 on page 1849](#) explains the issue and how Juniper Networks resolves it in its implementation of the EVPN-MPLS interworking feature.

Table 92: BUM Traffic: Split Horizon-Related Issue and Resolution

BUM Traffic Direction	EVPN Interworking with Junos Fusion Enterprise and MC-LAG Logic	Issue	Juniper Networks Implementation Approach
North bound (PE2 receives BUM packet from a locally attached dual-homed interface).	<ul style="list-style-type: none"> Per EVPN-MPLS forwarding logic: <ul style="list-style-type: none"> Only the designated forwarder (DF) for the Ethernet segment (ES) can forward BUM traffic. The local bias rule, in which the local peer forwards the BUM packet and the remote peer drops it, is not supported. Per MC-LAG forwarding logic, local bias is supported. 	The EVPN-MPLS and MC-LAG forwarding logic contradicts each other and can prevent BUM traffic from being forwarded to the ES.	Support local bias, thereby ignoring the DF and non-DF status of the port for locally switched traffic.
South bound (PE1 forwards BUM packet to PE2 and PE3).	Traffic received from PE1 follows the EVPN DF and non-DF forwarding rules for a multihomed ES.	None.	Not applicable.

MAC Learning


EVPN and MC-LAG use the same method for learning MAC addresses—namely, a PE device learns MAC addresses from its local interfaces and synchronizes the addresses to its peers. However, given that both EVPN and MC-LAG are synchronizing the addresses, an issue arises.

[Table 93 on page 1850](#) describes the issue and how the EVPN-MPLS interworking implementation prevents the issue. The use cases shown in [Figure 189 on page 1846](#) and [Figure 190 on page 1847](#) illustrate the issue. In both use cases, PE1, PE2, and PE3 are EVPN peers, and PE2 and PE3 are MC-LAG peers.

Table 93: MAC Learning: EVPN and MC-LAG Synchronization Issue and Implementation Details

MAC Synchronization Use Case	EVPN Interworking with Junos Fusion Enterprise and MC-LAG Logic	Issue	Juniper Networks Implementation Approach
MAC addresses learned locally on single- or dual-homed interfaces on PE2 and PE3.	<ul style="list-style-type: none"> Between the EVPN peers, MAC addresses are synchronized using the EVPN BGP control plane. Between the MC-LAG peers, MAC addresses are synchronized using the MC-LAG ICCP control plane. 	PE2 and PE3 function as both EVPN peers and MC-LAG peers, which result in these devices having multiple MAC synchronization paths.	<ul style="list-style-type: none"> For PE1: use MAC addresses synchronized by EVPN BGP control plane. For PE2 and PE3: use MAC addresses synchronized by MC-LAG ICCP control plane.
MAC addresses learned locally on single- or dual-homed interfaces on PE1.	Between the EVPN peers, MAC addresses are synchronized using the EVPN BGP control plane.	None.	Not applicable.

Handling Down Link Between Cascade and Uplink Ports in Junos Fusion Enterprise



NOTE: This section applies only to EVPN-MPLS interworking with a Junos Fusion Enterprise.

In the Junos Fusion Enterprise shown in [Figure 189 on page 1846](#), assume that aggregation device PE2 receives a BUM packet from PE1 and that the link between the cascade port on PE2 and the

corresponding uplink port on satellite device SD1 is down. Regardless of whether the BUM packet is handled by MC-LAG or EVPN multihoming active-active, the result is the same—the packet is forwarded via the ICL interface to PE3, which forwards it to dual-homed SD1.

To further illustrate how EVPN with multihoming active-active handles this situation with dual-homed SD1, assume that the DF interface resides on PE2 and is associated with the down link and that the non-DF interface resides on PE3. Typically, per EVPN with multihoming active-active forwarding logic, the non-DF interface drops the packet. However, because of the down link associated with the DF interface, PE2 forwards the BUM packet via the ICL to PE3, and the non-DF interface on PE3 forwards the packet to SD1.

Layer 3 Gateway Support

The EVPN-MPLS interworking feature supports the following Layer 3 gateway functionality for extended bridge domains and VLANs:

- Integrated routing and bridging (IRB) interfaces to forward traffic between the extended bridge domains or VLANs.
- Default Layer 3 gateways to forward traffic from a physical (bare-metal) server in an extended bridge domain or VLAN to a physical server or virtual machine in another extended bridge domain or VLAN.

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
17.4R1	Starting with Junos OS Release 17.4R1, you can use Ethernet VPN (EVPN) to extend a Junos Fusion Enterprise or multichassis link aggregation group (MC-LAG) network over an MPLS network to a data center or campus network.

Example: EVPN-MPLS Interworking With Junos Fusion Enterprise

IN THIS SECTION

- [Requirements | 1852](#)
- [Overview and Topology | 1852](#)
- [Aggregation Device \(PE1 and PE2\) Configuration | 1855](#)

This example shows how to use Ethernet VPN (EVPN) to extend a Junos Fusion Enterprise over an MPLS network to a geographically distributed campus or enterprise network.

EVPN-MPLS interworking is supported with a Junos Fusion Enterprise, which is based on a multichassis link aggregation group (MC-LAG) infrastructure to provide redundancy for the EX9200 switches that function as aggregation devices.

The aggregation devices in the Junos Fusion Enterprise are connected to a provider edge (PE) device in an MPLS network. The PE device can be either an MX Series router or an EX9200 switch.

This example shows how to configure the aggregation devices in the Junos Fusion Enterprise and the PE device in the MPLS network to interwork with each other.

Requirements

This example uses the following hardware and software components:

- Three EX9200 switches:
 - PE1 and PE2, which both function as aggregation devices in the Junos Fusion Enterprise and EVPN BGP peers in the EVPN-MPLS overlay network.
 - PE3, which functions as an EVPN BGP peer in the EVPN-MPLS overlay network.
- The EX9200 switches are running Junos OS Release 17.4R1 or later software.

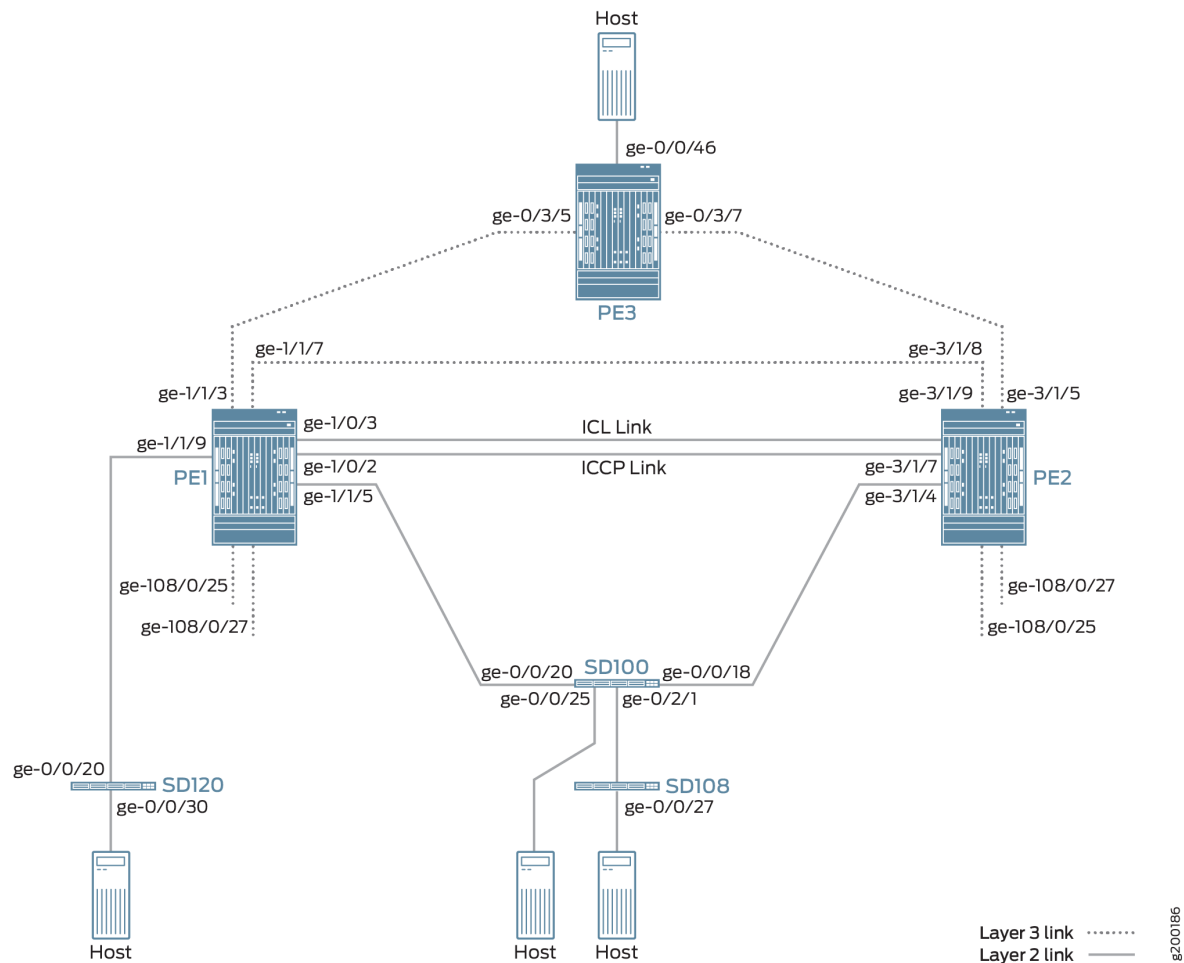


NOTE: Although the Junos Fusion Enterprise includes three satellite devices, this example focuses on the configuration of the PE1, PE2, and PE3. For more information about configuring satellite devices, see [Configuring or Expanding a Junos Fusion Enterprise](#).

Overview and Topology

[Figure 191 on page 1853](#) shows a Junos Fusion Enterprise with dual aggregation devices PE1 and PE2. The aggregation devices are connected using an interchassis link (ICL) and communicate with each other using the Inter-Chassis Control Protocol (ICCP).

Figure 191: EVPN-MPLS Interworking with Junos Fusion Enterprise



The Junos Fusion Enterprise also includes three satellite devices. Satellite device SD120 is a standalone satellite device that has a single-homed connection to PE1. Satellite devices SD100 and SD108 are included in a cluster named Cluster_100_108. SD100 is the only cluster member with a connection to an aggregation device, in this case, multihomed connections to PE1 and PE2.

The topology in [Figure 191 on page 1853](#) also includes PE3, which is positioned at the edge of an MPLS network. PE3 functions as the gateway between the Junos Fusion Enterprise network and a geographically distributed campus or enterprise network. PE1, PE2, and PE3 run EVPN, which enables hosts in the Junos Fusion Enterprise network to communicate with hosts in the campus or enterprise network by way of the intervening MPLS network.

From the perspective of the EVPN-MPLS interworking feature, PE3 functions solely as an EVPN BGP peer, and PE1 and PE2 in the Junos Fusion Enterprise have dual roles:

- Aggregation devices in the Junos Fusion Enterprise.
- EVPN BGP peers in the EVPN-MPLS network.

Because of the dual roles, PE1 and PE2 are configured with Junos Fusion Enterprise, EVPN, BGP, and MPLS attributes.

Table 94 on page 1854 outlines key Junos Fusion Enterprise and EVPN (BGP and MPLS) attributes configured on PE1, PE2, and PE3.

Table 94: Key Junos Fusion Enterprise and EVPN (BGP and MPLS) Attributes Configured on PE1, PE2, and PE3

Key Attributes	PE1	PE2	PE3
Junos Fusion Enterprise Attributes			
Interfaces	ICL: ge-1/0/3 ICCP: ge-1/0/2	ICL: ge-3/1/9 ICCP: ge-3/1/7	Not applicable
EVPN-MPLS			
Interfaces	Connection to PE3: ge-1/1/3 Connection to PE2: ge-1/1/7	Connection to PE3: ge-3/1/5 Connection to PE1: ge-3/1/8	Connection to PE1: ge-0/3/5 Connection to PE2: ge-0/3/7
IP addresses	BGP peer address: 10.25.0.1	BGP peer address: 10.25.0.2	BGP peer address: 10.25.0.3
Autonomous system	100	100	100
Virtual switch routing instances	evpn1	evpn1	evpn1

Note the following about the EVPN-MPLS interworking feature and its configuration:

- You must configure Ethernet segment identifiers (ESIs) on the dual-homed extended ports in the Junos Fusion Enterprise. The ESIs enable EVPN to identify the dual-homed extended ports.
- The only type of routing instance that is supported is the virtual switch instance (set routing-instances *name* instance-type virtual-switch).
- Only one virtual switch instance is supported with Junos Fusion Enterprise.

- On the aggregation devices in the Junos Fusion Enterprise, you must include the `bgp-peer` configuration statement in the `[edit routing-instances name protocols evpn mclag]` hierarchy level. This configuration statement enables the interworking of EVPN-MPLS with Junos Fusion Enterprise on the aggregation devices.
- Address Resolution Protocol (ARP) suppression is not supported.

Aggregation Device (PE1 and PE2) Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1855](#)
- [PE1: Configuring Junos Fusion Enterprise | 1859](#)
- [PE1: Configuring EVPN-MPLS | 1861](#)
- [PE2: Configuring Junos Fusion Enterprise | 1863](#)
- [PE2: Configuring EVPN-MPLS | 1865](#)

To configure aggregation devices PE1 and PE2, perform these tasks.



NOTE: This section focuses on enabling EVPN-MPLS on PE1 and PE2. As a result, the Junos Fusion Enterprise configuration on PE1 and PE2 is performed without the use of the configuration synchronization feature. For information about configuration synchronization, see [Understanding Configuration Synchronization](#).

CLI Quick Configuration

PE1: Junos Fusion Enterprise Configuration

```
set interfaces ge-1/1/9 cascade-port
set interfaces ge-1/1/5 cascade-port
set chassis satellite-management fpc 120 cascade-ports ge-1/1/9
set chassis satellite-management cluster Cluster_100_108 cluster-id 2
set chassis satellite-management cluster Cluster_100_108 cascade-ports ge-1/1/5
set chassis satellite-management cluster Cluster_100_108 fpc 100 alias SD100
set chassis satellite-management cluster Cluster_100_108 fpc 100 system-id 88:e0:f3:1f:3d:50
set chassis satellite-management cluster Cluster_100_108 fpc 108 alias SD108
set chassis satellite-management cluster Cluster_100_108 fpc 108 system-id 88:e0:f3:1f:c8:d1
```



```

set chassis satellite-management cluster Cluster_100_108 fpc 100 member-id 1
set chassis satellite-management cluster Cluster_100_108 fpc 108 member-id 8
set chassis satellite-management upgrade-groups upgrade_120 satellite 120
set chassis satellite-management upgrade-groups upgrade_100 satellite 100
set chassis satellite-management redundancy-groups rg1 redundancy-group-id 2
set chassis satellite-management redundancy-groups chassis-id 1
set chassis satellite-management redundancy-groups rg1 peer-chassis-id 2 inter-chassis-link
ge-1/0/3
set chassis satellite-management redundancy-groups rg1 cluster Cluster_100_108
set interfaces ge-1/0/2 description iccp-link
set interfaces ge-1/0/2 unit 0 family inet address 10.20.20.1/24
set interfaces ge-1/0/3 description icl-link
set interfaces ge-1/0/3 unit 0 family ethernet-switching interface-mode trunk
set interfaces ge-1/0/3 unit 0 family ethernet-switching vlan members 100
set switch-options service-id 1

```

PE1: EVPN-MPLS Configuration

```

set interfaces lo0 unit 0 family inet address 10.25.0.1/32
set interfaces ge-1/1/3 unit 0 family inet address 10.0.1.1/30
set interfaces ge-1/1/3 unit 0 family mpls
set interfaces ge-1/1/7 unit 0 family inet address 10.0.3.1/30
set interfaces ge-1/1/7 unit 0 family mpls
set interfaces ge-108/0/25 unit 0 esi 00:01:02:03:04:00:01:02:04:26
set interfaces ge-108/0/25 unit 0 esi all-active
set interfaces ge-108/0/25 unit 0 family ethernet-switching vlan members v100
set interfaces ge-108/0/27 unit 0 esi 00:01:02:03:04:00:01:02:04:28
set interfaces ge-108/0/27 unit 0 esi all-active
set interfaces ge-108/0/27 unit 0 family ethernet-switching vlan members v100
set routing-options router-id 10.25.0.1
set routing-options autonomous-system 100
set protocols mpls interface lo0.0
set protocols mpls interface ge-1/1/3.0
set protocols mpls interface ge-1/1/7.0
set protocols bgp local-address 10.25.0.1
set protocols bgp peer-as 100
set protocols bgp local-as 100
set protocols bgp group evpn-mes type internal
set protocols bgp group evpn-mes family evpn signaling
set protocols bgp group evpn-mes peer-as 100
set protocols bgp group evpn-mes neighbor 10.25.0.2
set protocols bgp group evpn-mes neighbor 10.25.0.3

```

```

set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-1/1/3.0
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 ospf area 0.0.0.0 interface ge-1/1/7.0
set protocols ldp interface lo0.0
set protocols ldp interface ge-1/1/3.0
set protocols ldp interface ge-1/1/7.0
set routing-instances evpn1 instance-type virtual-switch
set routing-instances evpn1 interface ge-108/0/25.0
set routing-instances evpn1 interface ge-108/0/27.0
set routing-instances evpn1 interface ge-1/0/3.0
set routing-instances evpn1 route-distinguisher 10.25.0.1:1
set routing-instances evpn1 vrf-target target:100:1
set routing-instances evpn1 protocols evpn label-allocation per-instance
set routing-instances evpn1 protocols evpn extended-vlan-list 100
set routing-instances evpn1 protocols evpn mlag bgp-peer 10.25.0.2
set routing-instances evpn1 switch-options service-id 2
set routing-instances evpn1 vlans v100 vlan-id 100

```

PE2: Junos Fusion Enterprise Configuration

```

set interfaces ge-3/1/4 cascade-port
set chassis satellite-management cluster Cluster_100_108 cluster-id 2
set chassis satellite-management cluster Cluster_100_108 cascade-ports ge-3/1/4
set chassis satellite-management cluster Cluster_100_108 fpc 100 alias SD100
set chassis satellite-management cluster Cluster_100_108 fpc 100 system-id 88:e0:f3:1f:3d:50
set chassis satellite-management cluster Cluster_100_108 fpc 108 alias SD108
set chassis satellite-management cluster Cluster_100_108 fpc 108 system-id 88:e0:f3:1f:c8:d1
set chassis satellite-management cluster Cluster_100_108 fpc 100 member-id 1
set chassis satellite-management cluster Cluster_100_108 fpc 108 member-id 8
set chassis satellite-management upgrade-groups upgrade_100 satellite 100
set chassis satellite-management redundancy-groups rg1 redundancy-group-id 2
set chassis satellite-management redundancy-groups chassis-id 2
set chassis satellite-management redundancy-groups rg1 peer-chassis-id 1 inter-chassis-link
ge-3/1/9
set chassis satellite-management redundancy-groups rg1 cluster Cluster_100_108
set interfaces ge-3/1/7 description iccp-link
set interfaces ge-3/1/7 unit 0 family inet address 10.20.20.2/24
set interfaces ge-3/1/9 description icl-link
set interfaces ge-3/1/9 unit 0 family ethernet-switching interface-mode trunk

```

```
set interfaces ge-3/1/9 unit 0 family ethernet-switching vlan members 100
set switch-options service-id 1
```

PE2: EVPN-MPLS Configuration

```
set interfaces lo0 unit 0 family inet address 10.25.0.2/32
set interfaces ge-3/1/5 unit 0 family inet address 10.0.4.2/30
set interfaces ge-3/1/5 unit 0 family mpls
set interfaces ge-3/1/8 unit 0 family inet address 10.0.3.2/30
set interfaces ge-3/1/8 unit 0 family mpls
set interfaces irb unit 0 family inet address 10.5.5.1/24 virtual-gateway-address 10.5.5.5
set interfaces ge-108/0/25 unit 0 esi 00:01:02:03:04:00:01:02:04:26
set interfaces ge-108/0/25 unit 0 esi all-active
set interfaces ge-108/0/25 unit 0 family ethernet-switching vlan members v100
set interfaces ge-108/0/27 unit 0 esi 00:01:02:03:04:00:01:02:04:28
set interfaces ge-108/0/27 unit 0 esi all-active
set interfaces ge-108/0/27 unit 0 family ethernet-switching vlan members v100
set routing-options router-id 10.25.0.2
set routing-options autonomous-system 100
set protocols mpls interface lo0.0
set protocols mpls interface ge-3/1/5.0
set protocols mpls interface ge-3/1/8.0
set protocols bgp local-address 10.25.0.2
set protocols bgp peer-as 100
set protocols bgp local-as 100
set protocols bgp group evpn-mes type internal
set protocols bgp group evpn-mes family evpn signaling
set protocols bgp group evpn-mes peer-as 100
set protocols bgp group evpn-mes neighbor 10.25.0.1
set protocols bgp group evpn-mes neighbor 10.25.0.3
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-3/1/5.0
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 ospf area 0.0.0.0 interface ge-3/1/8.0
set protocols ldp interface lo0.0
set protocols ldp interface ge-3/1/5.0
set protocols ldp interface ge-3/1/8.0
set routing-instances evpn1 instance-type virtual-switch
set routing-instances evpn1 interface ge-108/0/25.0
set routing-instances evpn1 interface ge-108/0/27.0
set routing-instances evpn1 interface ge-3/1/9.0
```

```

set routing-instances evpn1 route-distinguisher 10.25.0.2:1
set routing-instances evpn1 vrf-target target:100:1
set routing-instances evpn1 protocols evpn label-allocation per-instance
set routing-instances evpn1 protocols evpn extended-vlan-list 100
set routing-instances evpn1 protocols evpn mlag bgp-peer 10.25.0.1
set routing-instances evpn1 switch-options service-id 2
set routing-instances evpn1 vlans v100 vlan-id 100
set routing-instances evpn1 vlans v100 l3-interface irb.0

```

PE1: Configuring Junos Fusion Enterprise

Step-by-Step Procedure

1. Configure the cascade ports.

```

[edit]
user@switch# set interfaces ge-1/1/9 cascade-port
user@switch# set interfaces ge-1/1/5 cascade-port

```

2. Configure the FPC slot ID for standalone satellite device SD120 and map it to a cascade port.

```

[edit]
user@switch# set chassis satellite-management fpc 120 cascade-ports ge-1/1/9

```

3. Create a satellite device cluster, and assign a name and a cluster ID to it.

```

[edit]
user@switch# set chassis satellite-management cluster Cluster_100_108 cluster-id 2

```

4. Define the cascade ports associated with the satellite device cluster.

```

[edit]
user@switch# set chassis satellite-management cluster Cluster_100_108 cascade-ports ge-1/1/5
user@switch# set chassis satellite-management cluster Cluster_100_108 cascade-ports ge-1/1/9

```

5. Configure the FPC slot ID number, and map it to the MAC address of satellite devices SD100 and SD108, respectively.

```
[edit]
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 100 alias SD100
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 100 system-id
88:e0:f3:1f:3d:50
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 108 alias SD108
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 108 system-id
88:e0:f3:1f:c8:d1
```

6. Assign a member ID to each satellite device in the satellite device cluster.

```
[edit]
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 100 member-id 1
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 108 member-id 8
```

7. Create two satellite software upgrade groups—one that includes satellite device SD120 and another that includes satellite device SD100.

```
[edit]
user@switch# set chassis satellite-management upgrade-groups upgrade_120 satellite 120
user@switch# set chassis satellite-management upgrade-groups upgrade_100 satellite 100
```

8. Create and configure a redundancy group, which includes the aggregation devices and satellite devices in Cluster_100_108.

```
[edit]
user@switch# set chassis satellite-management redundancy-groups rg1 redundancy-group-id 2
user@switch# set chassis satellite-management redundancy-groups chassis-id 1
user@switch# set chassis satellite-management redundancy-groups rg1 peer-chassis-id 2 inter-
chassis-link ge-1/0/3
user@switch# set chassis satellite-management redundancy-groups rg1 cluster Cluster_100_108
```

9. Configure the ICL and ICCP links.

```
[edit]
user@switch# set interfaces ge-1/0/2 description iccp-link
```

```

user@switch# set interfaces ge-1/0/2 unit 0 family inet address 10.20.20.1/24
user@switch# set interfaces ge-1/0/3 description icl-link
user@switch# set interfaces ge-1/0/3 unit 0 family ethernet-switching interface-mode trunk
user@switch# set interfaces ge-1/0/3 unit 0 family ethernet-switching vlan members 100
user@switch# set switch-options service-id 1

```



NOTE: While this step shows the configuration of interface ge-1/0/2, which is designated as the ICCP interface, it does not show how to configure the ICCP attributes on interface ge-1/0/2. By default, ICCP is automatically provisioned in a Junos Fusion Enterprise using dual aggregation devices. For more information about the automatic provisioning of ICCP, see [Configuring or Expanding a Junos Fusion Enterprise](#).

PE1: Configuring EVPN-MPLS

Step-by-Step Procedure

1. Configure the loopback interface and the interfaces connected to the other PE devices.

```

[edit]
user@switch# set interfaces lo0 unit 0 family inet address 10.25.0.1/32
user@switch# set interfaces ge-1/1/3 unit 0 family inet address 10.0.1.1/30
user@switch# set interfaces ge-1/1/3 unit 0 family mpls
user@switch# set interfaces ge-1/1/7 unit 0 family inet address 10.0.3.1/30
user@switch# set interfaces ge-1/1/7 unit 0 family mpls

```

2. Configure the extended ports with EVPN multihoming in active-active mode, an ESI, and map the ports to VLAN v100..

```

[edit]
user@switch# set interfaces ge-108/0/25 unit 0 esi 00:01:02:03:04:00:01:02:04:26
user@switch# set interfaces ge-108/0/25 unit 0 esi all-active
user@switch# set interfaces ge-108/0/25 unit 0 family ethernet-switching vlan members v100
user@switch# set interfaces ge-108/0/27 unit 0 esi 00:01:02:03:04:00:01:02:04:28
user@switch# set interfaces ge-108/0/27 unit 0 esi all-active
user@switch# set interfaces ge-108/0/27 unit 0 family ethernet-switching vlan members v100

```

3. Assign a router ID and the autonomous system in which PE1, PE2, and PE3 reside.

```
[edit]
user@switch# set routing-options router-id 10.25.0.1
user@switch# set routing-options autonomous-system 100
```

4. Enable MPLS on the loopback interface and interfaces ge-1/1/3.0 and ge-1/1/7.0.

```
[edit]
user@switch# set protocols mpls interface lo0.0
user@switch# set protocols mpls interface ge-1/1/3.0
user@switch# set protocols mpls interface ge-1/1/7.0
```

5. Configure an IBGP overlay that includes PE1, PE2, and PE3.

```
[edit]
user@switch# set protocols bgp local-address 10.25.0.1
user@switch# set protocols bgp peer-as 100
user@switch# set protocols bgp local-as 100
user@switch# set protocols bgp group evpn-mes type internal
user@switch# set protocols bgp group evpn-mes family evpn signaling
user@switch# set protocols bgp group evpn-mes peer-as 100
user@switch# set protocols bgp group evpn-mes neighbor 10.25.0.2
user@switch# set protocols bgp group evpn-mes neighbor 10.25.0.3
```

6. Configure OSPF as the internal routing protocol for EVPN by specifying an area ID and interfaces on which EVPN-MPLS is enabled.

```
[edit]
user@switch# set protocols ospf traffic-engineering
user@switch# set protocols ospf area 0.0.0.0 interface ge-1/1/3.0
user@switch# set protocols ospf area 0.0.0.0 interface lo0.0
user@switch# set protocols ospf area 0.0.0.0 interface fxp0.0 disable
user@switch# set protocols ospf area 0.0.0.0 interface ge-1/1/7.0
```

7. Configure the Label Distribution Protocol (LDP) on the loopback interface and the interfaces on which EVPN-MPLS is enabled.

```
[edit]
user@switch# set protocols ldp interface lo0.0
user@switch# set protocols ldp interface ge-1/1/3.0
user@switch# set protocols ldp interface ge-1/1/7.0
```

8. Configure a virtual switch routing instance for VLAN v100, and include the interfaces and other entities associated with the VLAN.

```
[edit]
user@switch# set routing-instances evpn1 instance-type virtual-switch
user@switch# set routing-instances evpn1 interface ge-108/0/25.0
user@switch# set routing-instances evpn1 interface ge-108/0/27.0
user@switch# set routing-instances evpn1 interface ge-1/0/3.0
user@switch# set routing-instances evpn1 route-distinguisher 10.25.0.1:1
user@switch# set routing-instances evpn1 vrf-target target:100:1
user@switch# set routing-instances evpn1 protocols evpn label-allocation per-instance
user@switch# set routing-instances evpn1 protocols evpn extended-vlan-list 100
user@switch# set routing-instances evpn1 protocols evpn mclag bgp-peer 10.25.0.2
user@switch# set routing-instances evpn1 switch-options service-id 2
user@switch# set routing-instances evpn1 vlans v100 vlan-id 100
```

PE2: Configuring Junos Fusion Enterprise

Step-by-Step Procedure

1. Configure the cascade port.

```
[edit]
user@switch# set interfaces ge-3/1/4 cascade-port
```

2. Create a satellite device cluster, and assign a name and a cluster ID to it.

```
[edit]
user@switch# set chassis satellite-management cluster Cluster_100_108 cluster-id 2
```


3. Define the cascade port associated with the satellite device cluster.

```
[edit]
user@switch# set chassis satellite-management cluster Cluster_100_108 cascade-ports ge-3/1/4
```

4. Configure the FPC slot ID number, and map it to the MAC address of satellite devices SD100 and SD108, respectively.

```
[edit]
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 100 alias SD100
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 100 system-id
88:e0:f3:1f:3d:50
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 108 alias SD108
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 108 system-id
88:e0:f3:1f:c8:d1
```

5. Assign a member ID to each satellite device in the satellite device cluster.

```
[edit]
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 100 member-id 1
user@switch# set chassis satellite-management cluster Cluster_100_108 fpc 108 member-id 8
```

6. Create a satellite software upgrade group that includes satellite device SD100.

```
[edit]
user@switch# set chassis satellite-management upgrade-groups upgrade_100 satellite 100
```

7. Create and configure a redundancy group, which includes the aggregation devices and satellite devices in Cluster_100_108.

```
[edit]
user@switch# set chassis satellite-management redundancy-groups rg1 redundancy-group-id 2
user@switch# set chassis satellite-management redundancy-groups chassis-id 2
user@switch# set chassis satellite-management redundancy-groups rg1 peer-chassis-id 1 inter-
chassis-link ge-3/1/9
user@switch# set chassis satellite-management redundancy-groups rg1 cluster Cluster_100_108
```

8. Configure the ICL and ICCP links.

```
[edit]
user@switch# set interfaces ge-3/1/7 description iccp-link
user@switch# set interfaces ge-3/1/7 unit 0 family inet address 10.20.20.2/24
user@switch# set interfaces ge-3/1/9 description icl-link
user@switch# set interfaces ge-3/1/9 unit 0 family ethernet-switching interface-mode trunk
user@switch# set interfaces ge-3/1/9 unit 0 family ethernet-switching vlan members 100
user@switch# set switch-options service-id 1
```



NOTE: While this step shows the configuration of interface ge-3/1/7, which is designated as the ICCP interface, it does not show how to configure the ICCP attributes on interface ge-3/1/7. By default, ICCP is automatically provisioned in a Junos Fusion Enterprise using dual aggregation devices. For more information about the automatic provisioning of ICCP, see [Configuring or Expanding a Junos Fusion Enterprise](#).

PE2: Configuring EVPN-MPLS

Step-by-Step Procedure

1. Configure the loopback interface, the interfaces connected to the other PE devices, and an IRB interface that is also configured as a default Layer 3 gateway.

```
[edit]
user@switch# set interfaces lo0 unit 0 family inet address 10.25.0.2/32
user@switch# set interfaces ge-3/1/5 unit 0 family inet address 10.0.4.2/30
user@switch# set interfaces ge-3/1/5 unit 0 family mpls
user@switch# set interfaces ge-3/1/8 unit 0 family inet address 10.0.3.2/30
user@switch# set interfaces ge-3/1/8 unit 0 family mpls
user@switch# set interfaces irb unit 0 family inet address 10.5.5.1/24 virtual-gateway-address 10.5.5.5
```

2. Configure the extended ports with EVPN multihoming in active-active mode, an ESI, and map the ports to VLAN v100..

```
[edit]
user@switch# set interfaces ge-108/0/25 unit 0 esi 00:01:02:03:04:00:01:02:04:26
user@switch# set interfaces ge-108/0/25 unit 0 esi all-active
```

```

user@switch# set interfaces ge-108/0/25 unit 0 family ethernet-switching vlan members v100
user@switch# set interfaces ge-108/0/27 unit 0 esi 00:01:02:03:04:00:01:02:04:28
user@switch# set interfaces ge-108/0/27 unit 0 esi all-active
user@switch# set interfaces ge-108/0/27 unit 0 family ethernet-switching vlan members v100

```

3. Assign a router ID and the autonomous system in which PE1, PE2, and PE3 reside.

```

[edit]
user@switch# set routing-options router-id 10.25.0.2
user@switch# set routing-options autonomous-system 100

```

4. Enable MPLS on the loopback interface and interfaces ge-3/1/5.0 and ge-3/1/8.0.

```

[edit]
user@switch# set protocols mpls interface lo0.0
user@switch# set protocols mpls interface ge-3/1/5.0
user@switch# set protocols mpls interface ge-3/1/8.0

```

5. Configure an IBGP overlay that includes PE1, PE2, and PE3.

```

[edit]
user@switch# set protocols bgp local-address 10.25.0.2
user@switch# set protocols bgp peer-as 100
user@switch# set protocols bgp local-as 100
user@switch# set protocols bgp group evpn-mes type internal
user@switch# set protocols bgp group evpn-mes family evpn signaling
user@switch# set protocols bgp group evpn-mes peer-as 100
user@switch# set protocols bgp group evpn-mes neighbor 10.25.0.1
user@switch# set protocols bgp group evpn-mes neighbor 10.25.0.3

```

6. Configure OSPF as the internal routing protocol for EVPN by specifying an area ID and interfaces on which EVPN-MPLS is enabled.

```

[edit]
user@switch# set protocols ospf traffic-engineering
user@switch# set protocols ospf area 0.0.0.0 interface ge-3/1/5.0
user@switch# set protocols ospf area 0.0.0.0 interface lo0.0

```

```

user@switch# set protocols ospf area 0.0.0.0 interface fxp0.0 disable
user@switch# set protocols ospf area 0.0.0.0 interface ge-3/1/8.0

```

7. Configure the LDP on the loopback interface and the interfaces on which EVPN-MPLS is enabled.

```

[edit]
user@switch# set protocols ldp interface lo0.0
user@switch# set protocols ldp interface ge-3/1/5.0
user@switch# set protocols ldp interface ge-3/1/8.0

```

8. Configure a virtual switch routing instance for VLAN v100, and include the interfaces and other entities associated with the VLAN.

```

[edit]
user@switch# set routing-instances evpn1 instance-type virtual-switch
user@switch# set routing-instances evpn1 interface ge-108/0/25.0
user@switch# set routing-instances evpn1 interface ge-108/0/27.0
user@switch# set routing-instances evpn1 interface ge-3/1/9.0
user@switch# set routing-instances evpn1 route-distinguisher 10.25.0.2:1
user@switch# set routing-instances evpn1 vrf-target target:100:1
user@switch# set routing-instances evpn1 protocols evpn label-allocation per-instance
user@switch# set routing-instances evpn1 protocols evpn extended-vlan-list 100
user@switch# set routing-instances evpn1 protocols evpn mclag bgp-peer 10.25.0.1
user@switch# set routing-instances evpn1 switch-options service-id 2
user@switch# set routing-instances evpn1 vlans v100 vlan-id 100
user@switch# set routing-instances evpn1 vlans v100 l3-interface irb.0

```

PE3 Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1868](#)
- [PE3: Configuring EVPN-MPLS | 1869](#)

CLI Quick Configuration

PE3: EVPN-MPLS Configuration

```
set interfaces lo0 unit 0 family inet address 10.25.0.3/32
set interfaces ge-0/3/5 unit 0 family inet address 10.0.1.2/30
set interfaces ge-0/3/5 unit 0 family mpls
set interfaces ge-0/3/7 unit 0 family inet address 10.0.4.1/30
set interfaces ge-0/3/7 unit 0 family mpls
set interfaces ge-0/0/46 unit 0 esi 00:01:02:03:04:00:01:02:04:12
set interfaces ge-0/0/46 unit 0 esi all-active
set interfaces ge-0/0/46 unit 0 family ethernet-switching vlan members 100
set routing-options router-id 10.25.0.3
set routing-options autonomous-system 100
set routing-options forwarding-table export evpn-pplb
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set protocols mpls interface lo0.0
set protocols mpls interface ge-0/3/5.0
set protocols mpls interface ge-0/3/7.0
set protocols bgp local-address 10.25.0.3
set protocols bgp peer-as 100
set protocols bgp local-as 100
set protocols bgp group evpn-mes type internal
set protocols bgp group evpn-mes family evpn signaling
set protocols bgp group evpn-mes peer-as 100
set protocols bgp group evpn-mes neighbor 10.25.0.2
set protocols bgp group evpn-mes neighbor 10.25.0.1
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-0/3/5.0
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 ospf area 0.0.0.0 interface ge-0/3/7.0
set protocols ldp interface lo0.0
set protocols ldp interface ge-0/3/5.0
set protocols ldp interface ge-0/3/7.0
set routing-instances evpn1 instance-type virtual-switch
set routing-instances evpn1 interface ge-0/0/46.0
set routing-instances evpn1 route-distinguisher 10.25.0.3:1
set routing-instances evpn1 vrf-target target:100:1
set routing-instances evpn1 protocols evpn label-allocation per-instance
set routing-instances evpn1 protocols evpn extended-vlan-list 100
```

```
set routing-instances evpn1 switch-options service-id 2
set routing-instances evpn1 vlans v100 vlan-id 100
```

PE3: Configuring EVPN-MPLS

Step-by-Step Procedure

1. Configure the interfaces on EVPN-MPLS interworking occurs.

```
[edit]
user@switch# set interfaces lo0 unit 0 family inet address 10.25.0.3/32
user@switch# set interfaces ge-0/3/5 unit 0 family inet address 10.0.1.2/30
user@switch# set interfaces ge-0/3/5 unit 0 family mpls
user@switch# set interfaces ge-0/3/7 unit 0 family inet address 10.0.4.1/30
user@switch# set interfaces ge-0/3/7 unit 0 family mpls
```

2. Configure interface ge-0/0/46 with EVPN multihoming in active-active mode, an ESI, and map the ports to VLAN v100..

```
[edit]
user@switch# set interfaces ge-0/0/46 unit 0 esi 00:01:02:03:04:00:01:02:04:12
user@switch# set interfaces ge-0/0/46 unit 0 esi all-active
user@switch# set interfaces ge-0/0/46 unit 0 family ethernet-switching vlan members 100
```

3. Assign a router ID and the autonomous system in which the PE1, PE2, and PE3 reside.

```
[edit]
user@switch# set routing-options router-id 10.25.0.2
user@switch# set routing-options autonomous-system 100
```

4. Enable per-packet load-balancing for EVPN routes when EVPN multihoming active-active mode is used.

```
[edit]
user@switch# set routing-options forwarding-table export evpn-pplb
user@switch# set policy-options policy-statement evpn-pplb from protocol evpn
user@switch# set policy-options policy-statement evpn-pplb then load-balance per-packet
```

5. Enable MPLS on the loopback interface and interfaces ge-0/3/5.0 and ge-0/3/7.0.

```
[edit]
user@switch# set protocols mpls interface lo0.0
user@switch# set protocols mpls interface ge-0/3/5.0
user@switch# set protocols mpls interface ge-0/3/7.0
```

6. Configure an IBGP overlay that includes PE1, PE2, and PE3.

```
[edit]
user@switch# set protocols bgp local-address 10.25.0.3
user@switch# set protocols bgp peer-as 100
user@switch# set protocols bgp local-as 100
user@switch# set protocols bgp group evpn-mes type internal
user@switch# set protocols bgp group evpn-mes family evpn signaling
user@switch# set protocols bgp group evpn-mes peer-as 100
user@switch# set protocols bgp group evpn-mes neighbor 10.25.0.2
user@switch# set protocols bgp group evpn-mes neighbor 10.25.0.1
```

7. Configure OSPF as the internal routing protocol for EVPN by specifying an area ID and interfaces on which EVPN-MPLS is enabled.

```
[edit]
user@switch# set protocols ospf traffic-engineering
user@switch# set protocols ospf area 0.0.0.0 interface ge-0/3/5.0
user@switch# set protocols ospf area 0.0.0.0 interface lo0.0
user@switch# set protocols ospf area 0.0.0.0 interface fxp0.0 disable
user@switch# set protocols ospf area 0.0.0.0 interface ge-0/3/7.0
```

8. Configure the LDP on the loopback interface and the interfaces on which EVPN-MPLS is enabled.

```
[edit]
user@switch# set protocols ldp interface lo0.0
user@switch# set protocols ldp interface ge-0/3/5.0
user@switch# set protocols ldp interface ge-0/3/7.0
```

9. Configure a virtual switch routing instance for VLAN v100, and include the interfaces and other entities associated with the VLAN.

```
[edit]
user@switch# set routing-instances evpn1 instance-type virtual-switch
user@switch# set routing-instances evpn1 interface ge-0/0/46.0
user@switch# set routing-instances evpn1 route-distinguisher 10.25.0.3:1
user@switch# set routing-instances evpn1 vrf-target target:100:1
user@switch# set routing-instances evpn1 protocols evpn label-allocation per-instance
user@switch# set routing-instances evpn1 protocols evpn extended-vlan-list 100
user@switch# set routing-instances evpn1 switch-options service-id 2
user@switch# set routing-instances evpn1 vlans v100 vlan-id 100
```

RELATED DOCUMENTATION

[Understanding EVPN-MPLS Interworking with Junos Fusion Enterprise and MC-LAG | 1845](#)

Example: EVPN-MPLS Interworking With an MC-LAG Topology

IN THIS SECTION

- [Requirements | 1872](#)
- [Overview and Topology | 1872](#)
- [PE1 and PE2 Configuration | 1875](#)
- [PE3 Configuration | 1891](#)

This example shows how to use Ethernet VPN (EVPN) to extend a multichassis link aggregation (MC-LAG) network over an MPLS network to a data center network or geographically distributed campus network.

EVPN-MPLS interworking is supported with an MC-LAG topology in which two MX Series routers, two EX9200 switches, or a mix of the two Juniper Networks devices function as MC-LAG peers, which use the Inter-Chassis Control Protocol (ICCP) and an interchassis link (ICL) to connect and maintain the

topology. The MC-LAG peers are connected to a provider edge (PE) device in an MPLS network. The PE device can be either an MX Series router or an EX9200 switch.

This example shows how to configure the MC-LAG peers and PE device in the MPLS network to interwork with each other.

Requirements

This example uses the following hardware and software components:

- Three EX9200 switches:
 - PE1 and PE2, which both function as MC-LAG peers in the MC-LAG topology and EVPN BGP peers in the EVPN-MPLS overlay network.
 - PE3, which functions as an EVPN BGP peer in the EVPN-MPLS overlay network.
- The EX9200 switches are running Junos OS Release 17.4R1 or later software.

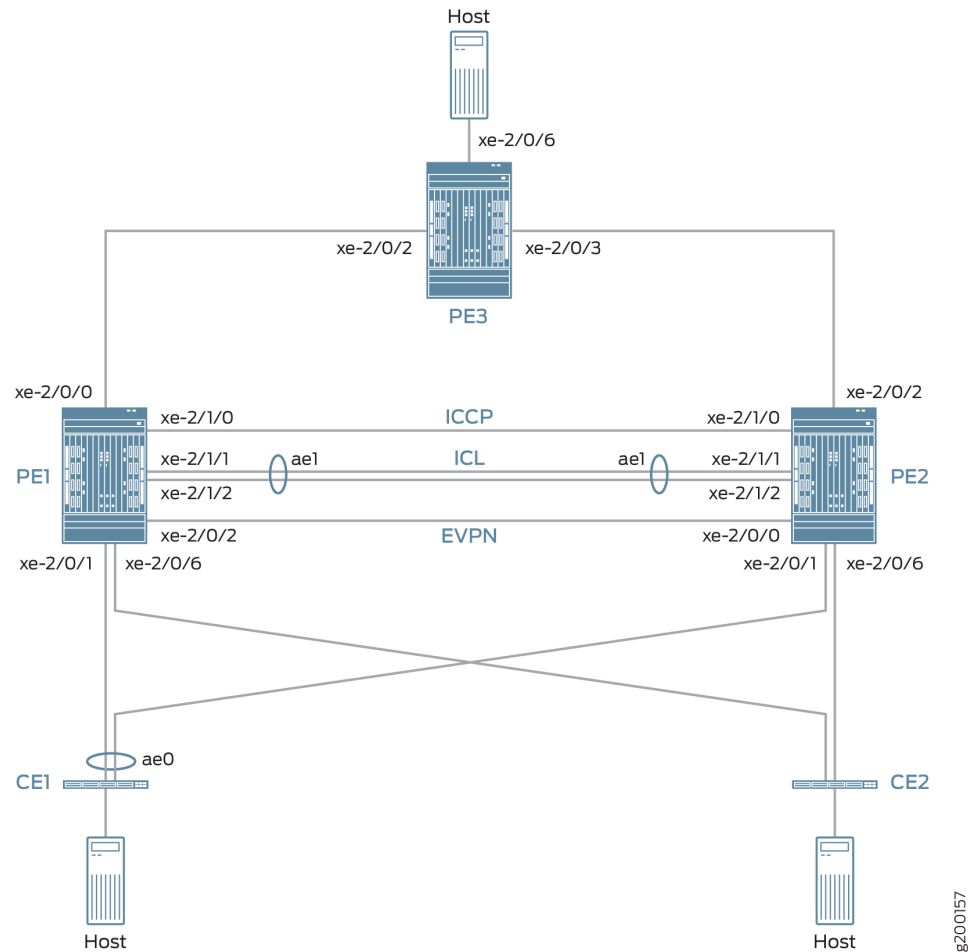


NOTE: Although the MC-LAG topology includes two customer edge (CE) devices, this example focuses on the configuration of the PE1, PE2, and PE3.

Overview and Topology

[Figure 192 on page 1873](#) shows an MC-LAG topology with provider edge devices PE1 and PE2 that are configured as MC-LAG peers. The MC-LAG peers exchange control information over an ICCP link and data traffic over an ICL. In this example, the ICL is an aggregated Ethernet interface that is comprised of two interfaces.

Figure 192: EVPN-MPLS Interworking With an MC-LAG Topology



The topology in [Figure 192 on page 1873](#) also includes CE devices CE1 and CE2, which are both multihomed to each PE device. The links between CE1 and the two PE devices are bundled as an aggregated Ethernet interface on which MC-LAG in active-active mode is configured.

The topology in [Figure 192 on page 1873](#) also includes PE3 at the edge of an MPLS network. PE3 functions as the gateway between the MC-LAG network and either a data center or a geographically distributed campus network. PE1, PE2, and PE3 run EVPN, which enables hosts in the MC-LAG network to communicate with hosts in the data center or other campus network by way of an intervening MPLS network.

From the perspective of the EVPN-MPLS interworking feature, PE3 functions solely as an EVPN BGP peer, and PE1 and PE2 in the MC-LAG topology have dual roles:

- MC-LAG peers in the MC-LAG network.
- EVPN BGP peers in the EVPN-MPLS network.

Because of the dual roles, PE1 and PE2 are configured with MC-LAG, EVPN, BGP, and MPLS attributes.

[Table 95 on page 1874](#) outlines key MC-LAG and EVPN (BGP and MPLS) attributes configured on PE1, PE2, and PE3.

Table 95: Key MC-LAG and EVPN (BGP and MPLS) Attributes Configured on PE1, PE2, and PE3

Key Attributes	PE1	PE2	PE3
MC-LAG Attributes			
Interfaces	ICL: aggregated Ethernet interface ae1, which is comprised of xe-2/1/1 and xe-2/1/2 ICCP: xe-2/1/0	ICL: aggregated Ethernet interface ae1, which is comprised of xe-2/1/1 and xe-2/1/2 ICCP: xe-2/1/0	Not applicable
EVPN-MPLS			
Interfaces	Connection to PE3: xe-2/0/0 Connection to PE2: xe-2/0/2	Connection to PE3: xe-2/0/2 Connection to PE1: xe-2/0/0	Connection to PE1: xe-2/0/2 Connection to PE2: xe-2/0/3
IP addresses	BGP peer address: 198.51.100.1	BGP peer address: 198.51.100.2	BGP peer address: 198.51.100.3
Autonomous system	65000	65000	65000
Virtual switch routing instances	evpn1, evpn2, evpn3	evpn1, evpn2, evpn3	evpn1, evpn2, evpn3

Note the following about the EVPN-MPLS interworking feature and its configuration:

- You must configure Ethernet segment identifiers (ESIs) on the dual-homed interfaces in the MC-LAG topology. The ESIs enable EVPN to identify the dual-homed interfaces.
- The only type of routing instance that is supported is the virtual switch instance (set `routing-instances name instance-type virtual-switch`).

- On the MC-LAG peers, you must include the `bgp-peer` configuration statement in the `[edit routing-instances name protocols evpn mclag]` hierarchy level. This configuration statement enables the interworking of EVPN-MPLS with MC-LAG on the MC-LAG peers.
- Address Resolution Protocol (ARP) suppression is not supported.

PE1 and PE2 Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1875](#)
- [PE1: Configuring MC-LAG | 1881](#)
- [PE1: Configuring EVPN-MPLS | 1883](#)
- [PE2: Configuring MC-LAG | 1886](#)
- [PE2: Configuring EVPN-MPLS | 1888](#)

To configure PE1 and PE2, perform these tasks:

CLI Quick Configuration

PE1: MC-LAG Configuration

```
set chassis aggregated-devices ethernet device-count 3
set interfaces xe-2/0/1 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:11:11:11:11
set interfaces ae0 aggregated-ether-options lacp admin-key 1
set interfaces ae0 aggregated-ether-options mc-ae mc-ae-id 1
set interfaces ae0 aggregated-ether-options mc-ae redundancy-group 2
set interfaces ae0 aggregated-ether-options mc-ae chassis-id 0
set interfaces ae0 aggregated-ether-options mc-ae mode active-active
set interfaces ae0 aggregated-ether-options mc-ae status-control active
set interfaces ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 unit 1 esi all-active
set interfaces ae0 unit 1 family ethernet-switching interface-mode trunk
```

```

set interfaces ae0 unit 1 family ethernet-switching vlan members 1
set interfaces ae0 unit 2 esi 00:11:11:11:11:11:11:11
set interfaces ae0 unit 2 esi all-active
set interfaces ae0 unit 2 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 2 family ethernet-switching vlan members 2
set interfaces ae0 unit 3 esi 00:11:22:22:22:22:22:22
set interfaces ae0 unit 3 esi all-active
set interfaces ae0 unit 3 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 3 family ethernet-switching vlan members 3
set interfaces xe-2/0/6 enable
set interfaces xe-2/0/6 flexible-vlan-tagging
set interfaces xe-2/0/6 encapsulation flexible-ethernet-services
set interfaces xe-2/0/6 unit 1 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 1 family ethernet-switching vlan members 1
set interfaces xe-2/0/6 unit 2 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 2 family ethernet-switching vlan members 2
set interfaces xe-2/0/6 unit 3 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 3 family ethernet-switching vlan members 3
set interfaces xe-2/1/0 unit 0 family inet address 203.0.113.1/24
set interfaces xe-2/1/1 gigether-options 802.3ad ae1
set interfaces xe-2/1/2 gigether-options 802.3ad ae1
set interfaces ae1 flexible-vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 1 family ethernet-switching vlan members 1
set interfaces ae1 unit 2 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 2 family ethernet-switching vlan members 2
set interfaces ae1 unit 3 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 3 family ethernet-switching vlan members 3
set multi-chassis multi-chassis-protection 203.0.113.2 interface ae1
set protocols iccp local-ip-addr 203.0.113.1
set protocols iccp peer 203.0.113.2 session-establishment-hold-time 600
set protocols iccp peer 203.0.113.2 redundancy-group-id-list 2
set protocols iccp peer 203.0.113.2 liveness-detection minimum-interval 10000
set protocols iccp peer 203.0.113.2 liveness-detection multiplier 3

```

PE1: EVPN-MPLS Configuration

```

set interfaces lo0 unit 0 family inet address 198.51.100.1/32 primary
set interfaces xe-2/0/0 unit 0 family inet address 192.0.2.2/24
set interfaces xe-2/0/0 unit 0 family mpls

```

```

set interfaces xe-2/0/2 unit 0 family inet address 192.0.2.111/24
set interfaces xe-2/0/2 unit 0 family mpls
set interfaces irb unit 1 family inet address 10.2.1.1/24 virtual-gateway-address 10.2.1.254
set interfaces irb unit 2 family inet address 10.2.2.1/24 virtual-gateway-address 10.2.2.254
set interfaces irb unit 3 family inet address 10.2.3.1/24 virtual-gateway-address 10.2.3.254
set routing-options router-id 198.51.100.1
set routing-options autonomous-system 65000
set routing-options forwarding-table export evpn-pplb
set protocols mpls interface xe-2/0/0.0
set protocols mpls interface xe-2/0/2.0
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 198.51.100.1
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn local-as 65000
set protocols bgp group evpn neighbor 198.51.100.2
set protocols bgp group evpn neighbor 198.51.100.3
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface xe-2/0/0.0
set protocols ospf area 0.0.0.0 interface xe-2/0/2.0
set protocols ldp interface xe-2/0/0.0
set protocols ldp interface xe-2/0/2.0
set protocols ldp interface lo0.0
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set routing-instances evpn1 instance-type virtual-switch
set routing-instances evpn1 interface xe-2/0/6.1
set routing-instances evpn1 interface ae0.1
set routing-instances evpn1 interface ae1.1
set routing-instances evpn1 route-distinguisher 1:10
set routing-instances evpn1 vrf-target target:1:5
set routing-instances evpn1 protocols evpn extended-vlan-list 1
set routing-instances evpn1 protocols evpn mlag bgp-peer 198.51.100.2
set routing-instances evpn1 switch-options service-id 1
set routing-instances evpn1 vlans v1 vlan-id 1
set routing-instances evpn1 vlans v1 l3-interface irb.1
set routing-instances evpn2 instance-type virtual-switch
set routing-instances evpn2 interface xe-2/0/6.2
set routing-instances evpn2 interface ae0.2
set routing-instances evpn2 interface ae1.2
set routing-instances evpn2 route-distinguisher 1:20
set routing-instances evpn2 vrf-target target:1:6
set routing-instances evpn2 protocols evpn extended-vlan-list 2
set routing-instances evpn2 protocols evpn mlag bgp-peer 198.51.100.2

```

```

set routing-instances evpn2 switch-options service-id 2
set routing-instances evpn2 vlans v1 vlan-id 2
set routing-instances evpn2 vlans v1 l3-interface irb.2
set routing-instances evpn3 instance-type virtual-switch
set routing-instances evpn3 interface xe-2/0/6.3
set routing-instances evpn3 interface ae0.3
set routing-instances evpn3 interface ae1.3
set routing-instances evpn3 route-distinguisher 1:30
set routing-instances evpn3 vrf-target target:1:7
set routing-instances evpn3 protocols evpn extended-vlan-list 3
set routing-instances evpn3 protocols evpn mclag bgp-peer 198.51.100.2
set routing-instances evpn3 switch-options service-id 3
set routing-instances evpn3 vlans v1 vlan-id 3
set routing-instances evpn3 vlans v1 l3-interface irb.3

```

PE2: MC-LAG Configuration

```

set chassis aggregated-devices ethernet device-count 3
set interfaces xe-2/0/1 gigether-options 802.3ad ae0
set interfaces xe-2/0/6 enable
set interfaces xe-2/0/6 flexible-vlan-tagging
set interfaces xe-2/0/6 encapsulation flexible-ethernet-services
set interfaces xe-2/0/6 unit 1 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 1 family ethernet-switching vlan members 1
set interfaces xe-2/0/6 unit 2 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 2 family ethernet-switching vlan members 2
set interfaces xe-2/0/6 unit 3 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 3 family ethernet-switching vlan members 3
set interfaces xe-2/1/0 unit 0 family inet address 203.0.113.2/24
set interfaces xe-2/1/1 gigether-options 802.3ad ae1
set interfaces xe-2/1/2 gigether-options 802.3ad ae1
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp periodic fast
set interfaces ae0 aggregated-ether-options lacp system-id 00:00:11:11:11:11
set interfaces ae0 aggregated-ether-options lacp admin-key 1
set interfaces ae0 aggregated-ether-options mc-ae mc-ae-id 1
set interfaces ae0 aggregated-ether-options mc-ae redundancy-group 2
set interfaces ae0 aggregated-ether-options mc-ae chassis-id 1
set interfaces ae0 aggregated-ether-options mc-ae mode active-active
set interfaces ae0 aggregated-ether-options mc-ae status-control standby

```

```

set interfaces ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae0 unit 1 esi all-active
set interfaces ae0 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 1 family ethernet-switching vlan members 1
set interfaces ae0 unit 2 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 unit 2 esi all-active
set interfaces ae0 unit 2 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 2 family ethernet-switching vlan members 2
set interfaces ae0 unit 3 esi 00:11:22:22:22:22:22:22:22:22
set interfaces ae0 unit 3 esi all-active
set interfaces ae0 unit 3 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 3 family ethernet-switching vlan members 3
set interfaces ae1 flexible-vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 1 family ethernet-switching vlan members 1
set interfaces ae1 unit 2 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 2 family ethernet-switching vlan members 2
set interfaces ae1 unit 3 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 3 family ethernet-switching vlan members 3
set multi-chassis multi-chassis-protection 203.0.113.1 interface ae1
set protocols iccp local-ip-addr 203.0.113.2
set protocols iccp peer 203.0.113.1 session-establishment-hold-time 600
set protocols iccp peer 203.0.113.1 redundancy-group-id-list 2
set protocols iccp peer 203.0.113.1 liveness-detection minimum-interval 10000
set protocols iccp peer 203.0.113.1 liveness-detection multiplier 3

```

PE2: EVPN-MPLS Configuration

```

set interfaces xe-2/0/0 unit 0 family inet address 192.0.2.222/24
set interfaces xe-2/0/0 unit 0 family mpls
set interfaces xe-2/0/2 unit 0 family inet address 192.0.2.22/24
set interfaces xe-2/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 198.51.100.2/32 primary
set interfaces irb unit 1 family inet address 10.2.1.2/24 virtual-gateway-address 10.2.1.254
set interfaces irb unit 2 family inet address 10.2.2.2/24 virtual-gateway-address 10.2.2.254
set interfaces irb unit 3 family inet address 10.2.3.2/24 virtual-gateway-address 10.2.3.254
set routing-options router-id 198.51.100.2
set routing-options autonomous-system 65000
set routing-options forwarding-table export evpn-pplb
set protocols mpls interface xe-2/0/2.0

```



```

set protocols mpls interface xe-2/0/0.0
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 198.51.100.2
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn local-as 65000
set protocols bgp group evpn neighbor 198.51.100.1
set protocols bgp group evpn neighbor 198.51.100.3
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface xe-2/0/0.0
set protocols ospf area 0.0.0.0 interface xe-2/0/2.0
set protocols ldp interface xe-2/0/0.0
set protocols ldp interface xe-2/0/2.0
set protocols ldp interface lo0.0
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set routing-instances evpn1 instance-type virtual-switch
set routing-instances evpn1 interface xe-2/0/6.1
set routing-instances evpn1 interface ae0.1
set routing-instances evpn1 interface ae1.1
set routing-instances evpn1 route-distinguisher 1:11
set routing-instances evpn1 vrf-target target:1:5
set routing-instances evpn1 protocols evpn extended-vlan-list 1
set routing-instances evpn1 protocols evpn mlag bgp-peer 198.51.100.1
set routing-instances evpn1 switch-options service-id 1
set routing-instances evpn1 vlans v1 vlan-id 1
set routing-instances evpn1 vlans v1 l3-interface irb.1
set routing-instances evpn2 instance-type virtual-switch
set routing-instances evpn2 interface xe-2/0/6.2
set routing-instances evpn2 interface ae0.2
set routing-instances evpn2 interface ae1.2
set routing-instances evpn2 route-distinguisher 1:21
set routing-instances evpn2 vrf-target target:1:6
set routing-instances evpn2 protocols evpn extended-vlan-list 2
set routing-instances evpn2 protocols evpn mlag bgp-peer 198.51.100.1
set routing-instances evpn2 switch-options service-id 2
set routing-instances evpn2 vlans v1 vlan-id 2
set routing-instances evpn2 vlans v1 l3-interface irb.2
set routing-instances evpn3 instance-type virtual-switch
set routing-instances evpn3 interface xe-2/0/6.3
set routing-instances evpn3 interface ae0.3
set routing-instances evpn3 interface ae1.3
set routing-instances evpn3 route-distinguisher 1:31
set routing-instances evpn3 vrf-target target:1:7

```

```

set routing-instances evpn3 protocols evpn extended-vlan-list 3
set routing-instances evpn3 protocols evpn mclag bgp-peer 198.51.100.1
set routing-instances evpn3 switch-options service-id 3
set routing-instances evpn3 vlans v1 vlan-id 3
set routing-instances evpn3 vlans v1 l3-interface irb.3

```

PE1: Configuring MC-LAG

Step-by-Step Procedure

1. Set the number of aggregated Ethernet interfaces on PE1.

```

[edit]
user@switch# set chassis aggregated-devices ethernet device-count 3

```

2. Configure aggregated Ethernet interface ae0 on interface xe-2/0/1, and configure LACP and MC-LAG on ae0. Divide aggregated Ethernet interface ae0 into three logical interfaces (ae0.1, ae0.2, and ae0.3). For each logical interface, specify an ESI, place the logical interface in MC-LAG active-active mode, and map the logical interface to a VLAN.

```

[edit]
user@switch# set interfaces xe-2/0/1 gigether-options 802.3ad ae0
user@switch# set interfaces ae0 flexible-vlan-tagging
user@switch# set interfaces ae0 encapsulation flexible-ethernet-services
user@switch# set interfaces ae0 aggregated-ether-options lacp active
user@switch# set interfaces ae0 aggregated-ether-options lacp periodic fast
user@switch# set interfaces ae0 aggregated-ether-options lacp system-id 00:00:11:11:11:11
user@switch# set interfaces ae0 aggregated-ether-options lacp admin-key 1
user@switch# set interfaces ae0 aggregated-ether-options mc-ae mc-ae-id 1
user@switch# set interfaces ae0 aggregated-ether-options mc-ae redundancy-group 2
user@switch# set interfaces ae0 aggregated-ether-options mc-ae chassis-id 0
user@switch# set interfaces ae0 aggregated-ether-options mc-ae mode active-active
user@switch# set interfaces ae0 aggregated-ether-options mc-ae status-control active
user@switch# set interfaces ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
user@switch# set interfaces ae0 unit 1 esi all-active
user@switch# set interfaces ae0 unit 1 family ethernet-switching interface-mode trunk
user@switch# set interfaces ae0 unit 1 family ethernet-switching vlan members 1
user@switch# set interfaces ae0 unit 2 esi 00:11:11:11:11:11:11:11:11:11
user@switch# set interfaces ae0 unit 2 esi all-active
user@switch# set interfaces ae0 unit 2 family ethernet-switching interface-mode trunk

```

```

user@switch# set interfaces ae0 unit 2 family ethernet-switching vlan members 2
user@switch# set interfaces ae0 unit 3 esi 00:11:22:22:22:22:22:22
user@switch# set interfaces ae0 unit 3 esi all-active
user@switch# set interfaces ae0 unit 3 family ethernet-switching interface-mode trunk
user@switch# set interfaces ae0 unit 3 family ethernet-switching vlan members 3

```

3. Configure physical interface xe-2/0/6, and divide it into three logical interfaces (xe-2/0/6.1, xe-2/0/6.2, and xe-2/0/6.3). Map each logical interface to a VLAN.

```

[edit]
user@switch# set interfaces xe-2/0/6 enable
user@switch# set interfaces xe-2/0/6 flexible-vlan-tagging
user@switch# set interfaces xe-2/0/6 encapsulation flexible-ethernet-services
user@switch# set interfaces xe-2/0/6 unit 1 family ethernet-switching interface-mode trunk
user@switch# set interfaces xe-2/0/6 unit 1 family ethernet-switching vlan members 1
user@switch# set interfaces xe-2/0/6 unit 2 family ethernet-switching interface-mode trunk
user@switch# set interfaces xe-2/0/6 unit 2 family ethernet-switching vlan members 2
user@switch# set interfaces xe-2/0/6 unit 3 family ethernet-switching interface-mode trunk
user@switch# set interfaces xe-2/0/6 unit 3 family ethernet-switching vlan members 3

```

4. Configure physical interface xe-2/1/0 as a Layer 3 interface, on which you configure ICCP. Specify the interface with the IP address of 203.0.113.2 on PE2 as the ICCP peer to PE1.

```

[edit]
user@switch# set interfaces xe-2/1/0 unit 0 family inet address 203.0.113.1/24
user@switch# set protocols iccp local-ip-addr 203.0.113.1
user@switch# set protocols iccp peer 203.0.113.2 session-establishment-hold-time 600
user@switch# set protocols iccp peer 203.0.113.2 redundancy-group-id-list 2
user@switch# set protocols iccp peer 203.0.113.2 liveness-detection minimum-interval 10000
user@switch# set protocols iccp peer 203.0.113.2 liveness-detection multiplier 3

```

5. Configure aggregated Ethernet interface ae1 on interfaces xe-2/1/1 and xe-2/1/2, and configure LACP on ae1. Divide aggregated Ethernet interface ae1 into three logical interfaces (ae1.1, ae1.2, and ae1.3), and map each logical interface to a VLAN. Specify ae1 as the multichassis protection link between PE1 and PE2.

```

[edit]
user@switch# set interfaces xe-2/1/1 gigether-options 802.3ad ae1
user@switch# set interfaces xe-2/1/2 gigether-options 802.3ad ae1
user@switch# set interfaces ae1 flexible-vlan-tagging

```

```

user@switch# set interfaces ae1 encapsulation flexible-ethernet-services
user@switch# set interfaces ae1 aggregated-ether-options lacp active
user@switch# set interfaces ae1 unit 1 family ethernet-switching interface-mode trunk
user@switch# set interfaces ae1 unit 1 family ethernet-switching vlan members 1
user@switch# set interfaces ae1 unit 2 family ethernet-switching interface-mode trunk
user@switch# set interfaces ae1 unit 2 family ethernet-switching vlan members 2
user@switch# set interfaces ae1 unit 3 family ethernet-switching interface-mode trunk
user@switch# set interfaces ae1 unit 3 family ethernet-switching vlan members 3
user@switch# set multi-chassis multi-chassis-protection 203.0.113.2 interface ae1

```

PE1: Configuring EVPN-MPLS

Step-by-Step Procedure

1. Configure the loopback interface, and the interfaces connected to the other PE devices.

```

[edit]
user@switch# set interfaces lo0 unit 0 family inet address 198.51.100.1/32 primary
user@switch# set interfaces xe-2/0/0 unit 0 family inet address 192.0.2.2/24
user@switch# set interfaces xe-2/0/0 unit 0 family mpls
user@switch# set interfaces xe-2/0/2 unit 0 family inet address 192.0.2.111/24
user@switch# set interfaces xe-2/0/2 unit 0 family mpls

```

2. Configure IRB interfaces irb.1, irb.2, and irb.3.

```

[edit]
user@switch# set interfaces irb unit 1 family inet address 10.2.1.1/24 virtual-gateway-address 10.2.1.254
user@switch# set interfaces irb unit 2 family inet address 10.2.2.1/24 virtual-gateway-address 10.2.2.254
user@switch# set interfaces irb unit 3 family inet address 10.2.3.1/24 virtual-gateway-address 10.2.3.254

```

3. Assign a router ID and the autonomous system in which PE1, PE2, and PE3 reside.

```

[edit]
user@switch# set routing-options router-id 198.51.100.1
user@switch# set routing-options autonomous-system 65000

```

4. Enable per-packet load-balancing for EVPN routes when EVPN multihoming active-active mode is used.

```
[edit]
user@switch# set routing-options forwarding-table export evpn-pplb
user@switch# set policy-options policy-statement evpn-pplb from protocol evpn
user@switch# set policy-options policy-statement evpn-pplb then load-balance per-packet
```

5. Enable MPLS on interfaces xe-2/0/0.0 and xe-2/0/2.0.

```
[edit]
user@switch# set protocols mpls interface xe-2/0/0.0
user@switch# set protocols mpls interface xe-2/0/2.0
```

6. Configure an IBGP overlay that includes PE1, PE2, and PE3.

```
[edit]
user@switch# set protocols bgp group evpn type internal
user@switch# set protocols bgp group evpn local-address 198.51.100.1
user@switch# set protocols bgp group evpn family evpn signaling
user@switch# set protocols bgp group evpn local-as 65000
user@switch# set protocols bgp group evpn neighbor 198.51.100.2
user@switch# set protocols bgp group evpn neighbor 198.51.100.3
```

7. Configure OSPF as the internal routing protocol for EVPN by specifying an area ID and interfaces on which EVPN-MPLS is enabled.

```
[edit]
user@switch# set protocols ospf area 0.0.0.0 interface lo0.0
user@switch# set protocols ospf area 0.0.0.0 interface xe-2/0/0.0
user@switch# set protocols ospf area 0.0.0.0 interface xe-2/0/2.0
```

8. Configure the Label Distribution Protocol (LDP) on the loopback interface and the interfaces on which EVPN-MPLS is enabled.

```
[edit]
user@switch# set protocols ldp interface lo0.0
```

```

user@switch# set protocols ldp interface xe-2/0/0.0
user@switch# set protocols ldp interface xe-2/0/2.0

```

9. Configure virtual switch routing instances for VLAN v1, which is assigned VLAN IDs of 1, 2, and 3, and include the interfaces and other entities associated with the VLAN.

```

[edit]
user@switch# set routing-instances evpn1 instance-type virtual-switch
user@switch# set routing-instances evpn1 interface xe-2/0/6.1
user@switch# set routing-instances evpn1 interface ae0.1
user@switch# set routing-instances evpn1 interface ae1.1
user@switch# set routing-instances evpn1 route-distinguisher 1:10
user@switch# set routing-instances evpn1 vrf-target target:1:5
user@switch# set routing-instances evpn1 protocols evpn extended-vlan-list 1
user@switch# set routing-instances evpn1 protocols evpn mclag bgp-peer 198.51.100.2
user@switch# set routing-instances evpn1 switch-options service-id 1
user@switch# set routing-instances evpn1 vlans v1 vlan-id 1
user@switch# set routing-instances evpn1 vlans v1 l3-interface irb.1
user@switch# set routing-instances evpn2 instance-type virtual-switch
user@switch# set routing-instances evpn2 interface xe-2/0/6.2
user@switch# set routing-instances evpn2 interface ae0.2
user@switch# set routing-instances evpn2 interface ae1.2
user@switch# set routing-instances evpn2 route-distinguisher 1:20
user@switch# set routing-instances evpn2 vrf-target target:1:6
user@switch# set routing-instances evpn2 protocols evpn extended-vlan-list 2
user@switch# set routing-instances evpn2 protocols evpn mclag bgp-peer 198.51.100.2
user@switch# set routing-instances evpn2 switch-options service-id 2
user@switch# set routing-instances evpn2 vlans v1 vlan-id 2
user@switch# set routing-instances evpn2 vlans v1 l3-interface irb.2
user@switch# set routing-instances evpn3 instance-type virtual-switch
user@switch# set routing-instances evpn3 interface xe-2/0/6.3
user@switch# set routing-instances evpn3 interface ae0.3
user@switch# set routing-instances evpn3 interface ae1.3
user@switch# set routing-instances evpn3 route-distinguisher 1:30
user@switch# set routing-instances evpn3 vrf-target target:1:7
user@switch# set routing-instances evpn3 protocols evpn extended-vlan-list 3
user@switch# set routing-instances evpn3 protocols evpn mclag bgp-peer 198.51.100.2
user@switch# set routing-instances evpn3 switch-options service-id 3
user@switch# set routing-instances evpn3 vlans v1 vlan-id 3
user@switch# set routing-instances evpn3 vlans v1 l3-interface irb.3

```

PE2: Configuring MC-LAG

Step-by-Step Procedure

1. Set the number of aggregated Ethernet interfaces on PE2.

```
[edit]
user@switch# set chassis aggregated-devices ethernet device-count 3
```

2. Configure aggregated Ethernet interface ae0 on interface xe-2/0/1, and configure LACP and MC-LAG on ae0. Divide aggregated Ethernet interface ae0 into three logical interfaces (ae0.1, ae0.2, and ae0.3). For each logical interface, specify an ESI, place the logical interface in MC-LAG active-active mode, and map the logical interface to a VLAN.

```
[edit]
user@switch# set interfaces xe-2/0/1 gigether-options 802.3ad ae0
user@switch# set interfaces ae0 flexible-vlan-tagging
user@switch# set interfaces ae0 encapsulation flexible-ethernet-services
user@switch# set interfaces ae0 aggregated-ether-options lacp active
user@switch# set interfaces ae0 aggregated-ether-options lacp periodic fast
user@switch# set interfaces ae0 aggregated-ether-options lacp system-id 00:00:11:11:11:11
user@switch# set interfaces ae0 aggregated-ether-options lacp admin-key 1
user@switch# set interfaces ae0 aggregated-ether-options mc-ae mc-ae-id 1
user@switch# set interfaces ae0 aggregated-ether-options mc-ae redundancy-group 2
user@switch# set interfaces ae0 aggregated-ether-options mc-ae chassis-id 1
user@switch# set interfaces ae0 aggregated-ether-options mc-ae mode active-active
user@switch# set interfaces ae0 aggregated-ether-options mc-ae status-control standby
user@switch# set interfaces ae0 unit 1 esi 00:11:22:33:44:55:66:77:88:99
user@switch# set interfaces ae0 unit 1 esi all-active
user@switch# set interfaces ae0 unit 1 family ethernet-switching interface-mode trunk
user@switch# set interfaces ae0 unit 1 family ethernet-switching vlan members 1
user@switch# set interfaces ae0 unit 2 esi 00:11:11:11:11:11:11:11:11:11
user@switch# set interfaces ae0 unit 2 esi all-active
user@switch# set interfaces ae0 unit 2 family ethernet-switching interface-mode trunk
user@switch# set interfaces ae0 unit 2 family ethernet-switching vlan members 2
user@switch# set interfaces ae0 unit 3 esi 00:11:22:22:22:22:22:22:22:22
user@switch# set interfaces ae0 unit 3 esi all-active
user@switch# set interfaces ae0 unit 3 family ethernet-switching interface-mode trunk
user@switch# set interfaces ae0 unit 3 family ethernet-switching vlan members 3
```

3. Configure physical interface xe-2/0/6, and divide it into three logical interfaces (xe-2/0/6.1, xe-2/0/6.2, and xe-2/0/6.3). Map each logical interface to a VLAN.

```
[edit]
set interfaces xe-2/0/6 enable
set interfaces xe-2/0/6 flexible-vlan-tagging
set interfaces xe-2/0/6 encapsulation flexible-ethernet-services
set interfaces xe-2/0/6 unit 1 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 1 family ethernet-switching vlan members 1
set interfaces xe-2/0/6 unit 2 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 2 family ethernet-switching vlan members 2
set interfaces xe-2/0/6 unit 3 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 3 family ethernet-switching vlan members 3
```

4. Configure physical interface xe-2/1/0 as a Layer 3 interface, on which you configure ICCP. Specify the interface with the IP address of 203.0.113.1 on PE1 as the ICCP peer to PE2.

```
[edit]
set interfaces xe-2/1/0 unit 0 family inet address 203.0.113.2/24
set protocols iccp local-ip-addr 203.0.113.2
set protocols iccp peer 203.0.113.1 session-establishment-hold-time 600
set protocols iccp peer 203.0.113.1 redundancy-group-id-list 2
set protocols iccp peer 203.0.113.1 liveness-detection minimum-interval 10000
set protocols iccp peer 203.0.113.1 liveness-detection multiplier 3
```

5. Configure aggregated Ethernet interface ae1 on interfaces xe-2/1/1 and xe-2/1/2, and configure LACP on ae1. Divide aggregated Ethernet interface ae1 into three logical interfaces (ae1.1, ae1.2, and ae1.3), and map each logical interface to a VLAN. Specify ae1 as the multichassis protection link between PE1 and PE2.

```
[edit]
set interfaces xe-2/1/1 gigether-options 802.3ad ae1
set interfaces xe-2/1/2 gigether-options 802.3ad ae1
set interfaces ae1 flexible-vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 unit 1 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 1 family ethernet-switching vlan members 1
set interfaces ae1 unit 2 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 2 family ethernet-switching vlan members 2
```



```

set interfaces ae1 unit 3 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 3 family ethernet-switching vlan members 3
set multi-chassis multi-chassis-protection 203.0.113.1 interface ae1

```

PE2: Configuring EVPN-MPLS

Step-by-Step Procedure

1. Configure the loopback interface, and the interfaces connected to the other PE devices.

```

[edit]
user@switch# set interfaces lo0 unit 0 family inet address 198.51.100.2/32 primary
user@switch# set interfaces xe-2/0/0 unit 0 family inet address 192.0.2.222/24
user@switch# set interfaces xe-2/0/0 unit 0 family mpls
user@switch# set interfaces xe-2/0/2 unit 0 family inet address 192.0.2.22/24
user@switch# set interfaces xe-2/0/2 unit 0 family mpls

```

2. Configure IRB interfaces irb.1, irb.2, and irb.3.

```

[edit]
user@switch# set interfaces irb unit 1 family inet address 10.2.1.2/24 virtual-gateway-
address 10.2.1.254
user@switch# set interfaces irb unit 2 family inet address 10.2.2.2/24 virtual-gateway-
address 10.2.2.254
user@switch# set interfaces irb unit 3 family inet address 10.2.3.2/24 virtual-gateway-
address 10.2.3.254

```

3. Assign a router ID and the autonomous system in which PE1, PE2, and PE3 reside.

```

[edit]
user@switch# set routing-options router-id 198.51.100.2
user@switch# set routing-options autonomous-system 65000

```

4. Enable per-packet load-balancing for EVPN routes when EVPN multihoming active-active mode is used.

```

[edit]
user@switch# set routing-options forwarding-table export evpn-pplb

```

```

user@switch# set policy-options policy-statement evpn-pplb from protocol evpn
user@switch# set policy-options policy-statement evpn-pplb then load-balance per-packet

```

5. Enable MPLS on interfaces xe-2/0/0.0 and xe-2/0/2.0.

```

[edit]
user@switch# set protocols mpls interface xe-2/0/0.0
user@switch# set protocols mpls interface xe-2/0/2.0

```

6. Configure an IBGP overlay that includes PE1, PE2, and PE3.

```

[edit]
user@switch# set protocols bgp group evpn type internal
user@switch# set protocols bgp group evpn local-address 198.51.100.2
user@switch# set protocols bgp group evpn family evpn signaling
user@switch# set protocols bgp group evpn local-as 65000
user@switch# set protocols bgp group evpn neighbor 198.51.100.1
user@switch# set protocols bgp group evpn neighbor 198.51.100.3

```

7. Configure OSPF as the internal routing protocol for EVPN by specifying an area ID and interfaces on which EVPN-MPLS is enabled.

```

[edit]
user@switch# set protocols ospf area 0.0.0.0 interface lo0.0
user@switch# set protocols ospf area 0.0.0.0 interface xe-2/0/0.0
user@switch# set protocols ospf area 0.0.0.0 interface xe-2/0/2.0

```

8. Configure the Label Distribution Protocol (LDP) on the loopback interface and the interfaces on which EVPN-MPLS is enabled.

```

[edit]
user@switch# set protocols ldp interface lo0.0
user@switch# set protocols ldp interface xe-2/0/0.0
user@switch# set protocols ldp interface xe-2/0/2.0

```

9. Configure virtual switch routing instances for VLAN v1, which is assigned VLAN IDs of 1, 2, and 3, and include the interfaces and other entities associated with the VLAN.

```
[edit]
user@switch# set routing-instances evpn1 instance-type virtual-switch
user@switch# set routing-instances evpn1 interface xe-2/0/6.1
user@switch# set routing-instances evpn1 interface ae0.1
user@switch# set routing-instances evpn1 interface ae1.1
user@switch# set routing-instances evpn1 route-distinguisher 1:11
user@switch# set routing-instances evpn1 vrf-target target:1:5
user@switch# set routing-instances evpn1 protocols evpn extended-vlan-list 1
user@switch# set routing-instances evpn1 protocols evpn mclag bgp-peer 198.51.100.1
user@switch# set routing-instances evpn1 switch-options service-id 1
user@switch# set routing-instances evpn1 vlans v1 vlan-id 1
user@switch# set routing-instances evpn1 vlans v1 l3-interface irb.1
user@switch# set routing-instances evpn2 instance-type virtual-switch
user@switch# set routing-instances evpn2 interface xe-2/0/6.2
user@switch# set routing-instances evpn2 interface ae0.2
user@switch# set routing-instances evpn2 interface ae1.2
user@switch# set routing-instances evpn2 route-distinguisher 1:21
user@switch# set routing-instances evpn2 vrf-target target:1:6
user@switch# set routing-instances evpn2 protocols evpn extended-vlan-list 2
user@switch# set routing-instances evpn2 protocols evpn mclag bgp-peer 198.51.100.1
user@switch# set routing-instances evpn2 switch-options service-id 2
user@switch# set routing-instances evpn2 vlans v1 vlan-id 2
user@switch# set routing-instances evpn2 vlans v1 l3-interface irb.2
user@switch# set routing-instances evpn3 instance-type virtual-switch
user@switch# set routing-instances evpn3 interface xe-2/0/6.3
user@switch# set routing-instances evpn3 interface ae0.3
user@switch# set routing-instances evpn3 interface ae1.3
user@switch# set routing-instances evpn3 route-distinguisher 1:31
user@switch# set routing-instances evpn3 vrf-target target:1:7
user@switch# set routing-instances evpn3 protocols evpn extended-vlan-list 3
user@switch# set routing-instances evpn3 protocols evpn mclag bgp-peer 198.51.100.1
user@switch# set routing-instances evpn3 switch-options service-id 3
user@switch# set routing-instances evpn3 vlans v1 vlan-id 3
user@switch# set routing-instances evpn3 vlans v1 l3-interface irb.3
```

PE3 Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1891](#)
- [PE3: Configuring EVPN-MPLS | 1892](#)

CLI Quick Configuration

PE3: EVPN-MPLS Configuration

```

set interfaces lo0 unit 0 family inet address 198.51.100.3/32 primary
set interfaces xe-2/0/2 unit 0 family inet address 192.0.2.1/24
set interfaces xe-2/0/2 unit 0 family mpls
set interfaces xe-2/0/3 unit 0 family inet address 192.0.2.11/24
set interfaces xe-2/0/3 unit 0 family mpls
set interfaces xe-2/0/6 enable
set interfaces xe-2/0/6 flexible-vlan-tagging
set interfaces xe-2/0/6 encapsulation flexible-ethernet-services
set interfaces xe-2/0/6 unit 1 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 1 family ethernet-switching vlan members 1
set interfaces xe-2/0/6 unit 2 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 2 family ethernet-switching vlan members 2
set interfaces xe-2/0/6 unit 3 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/6 unit 3 family ethernet-switching vlan members 3
set interfaces irb unit 1 family inet address 10.2.1.3/24 virtual-gateway-address 10.2.1.254
set interfaces irb unit 2 family inet address 10.2.2.3/24 virtual-gateway-address 10.2.2.254
set interfaces irb unit 3 family inet address 10.2.3.3/24 virtual-gateway-address 10.2.3.254
set routing-options router-id 198.51.100.3
set routing-options autonomous-system 65000
set routing-options forwarding-table export evpn-pplb
set protocols mpls interface xe-2/0/2.0
set protocols mpls interface xe-2/0/3.0
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 198.51.100.3
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn local-as 65000
set protocols bgp group evpn neighbor 198.51.100.1
set protocols bgp group evpn neighbor 198.51.100.2

```

```

set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface xe-2/0/2.0
set protocols ospf area 0.0.0.0 interface xe-2/0/3.0
set protocols ldp interface lo0.0
set protocols ldp interface xe-2/0/2.0
set protocols ldp interface xe-2/0/3.0
set policy-options policy-statement evpn-pplb from protocol evpn
set policy-options policy-statement evpn-pplb then load-balance per-packet
set routing-instances evpn1 instance-type virtual-switch
set routing-instances evpn1 interface xe-2/0/6.1
set routing-instances evpn1 route-distinguisher 1:12
set routing-instances evpn1 vrf-target target:1:5
set routing-instances evpn1 protocols evpn extended-vlan-list 1
set routing-instances evpn1 switch-options service-id 1
set routing-instances evpn1 vlans v1 vlan-id 1
set routing-instances evpn1 vlans v1 l3-interface irb.1
set routing-instances evpn2 instance-type virtual-switch
set routing-instances evpn2 interface xe-2/0/6.2
set routing-instances evpn2 route-distinguisher 1:22
set routing-instances evpn2 vrf-target target:1:6
set routing-instances evpn2 protocols evpn extended-vlan-list 2
set routing-instances evpn2 switch-options service-id 2
set routing-instances evpn2 vlans v1 vlan-id 2
set routing-instances evpn2 vlans v1 l3-interface irb.2
set routing-instances evpn3 instance-type virtual-switch
set routing-instances evpn3 interface xe-2/0/6.3
set routing-instances evpn3 route-distinguisher 1:32
set routing-instances evpn3 vrf-target target:1:7
set routing-instances evpn3 protocols evpn extended-vlan-list 3
set routing-instances evpn3 switch-options service-id 3
set routing-instances evpn3 vlans v1 vlan-id 3
set routing-instances evpn3 vlans v1 l3-interface irb.3

```

PE3: Configuring EVPN-MPLS

Step-by-Step Procedure

1. Configure the loopback interface, and the interfaces connected to the other PE devices.

[edit]

```
user@switch# set interfaces lo0 unit 0 family inet address 198.51.100.3/32 primary
```

```

user@switch# set interfaces xe-2/0/2 unit 0 family inet address 192.0.2.1/24
user@switch# set interfaces xe-2/0/2 unit 0 family mpls
user@switch# set interfaces xe-2/0/3 unit 0 family inet address 192.0.2.11/24
user@switch# set interfaces xe-2/0/3 unit 0 family mpls

```

2. Configure interface xe-2/0/6, which is connected to the host.

```

[edit]
user@switch# set interfaces xe-2/0/6 enable
user@switch# set interfaces xe-2/0/6 flexible-vlan-tagging
user@switch# set interfaces xe-2/0/6 encapsulation flexible-ethernet-services
user@switch# set interfaces xe-2/0/6 unit 1 family ethernet-switching interface-mode trunk
user@switch# set interfaces xe-2/0/6 unit 1 family ethernet-switching vlan members 1
user@switch# set interfaces xe-2/0/6 unit 2 family ethernet-switching interface-mode trunk
user@switch# set interfaces xe-2/0/6 unit 2 family ethernet-switching vlan members 2
user@switch# set interfaces xe-2/0/6 unit 3 family ethernet-switching interface-mode trunk
user@switch# set interfaces xe-2/0/6 unit 3 family ethernet-switching vlan members 3

```

3. Configure IRB interfaces irb.1, irb.2, and irb.3.

```

[edit]
user@switch# set interfaces irb unit 1 family inet address 10.2.1.3/24 virtual-gateway-  
address 10.2.1.254
user@switch# set interfaces irb unit 2 family inet address 10.2.2.3/24 virtual-gateway-  
address 10.2.2.254
user@switch# set interfaces irb unit 3 family inet address 10.2.3.3/24 virtual-gateway-  
address 10.2.3.254

```

4. Assign a router ID and the autonomous system in which PE1, PE2, and PE3 reside.

```

[edit]
user@switch# set routing-options router-id 198.51.100.3
user@switch# set routing-options autonomous-system 65000

```

5. Enable per-packet load-balancing for EVPN routes when EVPN multihoming active-active mode is used.

```

[edit]
user@switch# set routing-options forwarding-table export evpn-pplb

```

```

user@switch# set policy-options policy-statement evpn-pplb from protocol evpn
user@switch# set policy-options policy-statement evpn-pplb then load-balance per-packet

```

6. Enable MPLS on interfaces xe-2/0/2.0 and xe-2/0/3.0.

```

[edit]
user@switch# set protocols mpls interface xe-2/0/2.0
user@switch# set protocols mpls interface xe-2/0/3.0

```

7. Configure an IBGP overlay that includes PE1, PE2, and PE3.

```

[edit]
user@switch# set protocols bgp group evpn type internal
user@switch# set protocols bgp group evpn local-address 198.51.100.3
user@switch# set protocols bgp group evpn family evpn signaling
user@switch# set protocols bgp group evpn local-as 65000
user@switch# set protocols bgp group evpn neighbor 198.51.100.1
user@switch# set protocols bgp group evpn neighbor 198.51.100.2

```

8. Configure OSPF as the internal routing protocol for EVPN by specifying an area ID and interfaces on which EVPN-MPLS is enabled.

```

[edit]
user@switch# set protocols ospf area 0.0.0.0 interface lo0.0
user@switch# set protocols ospf area 0.0.0.0 interface xe-2/0/2.0
user@switch# set protocols ospf area 0.0.0.0 interface xe-2/0/3.0

```

9. Configure the LDP on the loopback interface and the interfaces on which EVPN-MPLS is enabled.

```

[edit]
user@switch# set protocols ldp interface lo0.0
user@switch# set protocols ldp interface xe-2/0/2.0
user@switch# set protocols ldp interface xe-2/0/3.0

```

10. Configure virtual switch routing instances for VLAN v1, which is assigned VLAN IDs of 1, 2, and 3, and include the interfaces and other entities associated with the VLAN.

```
[edit]
user@switch# set routing-instances evpn1 instance-type virtual-switch
user@switch# set routing-instances evpn1 interface xe-2/0/6.1
user@switch# set routing-instances evpn1 route-distinguisher 1:12
user@switch# set routing-instances evpn1 vrf-target target:1:5
user@switch# set routing-instances evpn1 protocols evpn extended-vlan-list 1
user@switch# set routing-instances evpn1 switch-options service-id 1
user@switch# set routing-instances evpn1 vlans v1 vlan-id 1
user@switch# set routing-instances evpn1 vlans v1 l3-interface irb.1
user@switch# set routing-instances evpn2 instance-type virtual-switch
user@switch# set routing-instances evpn2 interface xe-2/0/6.2
user@switch# set routing-instances evpn2 route-distinguisher 1:22
user@switch# set routing-instances evpn2 vrf-target target:1:6
user@switch# set routing-instances evpn2 protocols evpn extended-vlan-list 2
user@switch# set routing-instances evpn2 switch-options service-id 2
user@switch# set routing-instances evpn2 vlans v1 vlan-id 2
user@switch# set routing-instances evpn2 vlans v1 l3-interface irb.2
user@switch# set routing-instances evpn3 instance-type virtual-switch
user@switch# set routing-instances evpn3 interface xe-2/0/6.3
user@switch# set routing-instances evpn3 route-distinguisher 1:32
user@switch# set routing-instances evpn3 vrf-target target:1:7
user@switch# set routing-instances evpn3 protocols evpn extended-vlan-list 3
user@switch# set routing-instances evpn3 switch-options service-id 3
user@switch# set routing-instances evpn3 vlans v1 vlan-id 3
user@switch# set routing-instances evpn3 vlans v1 l3-interface irb.3
```


8

PART

PBB-EVPN

- [Configuring PBB-EVPN Integration | 1897](#)
 - [Configuring MAC Pinning for PBB-EVPNs | 2010](#)
-

Configuring PBB-EVPN Integration

IN THIS CHAPTER

- [Provider Backbone Bridging \(PBB\) and EVPN Integration Overview | 1897](#)
- [Example: Configuring PBB with Single-Homed EVPN | 1931](#)
- [Example: Configuring PBB with Multihomed EVPN | 1968](#)

Provider Backbone Bridging (PBB) and EVPN Integration Overview

IN THIS SECTION

- [Technology Overview of PBB-EVPN Integration | 1898](#)
- [Implementation Overview of PBB-EVPN Integration | 1919](#)
- [Configuration Overview of PBB-EVPN Integration | 1926](#)
- [Supported and Unsupported Features on PBB-EVPN | 1930](#)

Ethernet VPN (EVPN) provides a solution for multipoint Layer 2 VPN services with advanced multihoming capabilities using BGP for distributing MAC address reachability information over the core MPLS or IP network. However, with EVPN, several thousands of MAC addresses are carried from each virtual routing and forwarding (VRF) instance, requiring frequent updates on newly learned MAC routes and withdrawn routes. This increases the overhead on the provider network.

Provider backbone bridging (PBB) extends Layer 2 Ethernet switching to provide enhanced scalability, quality-of-service (QoS) features, and carrier-class reliability. With the integration of PBB with EVPN, instead of sending the customer MAC (C-MAC) addresses as control plane learning, the backbone MAC (B-MAC) addresses are distributed in the EVPN core. This simplifies the control plane learning across the core and allows a huge number of Layer 2 services, such as data center connectivity, to transit the network in a simple manner.

The following sections describe the technology and implementation overview of PBB-EVPN integration:

Technology Overview of PBB-EVPN Integration

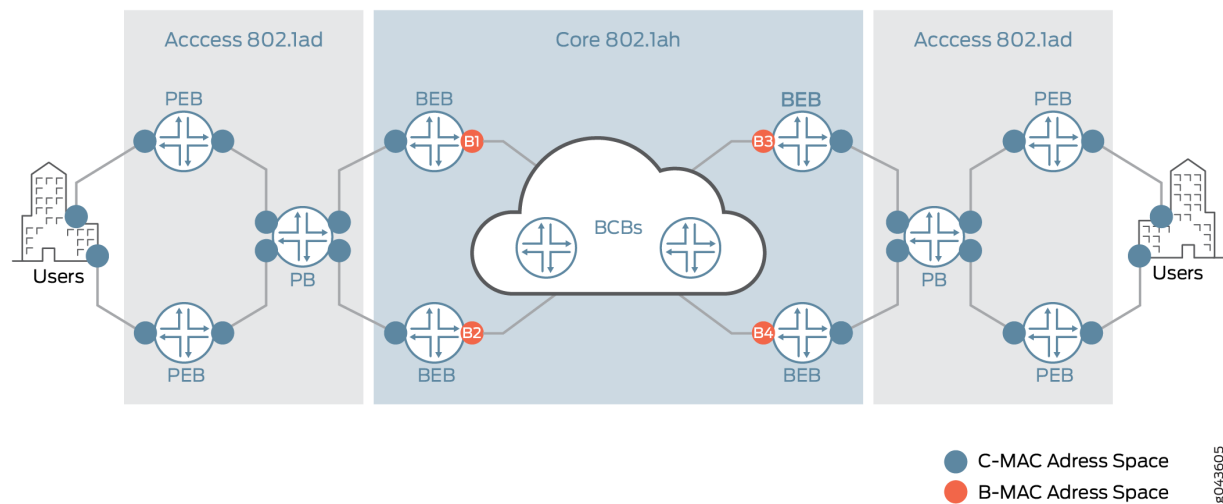
Understanding Provider Backbone Bridging (PBB)

Provider backbone bridging (PBB) was originally defined as IEEE 802.1ah standard, and operates in exactly the same manner as IEEE 802.1ad standard. However, instead of multiplexing VLANs, PBB duplicates the MAC layer of the customer frame and separates it from the provider domain, by encapsulating it in a 24 bit instance service identifier (I-SID). This allows for complete transparency between the customer network and the provider network.

When operating on customer MAC (C-MAC) and service MAC (S-MAC) addresses, PBB uses a new backbone MAC (B-MAC) address. The B-MAC address is added at the edge of the PBB network, that is administered by a carrier VPN or a carrier-of-carriers VPN. With the use of an I-SID for the customer routing instance (I-component) service group, PBB improves the scalability of Ethernet services.

Figure 193 on page 1898 illustrates a PBB network, describing the PBB network elements and MAC address spaces.

Figure 193: PBB Network Elements



The PBB terms are:

- **PB**—Provider bridge (802.1ad)
- **PEB**—Provider edge bridge (802.1ad)
- **BEB**—Backbone edge bridge (802.1ah)
- **BCB**—Backbone core bridge (802.1ah)

The BEB device is the first immediate point of interest within PBB, and forms the boundary between the access network and the core. This introduces two key components—the I-component and B-component in PBB.

- **I-component**

The I-component forms the customer or access facing interface or routing instance. The I-component is responsible for mapping customer Ethernet traffic to the appropriate I-SID. At first, the customer Ethernet traffic is mapped to a customer bridge domain. Then each customer bridge domain is mapped to an I-SID. This service mapping can be per port, per port with service VLAN (S-VLAN), or per port with S-VLAN and customer VLAN (C-VLAN). The I-component is used to learn and forward frames based on the C-MAC addresses, and maintains a C-MAC-to-B-MAC mapping table that is based on instance tag (I-TAG).

Within the I-component there are two ports:

- Customer Instance Port (CIP)

These ports are customer service instances at the customer-facing interfaces. Service definitions can be per port, per port with S-VLAN, or per port with S-VLAN and C-VLAN.

- Provider Instance Port (PIP)

This port performs PBB encapsulation, such as pushing the I-TAG, source and destination B-MAC addresses, and PBB decapsulation, such as popping the I-SID, learning source B-MAC-to-C-MAC mapping, in the ingress direction.

- **B-component**

the B-component is the backbone-facing PBB core instance. The B-component is used to learn and forward packets based on the B-MAC addresses. The B-component is then responsible for mapping the I-SIDs to the appropriate B-VLANs (in the case of PBB networks) or pushing and popping service MPLS labels for MPLS-based networks.

Within the B-component there are two ports:

- Customer Backbone Port (CBP)

These ports are backbone edge ports that can receive and transmit instance-tagged frames from multiple customers, and assign backbone VLAN IDs (B-VIDs) and translate the I-SID on the basis of the received I-SID.

- Provider Backbone Port (PBP)

These ports provide connectivity to the other bridges within and attached to the backbone. These are provider-facing ports. These ports support the S-VLAN component.

Figure 194 on page 1900 illustrates the key components of PBB. Figure 195 on page 1900 illustrates the PBB packet format.

Figure 194: PBB Key Components

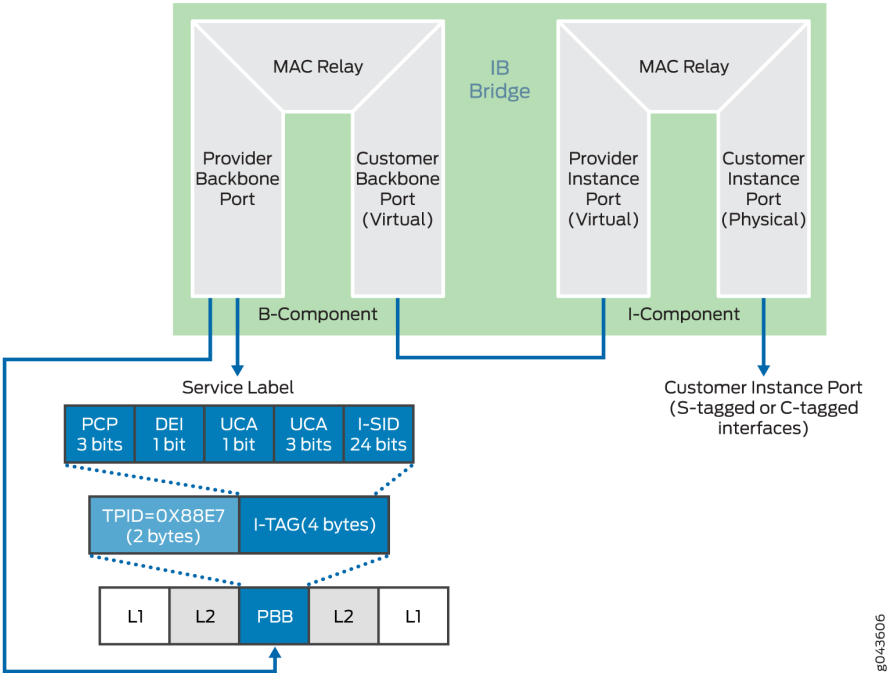
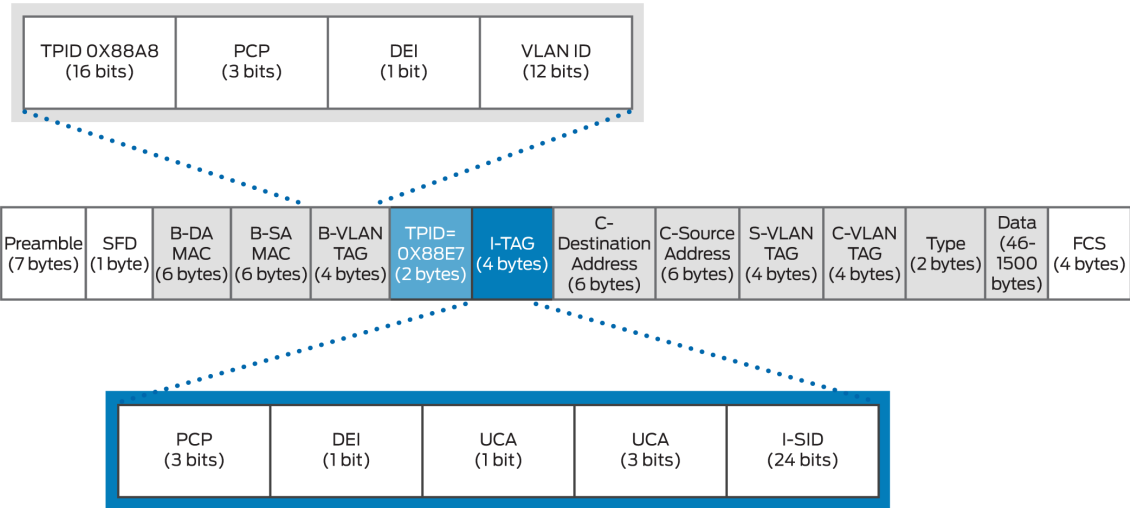


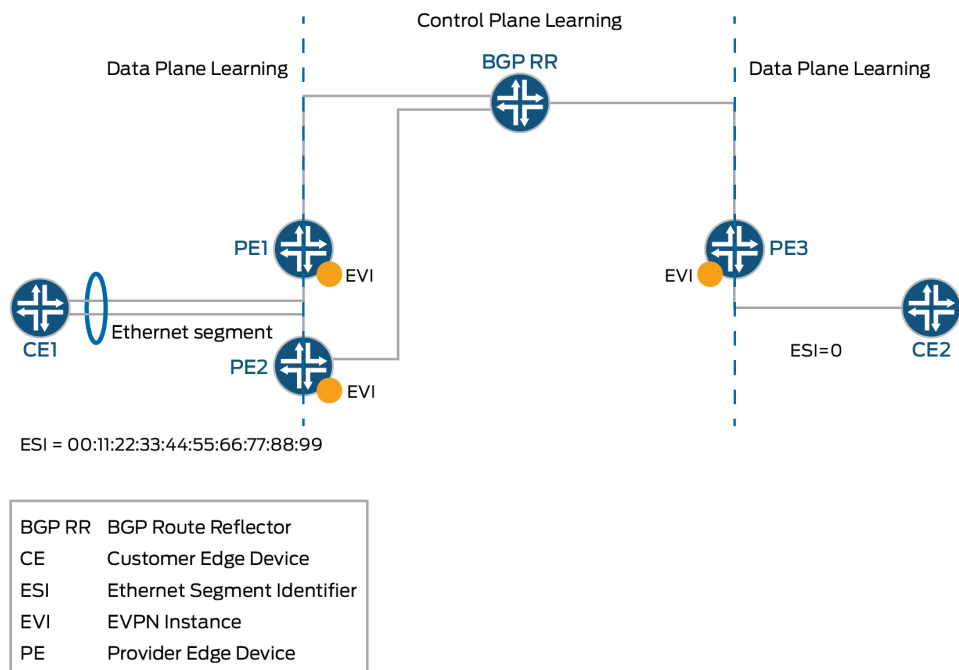
Figure 195: PBB Packet Format



Understanding EVPN

EVPN is a new standards-based technology that provides virtual multipoint bridged connectivity between different Layer 2 domains over an IP or IP/MPLS backbone network. Similar to other VPN technologies, such as IPVPN and VPLS, EVPN instances (EVI) are configured on PE routers to maintain logical service separation between customers. The provider edge (PE) devices connect to customer edge (CE) devices, which can be a router, switch, or host. The PE devices then exchange reachability information using MultiProtocol BGP (MP-BGP) and encapsulated traffic is forwarded between them. Because elements of the architecture are common with other VPN technologies, EVPN can be seamlessly introduced and integrated into existing service environments.

Figure 196: EVPN Overview



The EVPN technology provides mechanisms for next-generation data center interconnection (DCI) by adding extended control plane procedures to exchange the Layer 2 (MAC address) and Layer 3 (IP address) information among the participating Data Center Border Routers (DCBRs). These features help address some of the DCI challenges, such as seamless VM mobility and optimal IP routing. Seamless VM mobility refers to the challenge of Layer 2 extension and maintaining connectivity in the face of VM mobility, and optimal IP routing refers to the challenge of supporting default gateway behavior for a VM's outbound traffic and triangular routing avoidance of a VM's inbound traffic.

The EVPN technology is used by the data center operator to offer multi-tenancy, flexible, and resilient services that can be extended on demand. This flexibility and resiliency can require using compute resources among different physical data centers for a single service (Layer 2 extension) and VM motion.

EVPN supports all-active multihoming, which allows a CE device to connect to two or more PE devices such that traffic is forwarded using all of the links between the devices. This enables the CE device to load-balance traffic to the multiple PE devices. More importantly it allows a remote PE device to load-balance traffic to the multihomed PEs across the core network. This load balancing of traffic flows between data centers is known as aliasing. EVPN also has mechanisms that prevent the looping of broadcast, unknown unicast, and multicast (BUM) traffic in an all-active multihomed topology.



NOTE: We don't support IGMP snooping, MLD snooping, or PIM snooping multicast optimizations with PBB-EVPN.

Multihoming provides redundancy in the event that an access link or a PE device fails. In either case, traffic flows from the CE device toward the PE device use the remaining active links. For traffic in the other direction, the remote PE device updates its forwarding table to send traffic to the remaining active PE devices connected to the multihomed Ethernet segment. EVPN provides a fast convergence mechanism so that the time it takes to make this adjustment is independent of the number of MAC addresses learned by the PE device.

EVPN's MP-BGP control plane allows live virtual machines to be dynamically moved from one data center to another, also known as VM motion. After a VM is moved to a destination server/hypervisor, it transmits a gratuitous ARP that updates the Layer 2 forwarding table of the PE device at the destination data center. The PE device then transmits a MAC route update to all remote PE devices which in turn update their forwarding tables. In this manner, an EVPN tracks the movement of the VM, also known as MAC Mobility. EVPN also has mechanisms to detect and stop MAC flapping.

The EVPN technology, similar to Layer 3 MPLS VPN, introduces the concept of routing MAC addresses using MP-BGP over the MPLS core. Some of the important benefits of using EVPNs include:

- Ability to have a dual-active multihomed edge device
- Provides load balancing across dual-active links
- Provides MAC address mobility
- Provides multi-tenancy
- Provides aliasing
- Enables fast convergence

PBB-EVPN Integration

The integration of PBB with EVPN is described in the following sections:

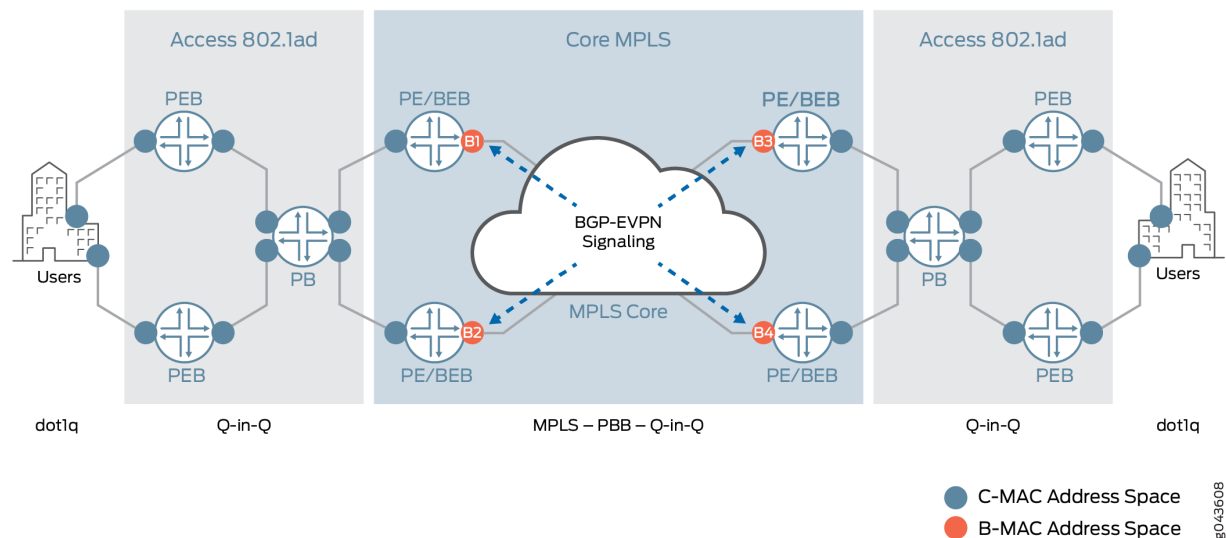
Integrating the PBB and EVPN Network Elements

In a PBB network, a huge amount of customer MAC (C-MAC) addresses are hid behind a drastically smaller number of backbone MAC (B-MAC) addresses, without the devices in the core having to learn and process all the individual customer states. The I-SID creates an encapsulation that enables a large number of services to be deployed. However, unlike modern networks that have a simple MPLS core composed of PE and provider devices, the devices in the PBB core need to act as switches, called the backbone core bridge (BCB), performing forwarding decisions based on B-MAC addresses. This causes incompatibility issues with modern MPLS networks, where packets are switched between edge loopback addresses using MPLS labels and recursion.

With the integration of PBB with EVPN, the BCB element in the PBB core is replaced with MPLS, while retaining the service scaling properties of the BEB PBB edge device. The B-component is signaled using EVPN BGP signaling and encapsulated inside MPLS using PE and provider devices. As a result, the vast scale of PBB is combined with the simplicity of a traditional basic MPLS core network and the amount of network-wide state information is significantly reduced, as opposed to regular PBB.

Figure 197 on page 1903 illustrates the PBB-EVPN integration using the different elements of a PBB and EVPN network.

Figure 197: PBB-EVPN Integration

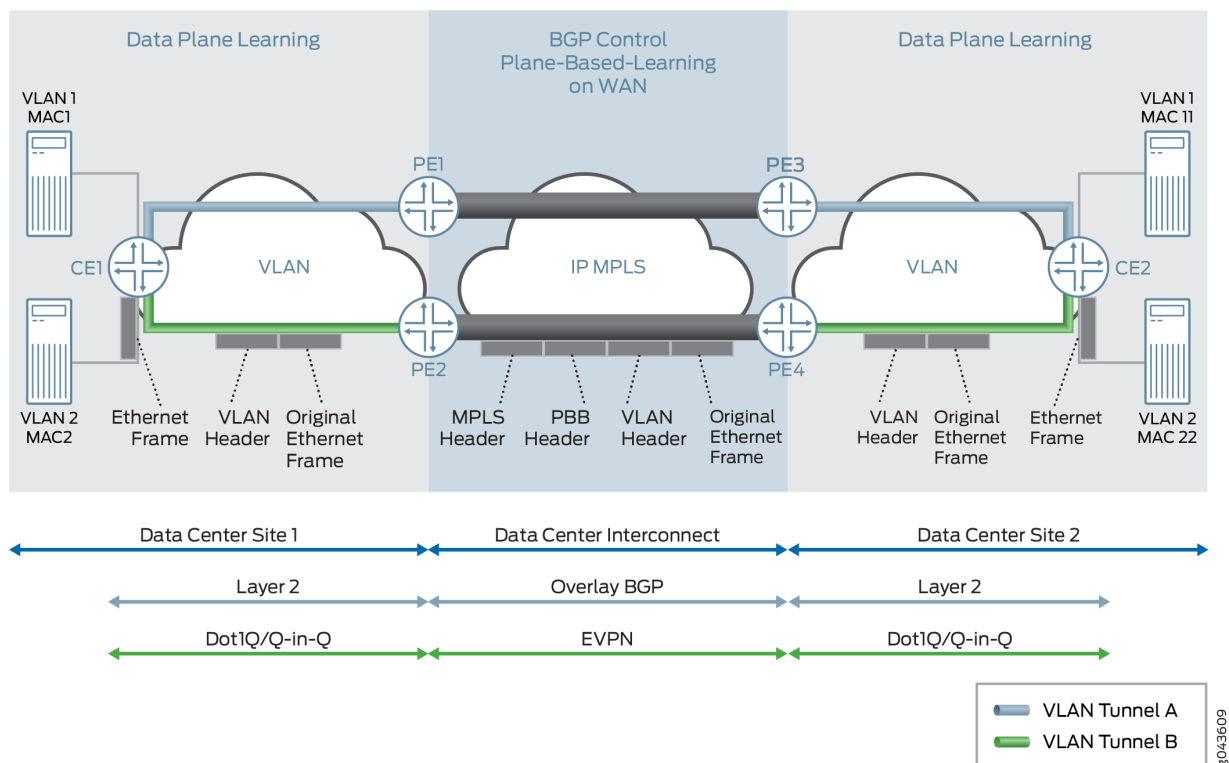


PBB-EVPN Control Plane Initialization

In a PBB-EVPN network, B-MAC addresses are distributed over the EVPN core and C-MAC addresses are learned in the data plane and aggregated behind the B-MAC addresses.

Figure 198 on page 1904 illustrates the control plane handling in a sample PBB-EVPN network with four PE devices and two top-of-rack devices across two data centers.

Figure 198: PBB-EVPN Control Plane Handling



The control plane handling at the Data Center Site 1 is as follows:

1. The C-MAC address lookup occurs and the C-MAC address is learned.
2. The B-MAC source address and I-SID are pushed on the packet.
3. Destination address lookup of C-MAC to B-MAC is done in the I-SID table. If the MAC address is present, the packet is routed using an EVPN MAC route; otherwise a multicast route is used.
4. This route gives the service label for the packet, which has PBB and also the original frame.

The control plane handling at the Data Center Site 2 is as follows:

1. At the disposition PE device, the packet is received with a single service label, indicating that it is a PBB frame.
2. The source address allocation of C-MAC to B-MAC is learned in the I-SID table.
3. The source address of the C-MAC is learned in the customer bridge domain (C-BD) MAC table.

Discovery of EVPN Routes in PBB-EVPN

PBB with Dot1ah functionality is implemented at the PE devices. In the case of PBB-EVPN, the PE devices implement the instance and backbone bridge functionality. Only B-MAC addresses are distributed in the control plane, the C-MAC addresses are learned in the data plane. The following EVPN routes are discovered on the different PE devices:

VPN Autodiscovery

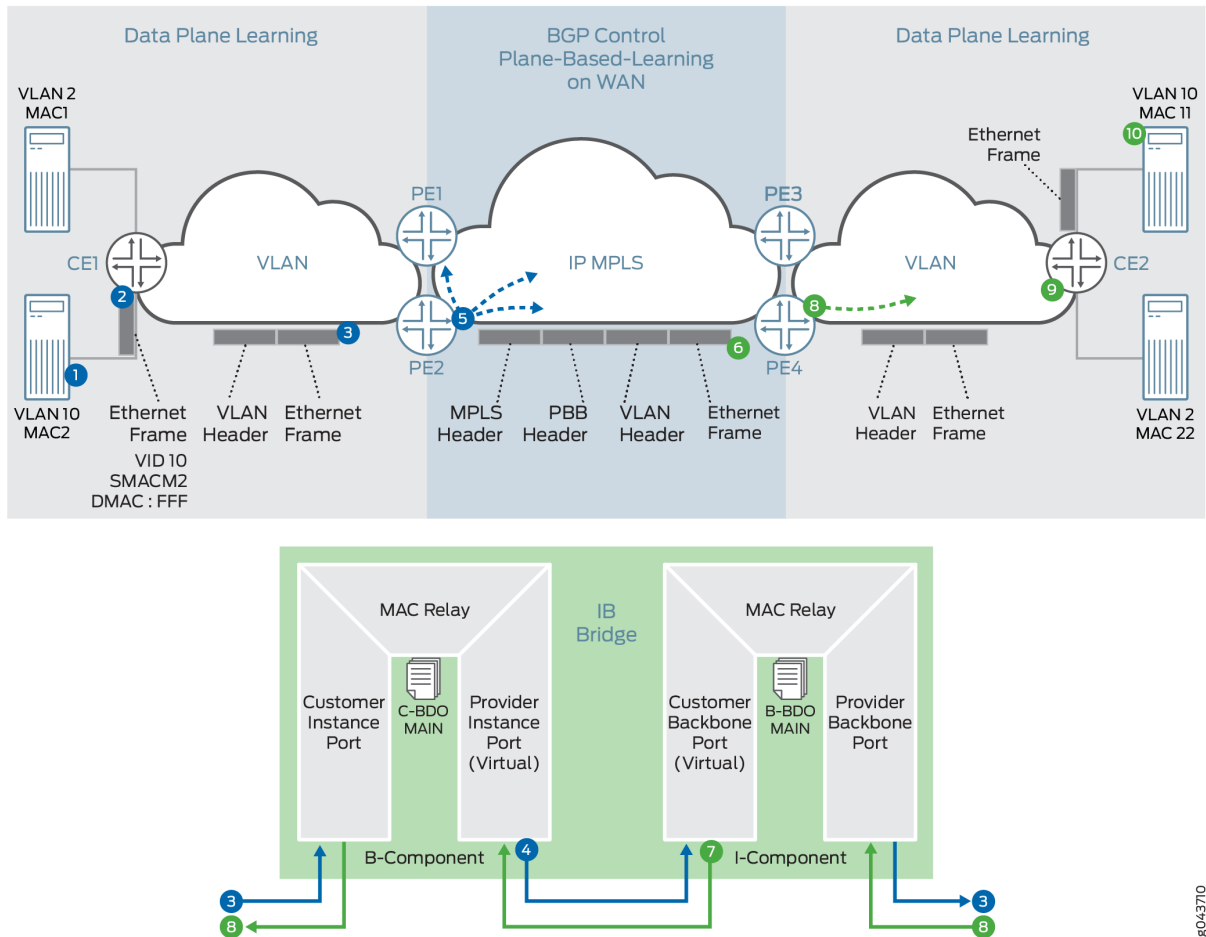
When an EVPN instance (EVI) is configured on different PE devices, autodiscovery of the VPN happens first for discovering the EVPN endpoints. Each PE device that is configured with an EVI sends the inclusive multicast route.

The inclusive multicast route fields are as follows:

- **RD**—Unique route distinguisher value per advertising PE device per EVI. The significance of the RD is local to a PE device.
- **TAG ID**—Service ID equivalent to the I-SID value. One I-SID is assigned to one bridge domain under an EVI when service is supported. The TAG ID is set to 0 for the I-SID bundle service, where multiple I-SIDs are mapped to one bridge domain.
- **Originating IP addr**—Loopback IP address.
- **P-Multicast Service Interface (PMSI) Attributes**—Attributes required for transmitting the BUM traffic. There are two type of tunnels—point-to-multipoint LSP and ingress replication. In the case of ingress replication, the multicast label for BUM traffic is downstream assigned. In the case of point-to-multipoint LSP, the PMSI attribute includes the point-to-multipoint LSP identifier. If the multicast tree is shared or aggregated among multiple EVIs, the PE device uses the upstream assigned label to associate or bind to the EVI.
- **RT Extended Community**—Route target associated with an EVI. This attribute is of global significance in EVPN.

In [Figure 199 on page 1906](#), each PE device sends the inclusive multicast route to each BGP neighbor. Device PE1 sends an inclusive multicast route to Devices PE2, PE3, and PE4 for VPN autodiscovery. The handling of BUM traffic is also illustrated in the figure. During the startup sequence, Devices PE1, PE2, PE3, and PE4 send inclusive multicast route that include the multicast label.

Figure 199: VPN Autodiscovery



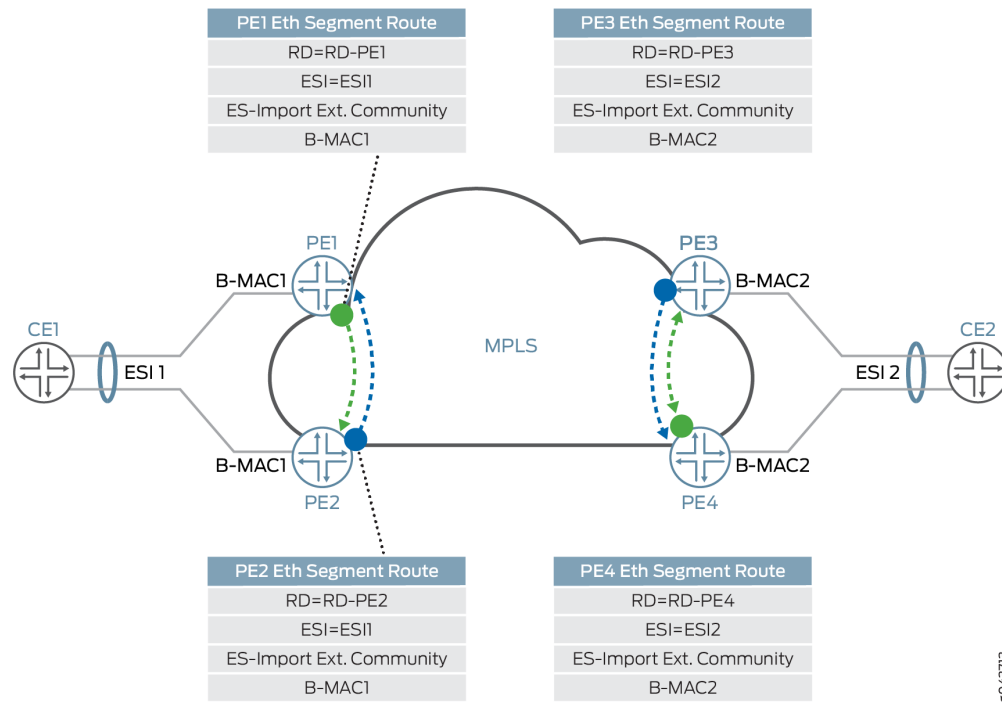
Ethernet Segment Discovery

The Ethernet segment route is encoded in the EVPN NLRI using the Route Type of value 4. This NLRI is used for discovering the multihomed Ethernet segment and for DF election.

ES-import route target, a transitive route target extended community, is also carried with the Ethernet segment route. ES-import extended community enables all the PE devices connected to the same multihomed site to import the Ethernet segment routes. The import of this route is done by the PE device that has the ESI configured. All other PE devices discard this Ethernet segment route.

[Figure 200 on page 1907](#) provide details about the procedure of Ethernet segment route for multihomed Ethernet segment autodiscovery.

Figure 200: Ethernet Segment Discovery



In this figure, Devices PE1 and PE2 are connected to a multihomed segment with ESI value of ESI1 and B-MAC address of B-MAC1. In the case of an active-active multihomed segment, this B-MAC should be the on Devices PE1 and PE2. Similarly, Devices PE3 and PE4 are active/active multihomed for ESI2 with B-MAC address of B-MAC2. Devices PE1 and PE2 send the Ethernet segment route for ESI1, which is received by Devices PE3 and PE4, but is ignored because the devices are not configured for ESI1. Only Devices PE1 and PE2 are in one redundant group and the DF election is performed in this group. Similarly, Devices PE3 and PE4 are in another redundant group and either Device PE3 or PE4 is selected as the DF.

Ethernet MAC Routes Discovery

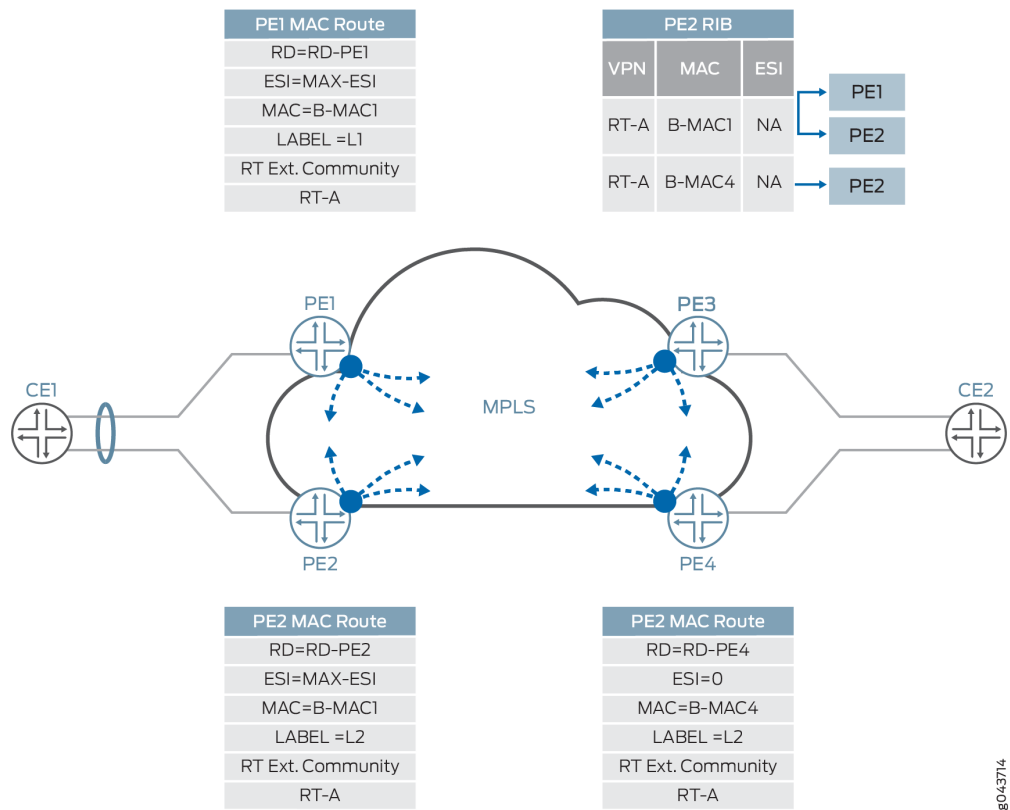
Ethernet MAC Advertisement route is used for distributing B-MAC addresses of PE nodes. The MAC advertisement route is encoded with following fields:

- MAC address field contains B-MAC address.
- Ethernet Tag field is set to 0.
- Ethernet segment identifier field must be set to 0 (for single-homed segments or multihomed segments with per-I-SID load balancing) or to MAX-ESI (for multihomed segment with per-flow load balancing).
- Label is associated with unicast forwarding of traffic from different PE devices.

- RT (route target) extended community associated with its EVI.

Figure 201 on page 1908 illustrates the MAC route advertisement in PBB-EVPN.

Figure 201: Ethernet MAC Routes Discovery



8043714

Differences Between PBB-EVPN and EVPN

Table 96 on page 1909 and Table 97 on page 1909 list the differences between PBB-EVPN and pure EVPN for Layer 2 networks in the context of different route types and route attributes, respectively.

Table 96: Route Differences Between PBB-EVPN and EVPN

Route	Usage	Applicability
Ethernet autodiscovery route	<ul style="list-style-type: none"> • MAC mass withdrawal • Aliasing • Advertising split horizon labels 	EVPN only
MAC advertisement route	<ul style="list-style-type: none"> • Advertise MAC address reachability • Advertise IP and MAC bindings 	EVPN PBB-EVPN
Inclusive multicast route	Multicast tunnel endpoint discovery	EVPN PBB-EVPN
Ethernet segment route	<ul style="list-style-type: none"> • Redundancy • Designated forwarder (DF) election 	EVPN PBB-EVPN

Table 97: Route Attributes and Route Usage Differences Between PBB-EVPN and EVPN

Attribute	Usage	Applicability
ESI MPLS lable extended community	<ul style="list-style-type: none"> • Encode split horizon label for the Ethernet segment. • Indicate redundancy mode (active/standby or active/active). 	Ethernet autodiscovery route
ES-import extended community	Limit the import scope of the Ethernet segment routes.	Ethernet segment route
MAC mobility extended community	<ul style="list-style-type: none"> • EVPN: Indicates that a MAC address has moved from one segment to another across PE devices. • PBB-EVPN: Signal C-MAC address flush notification. 	MAC advertisement route

Table 97: Route Attributes and Route Usage Differences Between PBB-EVPN and EVPN *(Continued)*

Attribute	Usage	Applicability
Default gateway extended community	Indicate the MAC or IP bindings of a gateway.	MAC advertisement route

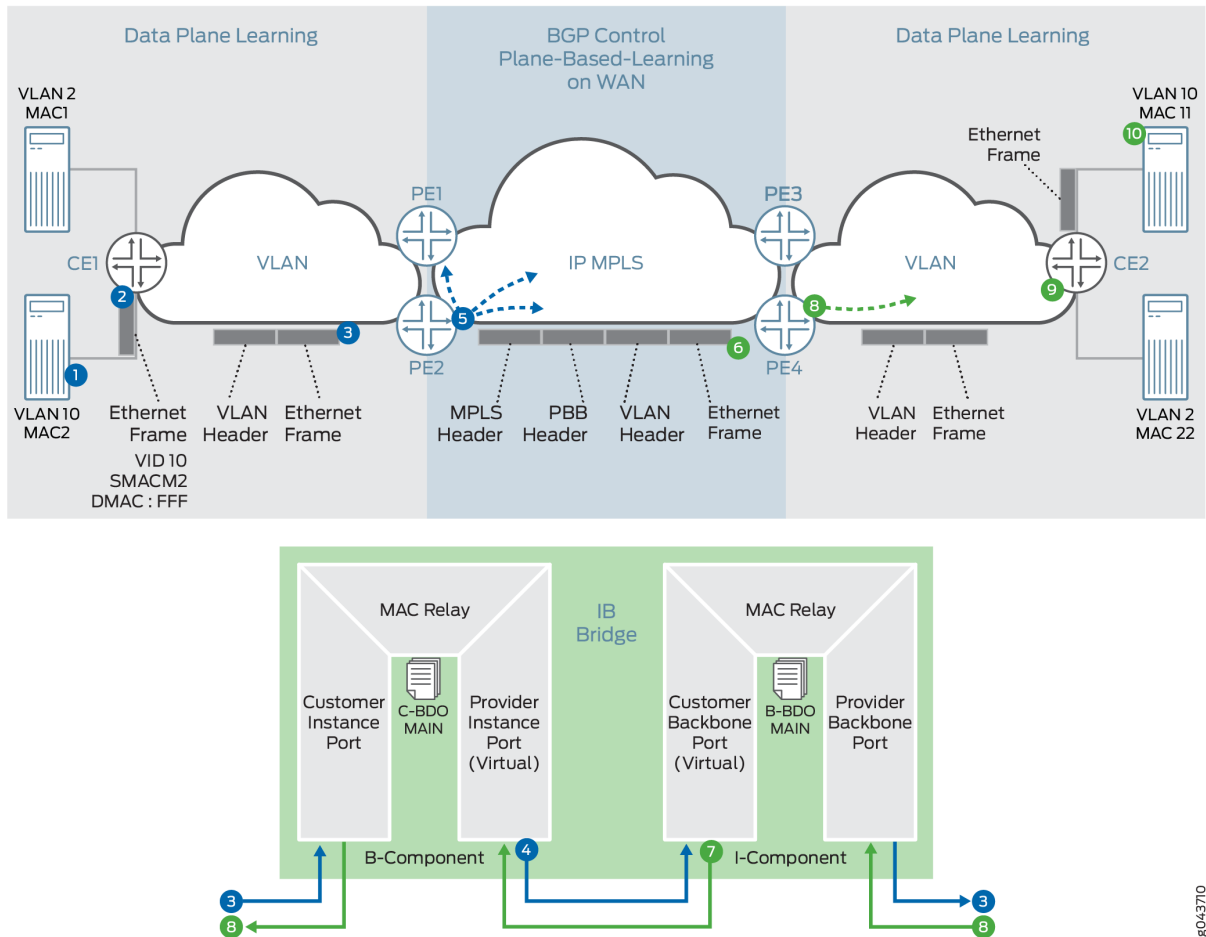
PBB-EVPN Packet Walkthrough

Based on the PBB and EVPN configuration on different PE devices of the network, the Ethernet segment, B-MAC address reachability, and multicast routes are already programmed on different PE devices in the EVPN cloud. The packet walkthrough of PBB-EVPN includes the handling of the following traffic types:

Handling BUM Traffic in PBB-EVPN

[Figure 202 on page 1911](#) illustrates the PBB-EVPN control plane and BUM traffic handling.

Figure 202: PBB-EVPN BUM Traffic Handling



The PBB-EVPN handling of BUM traffic over the EVPN cloud is as follows:

1. After Server A boots up, Server A attempts to send traffic to Server B. Server A does not have the ARP binding in the ARP table for Server B, so Server A builds an ARP broadcast request and sends it. The contents of the ARP packets are VLAN 10, S-MAC=M2 (server A interface MAC), Destination MAC=ff.ff.ff.ff.ff, source IP address=IP address of Server A or VM IP address, and destination IP address=IP address of Server B. Ether type of packet for ARP is 0x0806. Layer 2 frame is sent to Device CE1.
2. Device CE1 does the Layer 2 switching operation on this frame. Because this is a Layer 2 broadcast frame, the frame is classified to an interface and based on the bridge domain configuration for this service and broadcast behavior. The packet is forwarded to all the members of the bridge domain except to the one on which it was received. There might be VLAN translation, such as push, pop, or translate, done on this frame. This frame is sent over to Device PE2. This frame might be untagged, single tagged, or Q-in-Q.

3. After Device PE2 receives this frame, at first it goes through the classification engine to classify the frame into a service. Based on the classification result interface (that is, the Customer Instance Port [CIP]) the service is identified. The source MAC address is learned (if it is not present in the MAC table). This classification results in the C-BD. Because this frame is a broadcast frame, it is sent to all the member interfaces of this bridge domain. One of the member interfaces of this bridge domain is the Provider Instance Port (PIP) interface. Now the packet is formed based on the I-SID configured for this PIP interface. The outer header of the packet at the PIP egress interface is formed based on the following information:
 - I-SID—Configured I-SID value on this PIP interface.
 - Source MAC address—B-MAC address configured or autogenerated for this frame.
 - Destination MAC address—Based on the per-I-SID mapping table that is built based on the source C-MAC-to-B-MAC address learning and the destination C-MAC-to-B-MAC address. For BUM traffic, the default value of the bridge destination address (B-DA) is the Backbone Service Instance Group address. When the B-DA of a frame is a Backbone Service Instance Group address, the normal behavior is to deliver the frame to all Customer Backbone Ports (CBPs) reachable within the backbone VLAN (B-VLAN) to which the backbone service instance is mapped. Filtering based on I-SID by the egress CBP ensures that frames are not transmitted by CBPs that are not part of the backbone service instance.
 - Layer 2 Ethernet type—0x88E7.
 - Payload—Customer frame.
4. The I-SID-formed packet is sent to the CBP for identifying the backbone bridge domain (B-BD) associated with the I-SID.
5. Lookup in the B-BD is done to send the packet to the right destination. Because this frame is a broadcast frame and the destination B-MAC is a multicast address (00-1E-83-<ISID value>), the packet needs to be handled as the ingress replication (that is, VPLS edge flood next hop) for EVPN. This next hop pushes the service label (multicast MPLS label associated with B-VLAN per peer ID and bridge VLAN ID). The MPLS packet is formed and sent over the MPLS cloud for Devices PE1, PE3 and PE4.
6. The frame is received by Device PE4 as a MPLS packet. The bridge domain identification is accomplished by doing an MPLS label L1 lookup in the mpls.0 routing table. The MPLS lookup points to the table next hop for the bridge domain next hop. After the bridge domain is identified, the packet is identified as a broadcast packet. The BUM composite flood next hop is executed, and this next hop points to the CBP.
7. The egress interfaces are identified. One of the egress interfaces is a PIP interface where the I-SID is configured, and I-SID based filtering (MAC filtering) is applied for filtering the frame. The source C-MAC-to-B-MAC address is learned for the I-SID MAC mapping table. This table is used for building the destination B-MAC address for unicast traffic. The outer I-SID header is popped from

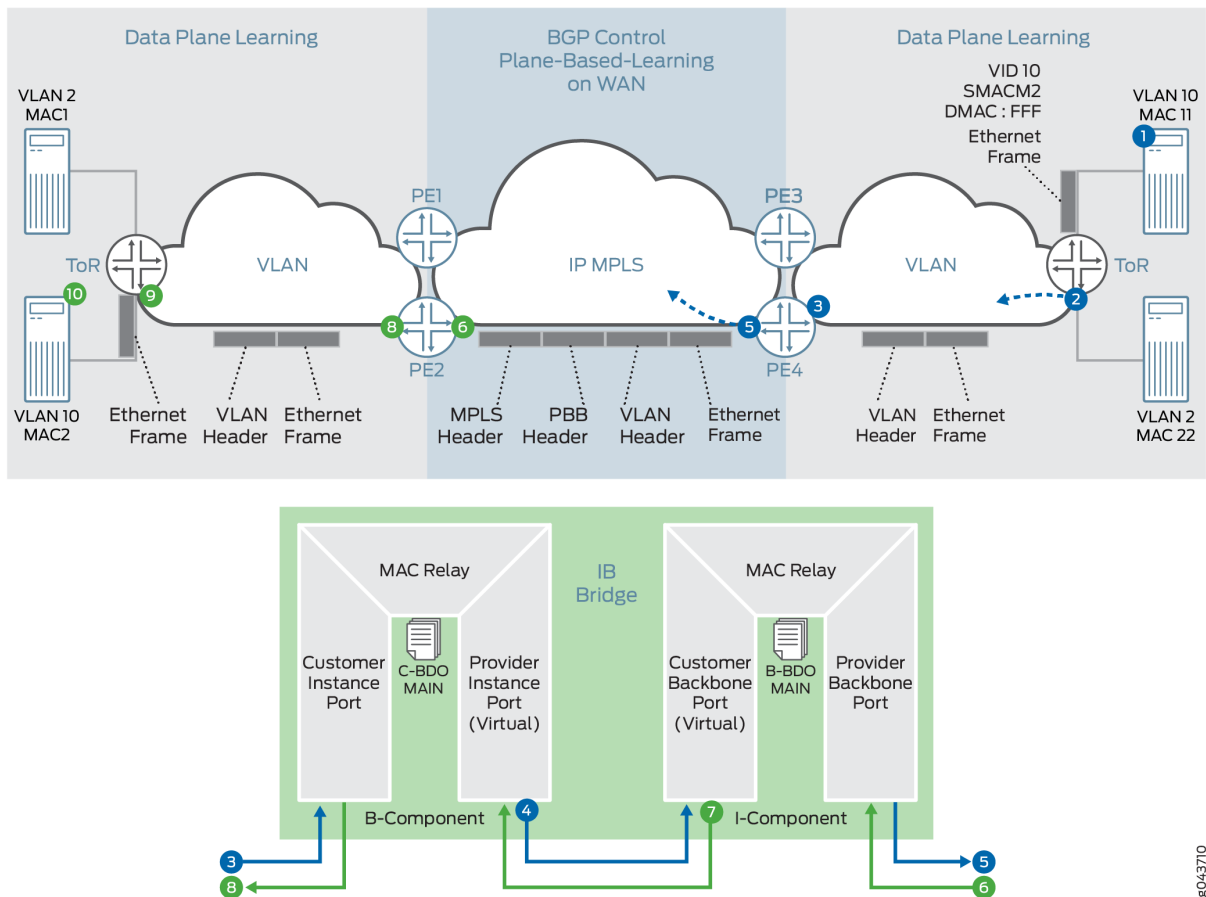
the customer Layer 2 frame. The customer bridge domain (C-BD) is found based on the I-SID classification to the PIP interface.

8. The source C-MAC address is learned. The destination C-MAC lookup is done. This is a broadcast frame, and based on the BUM handling (flood next hop), the frame is forwarded to all the member of the C-BD, except the member interface on which this frame was received.
9. Device CE2 receives this frame. Service classification is done based on the frame VLAN. Based on the classification, the bridge domain forwarding service is found and MAC learning is done. Because the frame is a broadcast frame, it is handled by flood next hop.
10. Server B receives the ARP request packet and sends the ARP reply to Server A.

Handling Unicast Traffic in PBB-EVPN

[Figure 203 on page 1914](#) illustrates the PBB-EVPN control plane and unicast traffic handling in the form of an ARP reply from Server B.

Figure 203: PBB-EVPN Unicast Traffic Handling



For the unicast traffic flow, it is assumed that both the data and control plane MAC learning has already happened.

1. Server B generates an ARP reply. The contents of the ARP packets are VLAN 10, S-MAC=MAC11 (Server B interface MAC), destination MAC=MACA, source IP address=IP address of Server B or VM IP address, and destination IP address=IP address of Server A. This frame is forwarded to top-of-rack B.
2. After receiving the frame, the CE device classifies the incoming frame. Based on the interface family, the bridge domain associated with the interface is identified. Source MAC address learning takes place on the bridge domain. Next the bridge domain destination MAC (MACA) lookup is done, and the lookup provides the Layer 2 egress interface. The output feature of the egress interface is applied before the CE device sends the frame to the egress interface.

3. Layer 2 encapsulated frame is received by Device PE4. The Layer 2 service classification is done to identify the customer bridge domain (C-BD) associated with this frame. The source MAC address (MAC11) learning is done in the context of the C-BD on the CIP interface.
4. Destination MAC lookup in the context of C-BD points to the PIP interface. At this point, the PIP interface egress feature list is executed. Based on the feature list, the outer I-SID header is pushed on the original Ethernet frame.
 - Source MAC—B-MAC of Device PE4
 - Destination MAC—B-MAC of Device PE2 (lookup result in I-SID C-MAC-to-B-MAC table)
 - I-SID—Configured value of I-SID
 - Layer 2 Ether typ—0x88E7
5. The destination MAC address (B-MAC of Device PE2) lookup is done in the B-BD MAC address table. This lookup results in a unicast next hop (that is, an EVPN next hop). This next hop contains a unicast MPLS service label. This label is distributed through the multiprotocol BGP (MP-BGP) control plane. The downstream peer allocates this MPLS service label. Allocation of this label can be per EVI; per EVI and VLAN; per EVI, VLAN, and attachment circuit; or per MAC address. Based on the information in the next hop, the MPLS packet is formed and forwarded on the MPLS network.
6. Device PE2 receives the frame. It is identified as an MPLS packet. The MPLS label lookup is done in the mpls.0 routing table. This lookup results in the table next hop. This lookup results in the B-BD table. The B-MAC rule (that is, source B-MAC is destination B-MAC) and I-SID filtering (CBP configured ISID=packet ISID) rules are applied. Based on the received frame I-SID, CBP is identified and B-VLAN is popped.
7. The frame header is passed to the PIP interface for further processing. The C-MAC address (M11 to B-MAC-PE2) to B-MAC mapping is learned in the I-SID table. The outer I-SID header is popped.
8. The inner source MAC address is learned on the PIP interface in the context of C-BD. The inner destination MAC address lookup is done, resulting in the egress CIP interface.
9. The CE device receives the Layer 2 frame, and Layer 2 forwarding is done.
10. Server A receives the unicast ARP reply packet from Server B.

Handling Path Forwarding in PBB-EVPN

In a PBB-EVPN network, a frame can come from either the customer edge (CE) side (bridge interface) or the MPLS-enabled interface (core-facing interface).

The packet flow for packets received from the CE side is as follows:

1. If the frame is received from a CE interface, the interface belongs to the bridge family, and the MAC address lookup and learning are done in the customer bridge domain (C-BD) context. The result of the lookup is a unicast MAC route or a flood MAC route.
2. The next lookup is done in the I-SID MAC table to determine the destination B-MAC associated with the destination C-MAC.
3. The I-SID header is prepended to the packet.
4. The next lookup is done in the B-BD because the PIP interface belongs to the bridge family.
5. The B-BD lookup points to either the unicast MAC route or the flood MAC route and this route points to either the EVPN indirect multicast next hop or the unicast indirect next hop.

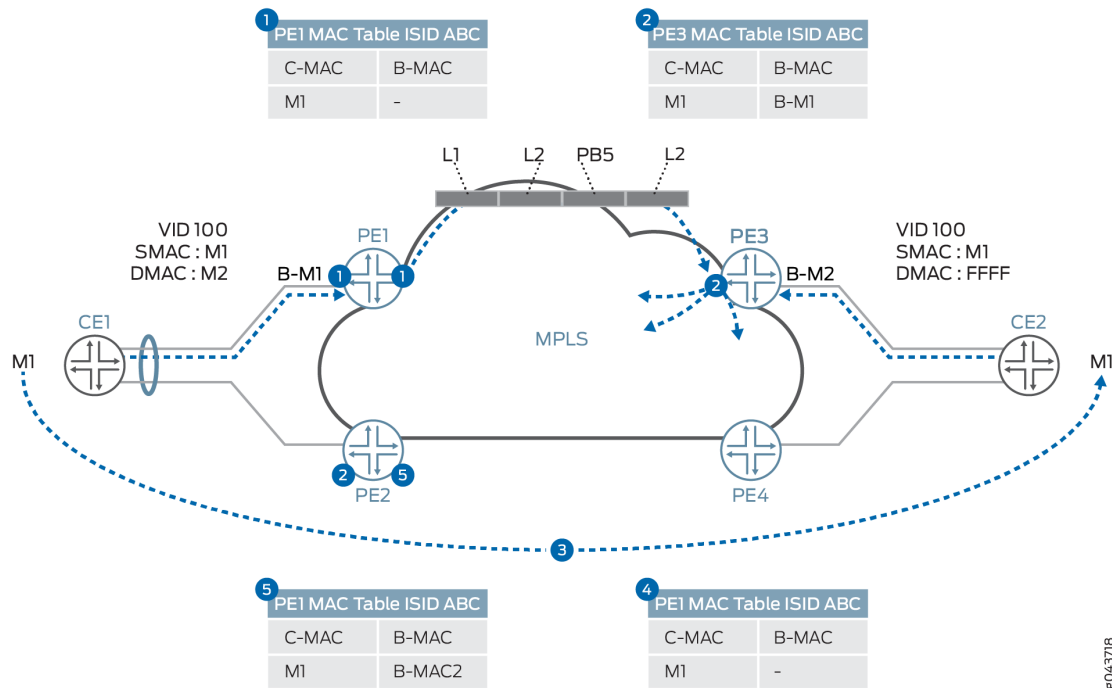
The packet flow for packets received from the core side is as follows:

1. If the frame is received from the core-facing interface, the interface belongs to the MPLS family, and the MPLS label lookup is done in the mpls.0 routing table next hop. The result of this lookup is the routing instance context.
2. The next lookup is done based on the I-SID from the packet to the BBD lookup.
3. If the BBD is found, then I-SID based filtering rules are applied, where the I-SID configured MAC should match the packet source B-MAC, then the frame is dropped.
4. The I-SID MAC table is updated for the destination B-MAC associated with the destination C-MAC for building the C-MAC-to-B-MAC association.
5. The I-SID header is removed and C-BD is found based on the PIP interface.
6. The next lookup is done in the C-BD because the PIP interface belongs to the bridge family.
7. The C-BD lookup points to either a unicast MAC route or a flood MAC route, and this route points to a CE interface or flood route.

Handling MAC Mobility in PBB-EVPN

[Figure 204 on page 1917](#) illustrates PBB-EVPN MAC mobility from the point of the forwarding and control plane.

Figure 204: PBB-EVPN MAC Mobility Handling



MAC mobility from the point of the forwarding and control plane is handled as follows:

1. Device PE1 learns the C-MAC address, M1, on the local port and forwards it across the core according to the destination-C-MAC-to-remote-B-MAC mapping. This mapping is either statically configured or learned through the data plane. If the destination-C-MAC-to-remote-B-MAC mapping is not found in the I-SID mapping table, then the remote B-MAC is derived by using the I-SID.
2. Device PE3 learns the C-MAC address, M1, through the B-MAC address, B-M1, from the dataplane.
3. Customer M1 is moved from Device CE1 to behind Device CE2.
4. When Customer M1 wants to communicate with a customer behind Device CE1, a broadcast traffic with VID: 100, Source MAC: M1, and destination MAC: ff.ff.ff.ff.ff is sent. Device PE3 learns the MAC M1 in the C-BD MAC table and updates the M1 location in the I-SID mapping table.
5. Device PE1 receives the packet and M1 is learned and updated in the I-SID mapping table as reachable through the remote MAC is B-M2.

Handling End-to-End OAM for PBB-EVPN

You can run provider-level Operation, Administration, and Maintenance (OAM) by running connectivity fault management (CFM) on the inward or outward facing maintenance endpoints (MEPs) either on the PIP interface or over the EVPN service.

Currently, the designated forwarder (DF) election is decided based on the DF election algorithm and it is the local decision of the PE devices. This is useful in an end-to-end service handling scenario, where the decision of DF election can be done with operator's consent as well and vice versa. Another scenario in which it might be useful to influence the DF role per service basis or propagating the DF to a CE device is for multihomed networks, where there is no direct link between the CE and PE devices.

Handling QoS and Firewall Policer Support for PBB-EVPN

Table 98 on page 1918 provides details about the QoS and firewall features that are supported in the context of PBB-EVPN integration.

Table 98: Firewall and QoS Feature Support in PBB-EVPN

Feature	Description	Support on round-trip time (RTT)	Support on CE Interface	Support on Core Interface
Classification	Fixed classification to one FC	Yes	Yes	Yes
	Behavior aggregate (BA) and multifield classifier (MF) classification for inner output VLAN 802.1p bit	Yes	Yes	Yes
	BA and MF classification based on drop eligible indicator (DEI) and priority code point (PCP) fields	No	Not required	No
	BA and MF classification based on MPLS experimental (EXP) field	No	No	Yes
CoS Marking	802.1p to I-SID PCP and DEI fields: Customer VLAN 802.1p	No	By default, 802.1p is mapped to PCP and DEI fields	Yes
	802.1p to MPLS EXP field: Customer VLAN 802.1p	No	No	Yes

Table 98: Firewall and QoS Feature Support in PBB-EVPN (Continued)

Feature	Description	Support on round-trip time (RTT)	Support on CE Interface	Support on Core Interface
	MPLS EXP field to I-SID PCP and DEI fields	No	Default behavior	No
	EXP field to 802.1p	No	Yes	No
QoS shaping	Hierarchical scheduling and shaping on ingress device	No	Yes	Yes
	Hierarchical scheduling and shaping on egress device	No	Yes	Yes
Firewall filter	Filtering BUM traffic	Unknown traffic only	Broadcast and multicast traffic only	Broadcast and multicast traffic only
	I-SID-based firewall filter	No	No	No
	Customer VLAN-based filter	No	Yes	Yes
Policer (2 rate 3 color)	Ingress direction	No	Yes	Yes
	Egress direction	No	Yes	Yes

Implementation Overview of PBB-EVPN Integration

The following sections provide use case scenarios for PBB-EVPN integration for DCI.

PBB-EVPN Failure Scenarios

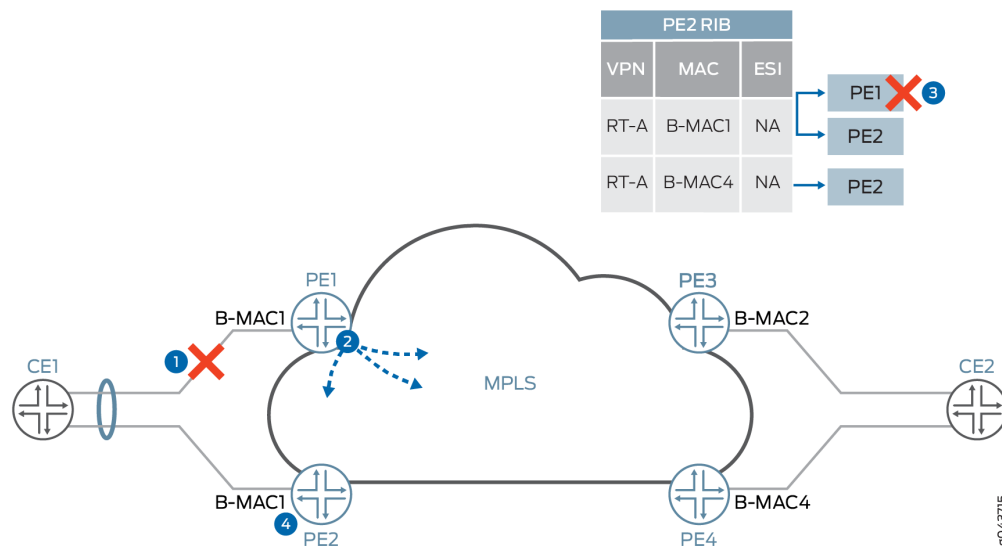
There are different PBB-EVPN failure scenarios that should be taken care of while providing an end-to-end solution. These failure scenarios can be of the following types:

Segment Failure

A segment, or CE-facing link, failure is handled in the active/active and active/standby multihoming redundancy modes.

Figure 205 on page 1920 shows the handling of segment failure for flow-based load balancing at Device CE1.

Figure 205: PBB-EVPN Segment Failure



A segment failure in PBB-EVPN is handled as follows:

1. Ethernet link between Devices CE1 and PE1 failed due to fiber cut or the interface is down. Device PE1 detects the failed segment.
2. Device PE1 withdraws the B-MAC address that is advertised for the failed segment (B-M1).
3. The CE1-facing link goes down. If the link failure happens in the single-active redundancy mode or no redundancy case, the C-MAC flush is also done.

The C-MAC address flushing happens in two ways:

- If Device PE2 uses the shared B-MAC address for multiple I-SIDs, it notifies the remote PE device by readvertising the B-MAC address with the MAC mobility extended community attribute by incrementing the value of counter. This causes the remote PE device to flush all C-MAC addresses associated with the B-MAC address for Device PE1.
- If Device PE2 uses the dedicated B-MAC address, then it withdraws the B-MAC address associated with the failed segment and sends it to Devices PE2, PE3, and PE4.

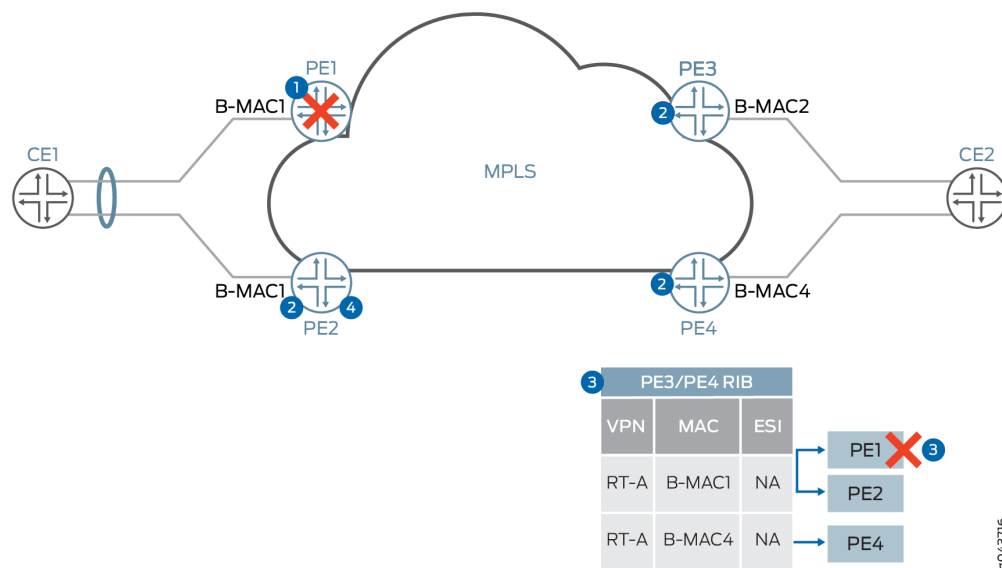
4. After receiving the B-MAC withdraw from Device PE1, Device PE3 removes PE1 reachability for B-MAC1 from its forwarding table. Reachability of B-MAC1 through Device PE2 still exists.
5. The DF election is rerun on Device PE2 for all the I-SIDs for the Ethernet segment ESI.

Node Failure

A node, or PE device, failure scenario is similar to a segment failure from the point of view of CE side failure handling, but it is different from core side failure handling. In the case of core side failure handling, EVPN depends on the BGP session timeout for clearing the state of the EVPN sessions on affected PE devices.

Figure 206 on page 1921 illustrates a node failure scenario for node failure handling.

Figure 206: PBB-EVPN Node Failure



1. Device PE1 failed and the CE side switchover to Device PE2 is done by an interface down event.
2. Devices PE2, PE3, and PE4, or BGP route reflector, detect the BGP session timeout with Device PE1.
3. As soon as the BGP session timeout happens, Devices PE3 and PE4 remove Device PE1 from the forwarding table by marking the Device PE1 next hop as unreachable or deleted. In the case of single-active redundancy mode, the I-SID table for C-MAC-to-B-MAC mapping table has to be flushed or updated. In the case of active/active redundancy mode, it is not required to flush the I-SID table, because the same B-MAC address is used for both Devices PE1 and PE2 for a given EVI.

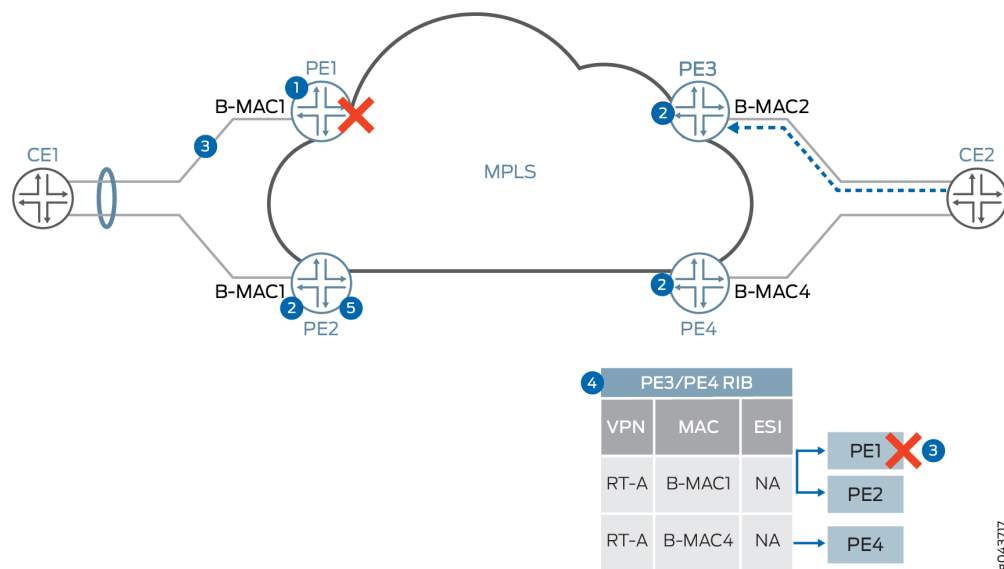
4. At Device PE2, after a BGP timeout, the DF election algorithm is rerun and Device PE2 becomes the DF for all I-SIDs on an affected Ethernet segment.

Core Failure

The handling of core side isolation in the EVPN network is similar to the PE side failure, with some differences in handling of the CE device or Ethernet segment.

Figure 207 on page 1922 provides details about the handling of core isolation.

Figure 207: PBB-EVPN Core Failure



Core isolation in PBB-EVPN is handled as follows:

1. Device PE1 loses connectivity to the core.
2. Devices PE2, PE3, and PE4, or BGP route reflector, detect the BGP session timeout with Device PE1.
3. Device PE1 sends an LACP OUT_OF_SYNC message to Device CE1 to take the port out of the bundle.
4. Device PE2, or BGP route reflector, detects the BGP session timeout with Device PE1.
5. Device PE2 reruns the DF election and is selected as the DF for all the I-SIDs on the segment.

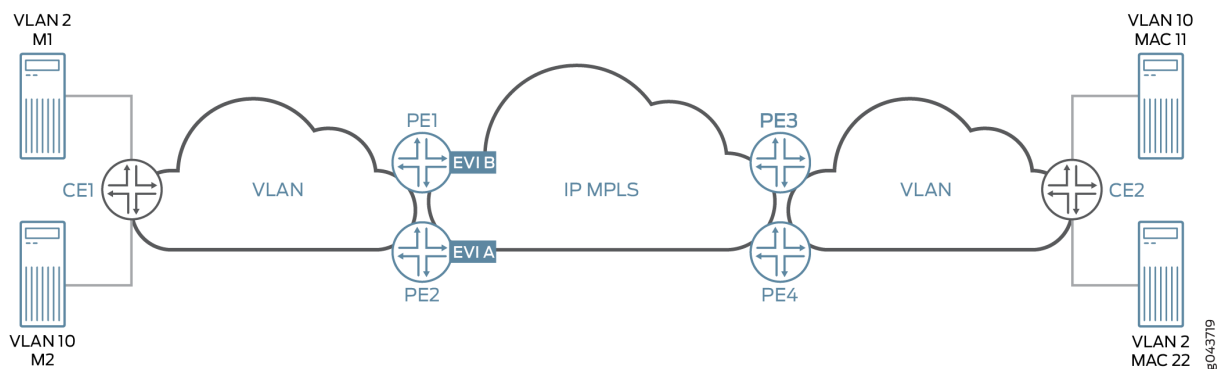
PBB-EVPN I-SID Use Case Scenarios

I-SID Base Service

In the case of the I-SID base service, there is one-to-one mapping between a bridge domain and an EVI. In this case, there is no need to carry the I-SID in the MAC advertisement route, because the bridge domain ID can be derived from the route target (RT) associated with this route. The MPLS label allocation is done on a per-EVI basis.

Figure 208 on page 1923 provides an overview of the I-SID base use case scenario.

Figure 208: I-SID Base Service Use Case



In the case of I-SID load balancing, where load balancing of traffic is done on a per-service basis from the originating CE link aggregation group (LAG) configuration, there are two models for B-MAC addresses:

- Shared source B-MAC

In this model, all I-SIDs from an Ethernet segment share one source B-MAC address. This model has limitations from the point of view of B-MAC withdrawal because of service failure. The remote PE device needs to flush B-MAC-to-C-MAC mapping for all I-SIDs. This creates problem for convergence because MAC flush is done for all I-SIDs.

- Unique source B-MAC per I-SID

Unique unicast B-MAC addresses (one per I-SID) are allocated per multihomed Ethernet segment. The DF filtering is applied to unicast and multicast traffic, in both the core-to-segment and segment-to-core directions.

I-SID-Aware Service

In the case of I-SID-aware service, multiple I-SIDs can be mapped to the same EVI. But there is one-to-one mapping between a bridge domain and an I-SID. The Ethernet Tag ID must be set to the I-SID in the BGP routes advertisements. The MPLS label allocation is done on a per-EVI or per-EVI/I-SID basis so that the PBB can be terminated at the ingress PE device and recreated at the egress PE device.

VPLS Integration with PBB-EVPN Use Case Scenario

In this use case scenario, VPLS is one cloud that is getting integrated with PBB-EVPN by using logical tunnel interfaces. The logical tunnel interface is being terminated into the customer bridge domain (C-BD). MAC address learning from the VPLS cloud is happening in the context of the C-BD. The C-bridge domain gets mapped to the backbone bridge domain and going to the EVPN cloud.

PBB-EVPN Redundancy Use Case Scenarios

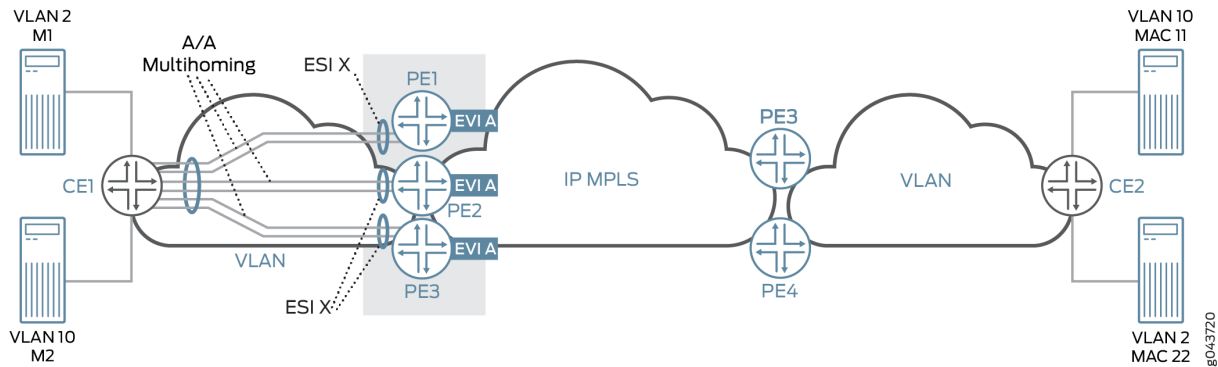
Single Active Redundancy Use Case Scenario

In this use case scenario, a CE device is multihomed to multiple PE devices. The Ethernet segment for this scenario is defined by configuring the same ESI ID on multiple physical interfaces or aggregated Ethernet interfaces on the PE devices along with the mode of operation. In this mode of operation, only one PE device (that is, the DF) is allowed to forward the traffic to and from this Ethernet segment for BUM traffic. The DF election is done based on each ESI in an EVI and by taking the lowest configured I-SID into consideration. This also depends on this number of PE devices to which a segment is multihomed. The service carving is achieved by putting different I-SIDs in different EVIs. The DF election is similar to the DF election in the VLAN-based EVPN. A default timer of 3 seconds is used for reception of Ethernet segment routes from other PE nodes, and this timer can be configured with the same command as used in EVPN—the `designated-forwarder-election hold-time` statement.

In case of PBB-EVPN, the Ethernet autodiscovery route is not used. This mode is specified in the B-MAC advertisement route. In a single-homed or multihomed setup, with the single-active mode, the ESI field must be set to 0 in the B-MAC route advertisement. If ESI 0 is used in the MAC advertisement route, then I-SID-based load balancing is performed. An I-SID value can serve as a single home, an active/standby scenario, or an active/active scenario. It cannot, however, be used in a mixed mode of operation.

[Figure 209 on page 1925](#) provides the use case scenario for active/standby multihoming along with DF election.

Figure 209: PBB-EVPN Redundancy Use Case



Active/Standby Redundancy Use Case Scenario

In the case of the active/active redundancy use case scenario, the DF election is used for handling the BUM traffic. In the case of PBB-EVPN, split horizon of EVPN is not used for filtering the BUM traffic. Instead, BUM traffic is filtered by filtering the destination B-MAC, where the configured B-MAC is the same as the received packet B-MAC; therefore that packet is from the same segment.

The aliasing approach is the same as the EVPN, but the B-MAC route is advertised by setting the ESI field to MAX-ESI. When the remote PE device receives the B-MAC route with the MAX-ESI value, then the remote PE device does the load balancing between Devices PE1 and PE2.

Active/Active Redundancy Use Case Scenario

In a PBB-EVPN active-active multihomed network, all the multihomed PE devices require identical MAC installations. For this purpose, BGP is used to sync the source C-MAC addresses (on the CE side) or the remote C-MAC addresses (from the core) across the multihomed PE devices for same ESI.

To enable MAC sync:

1. For the source C-MAC address sync:

- Configure per-packet-load-balancing on the CE device.
- Ensure that there are minimum flows per source C-MAC, so that each source C-MAC takes both links toward the multihomed PE devices at least once. This ensures that both the multihomed PE devices learn each source C-MAC.

2. For remote C-MAC address sync:

- Ensure that there are minimum flows per remote C-MAC, so that each remote C-MAC takes both links(aliasing) towards the multihomed PE devices at least once while traversing the core. This ensures that both the multihomed PE devices learn each remote C-MAC.

Configuration Overview of PBB-EVPN Integration

The PBB-EVPN configuration is done using the following models:

- One-to-one mapping between the I-SID and the bridge domain

In this configuration model, there is a shared EVPN instance (EVI) between different services, although there is one-to-one mapping between the bridge domain and the I-SID.

- Many-to-one mapping between the I-SID and the bridge domain

In this configuration model, virtual switch configuration is used to allow multiple I-SIDs to be mapped to one bridge domain. This model allows only one bridge domain in a given EVI, and all other bridge domains are mapped to the other Layer 2 services.

Sample PBB-EVPN Port Configuration:

- Provider Backbone Port (PBP) Configuration:

```
routing-instances {
  evpnA {
    instance-type virtual-switch;
    route-distinguisher 192.0.2.1;
    vrf-target target:65221:111;
    protocols {
      evpn {
        label-allocation per-instance;
        extended-isid-list [10000 10401 20000-20001]
      }
    }
  }
}
```

- Customer Backbone Port (CBP) Configuration:

```
interfaces {
  cbp {
    flexible-valn-tagging;
    unit 0 {
      family bridge {
        interface-mode trunk;
        isid-list [10000 10401 20000-20001] [all];
      }
    }
  }
}
```

```

    }
  }
}

```

- Provider Instance Port (PIP) Configuration:

```

interfaces {
  pip {
    flexible-valn-tagging;
    unit 0 {
      family bridge {
        interface-mode trunk;
        isid-list [20000 20001];
      }
    }
    unit 1 {
      family bridge {
        interface-mode trunk;
        isid-list [10000 10401];
      }
    }
  }
}

```

- Customer Instance Port (CIP) Configuration:

```

interfaces {
  ge-4/0/0 {
    flexible-valn-tagging;
    esi 1 primary;
    unit 0 {
      family bridge {
        interface-mode trunk;
        vlan-id-list [1-399 500 800];
      }
    }
  }
  ge-7/0/0/ {
    flexible-vlan-tagging;
    unit 0 {
      family bridge {
        interface-mode trunk;

```



```

        vlan-id-list [1-401];
    }
}
ge-8/0/0/ {
    flexible-vlan-tagging;
    unit 0 {
        family bridge {
            interface-mode trunk;
            vlan-id-list [500-501];
        }
    }
}
}

```

Sample PBB-EVPN Routing Instance Configuration:

- Provider Routing Instance Configuration:

```

routing-instances {
    EVPN A {
        instance-type virtual-switch;
        route-distinguisher 192.0.2.1;
        vrf-target target:65221:111;
        protocols {
            pbb-evpn {
                label-allocation per-instance;
                extended-isid-list [10000 10401 20000] [all];
            }
            evpn {
                label-allocation per-instance;
                extended-vlan-list 1000;
            }
        }
        interface cbp.0;
        bridge-domains {
            B-BD1 {
                isid-list 10000;
            }
            B-BD2 {
                isid-list 10401;
            }
        }
    }
}

```

```

        B-BD3 {
            isid-list 20000;
        }
        B-BD4 {
            isid-list 1000;
        }
    }
    pbb-options {
        vlan-id 1000 isid-list [20001];
        default-bvlan 22;
    }
}
}

```

- Customer Routing Instance Configuration:

```

routing-instances {
    PBN-1 {
        instance-type virtual-switch;
        interface ge-4/0/0.0;
        interface pip.0;
        bridge-domains {
            customer-BDs {
                vlan-id-list [1-399 500 800];
            }
        }
        pbb-options {
            peer-instance EVPN A;
            source-bmac <mac-address>;
            service-groups {
                pbb-1-isid {
                    source-bmac <mac-address>;
                    isid {
                        isid-1 {
                            isid 20000 vlan-id-list [1-399 500] [all];
                            source-bmac <mac-address>;
                            map-dest-bmac-to-dest-cmac <b-mac> <c-mac>;
                        }
                        isid-2 {
                            isid 20001 vlan-id-list [800] [all];
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
  }
}
PBN-2 {
  instance-type virtual-switch;
  interface ge-7/0/0.0;
  interface pip.1;
  bridge-domains {
    customer-BDs {
      vlan-id-list [1-401];
    }
  }
  pbb-options {
    peer-instance EVPN A;
    default-isid <i-sid>;
    service-groups {
      pbb-2-isid-1 {
        isid {
          isid-1 {
            isid 10000 vlan-id-list [1-400];
          }
        }
      }
      pbb-2-isid-2 {
        isid {
          isid-1 {
            isid 10401 vlan-id-list [401];
          }
        }
      }
    }
  }
}
}

```

Supported and Unsupported Features on PBB-EVPN

Junos OS supports the following features with PBB-EVPN:

- Graceful Routing Engine switchover (GRES), unified in-service software upgrade (ISSU), and nonstop software upgrade (NSSU).

- Nonstop active routing (NSR) for BGP peers configured with EVPN family.

NSR on PBB-EVPN replicates and recreates backbone MAC (B-MAC) routes, inclusive multicast routes, and Ethernet segment identifier (ESI) routes.

- Feature support on 64-bit platforms.
- IEEE assigned standard ether type as 0x88E7 for I-SID frames. In addition to this, 802.1x can be used.

The following security considerations are supported:

- Packet destined to the Layer 2 ether type as 0x88E7 is processed only if PBB is enabled on the ingress core PE device.
- Packet received from the core is processed only if the I-SID is known and configured on the ingress PE device; otherwise, the frame is dropped.

Junos OS does not support the following features for PBB-EVPN integration:

- Complete support of EVPN NSR
- Integrated routing and bridging (IRB) interfaces
- IPv6 IP addresses for PBB-EVPN signaling (however, we do support IPv4 or IPv6 client traffic over a PBB-EVPN network)
- Logical systems

RELATED DOCUMENTATION

[Example: Configuring PBB with Multihomed EVPN | 1968](#)

[Example: Configuring PBB with Single-Homed EVPN | 1931](#)

pbb-evpn-core

Example: Configuring PBB with Single-Homed EVPN

IN THIS SECTION

- [Requirements | 1932](#)

- [Overview and Topology | 1932](#)
- [Configuration | 1933](#)
- [Verification | 1960](#)

This example shows how to integrate provider backbone bridging (PBB) with Ethernet VPN (EVPN). With this integration, the control plane operations in the core are simplified, providing faster convergence and scalability enhancements than regular EVPN. The PBB-EVPN applications include Data Center Interconnect (DCI) and carrier Ethernet E-LAN services.

Requirements

This example uses the following hardware and software components:

- Three provider edge (PE) devices each connected to single-homed customer sites.
- Four customer edge (CE) devices that are single-homed to the PE devices.
- Junos OS Release 17.2R1 or later running on all the PE routers.

Before you begin:

- Configure the device interfaces.
- Configure an IGP, such as OSPF, on all the PE devices.
- Establish an internal BGP session between the PE devices.
- Enable RSVP on the PE devices.
- Configure MPLS and label-switched paths (LSPs) between the PE devices.

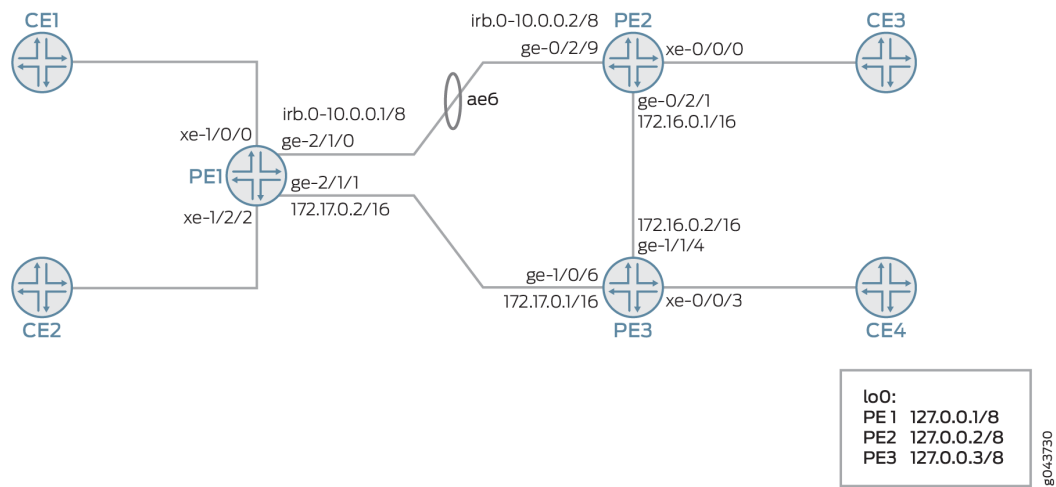
Overview and Topology

With the introduction of the Integrating PBB with EVPN feature, PBB became integrated with Ethernet VPN (EVPN) to enable significant reduction in the control plane learning across the core, allowing a huge number of Layer 2 services, such as data center connectivity, to transit the network in a simplified manner. See the [Integrating PBB with EVPN](#) entry in [Feature Explorer](#) for platform and software release support for the Integrating PBB with EVPN feature.

In a PBB-EVPN network, the backbone core bridge (BCB) device in the PBB core is replaced with MPLS, while retaining the service scaling properties of the PBB backbone edge bridge (BEB). The B-component (provider routing instance) is signalled using EVPN BGP signaling and encapsulated inside MPLS using provider edge (PE) and provider (P) devices. Thus, PBB-EVPN combines the vast scaling property of PBB

with the simplicity of a traditional basic MPLS core network, resulting in significant reduction in the amount of network-wide state information, as opposed to regular PBB.

Figure 210: PBB with Single-Homed EVPN



In [Figure 210 on page 1933](#), PBB is integrated with EVPN, where the CE devices are single-homed to Devices PE1, PE2, and PE3.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1933](#)
- [Procedure | 1947](#)
- [Results | 1953](#)

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.

PE1

```
set chassis aggregated-devices ethernet device-count 16

set chassis network-services enhanced-ip

set interfaces xe-1/0/0 flexible-vlan-tagging

set interfaces xe-1/0/0 encapsulation flexible-ethernet-services

set interfaces xe-1/0/0 unit 0 encapsulation vlan-bridge

set interfaces xe-1/0/0 unit 0 vlan-id 10

set interfaces xe-1/0/0 unit 1 encapsulation vlan-bridge

set interfaces xe-1/0/0 unit 1 vlan-id 20

set interfaces xe-1/2/2 flexible-vlan-tagging

set interfaces xe-1/2/2 encapsulation flexible-ethernet-services

set interfaces xe-1/2/2 unit 0 encapsulation vlan-bridge

set interfaces xe-1/2/2 unit 0 vlan-id 10

set interfaces xe-1/2/2 unit 0 family bridge filter input BRI

set interfaces xe-1/2/2 unit 1 encapsulation vlan-bridge

set interfaces xe-1/2/2 unit 1 vlan-id 20

set interfaces xe-1/2/2 unit 2 encapsulation vlan-bridge

set interfaces xe-1/2/2 unit 2 vlan-id 11

set interfaces xe-1/2/2 unit 2 family bridge

set interfaces xe-1/2/2 unit 3 encapsulation vlan-bridge
```

```
set interfaces xe-1/2/2 unit 3 vlan-id 21

set interfaces xe-1/2/2 unit 3 family bridge

set interfaces ge-2/1/0 gigether-options 802.3ad ae6

set interfaces ge-2/1/1 unit 0 family inet address 10.0.0.1/8

set interfaces ge-2/1/1 unit 0 family iso

set interfaces ge-2/1/1 unit 0 family mpls

set interfaces ae6 encapsulation ethernet-bridge

set interfaces ae6 unit 0 family bridge

set interfaces cbp0 unit 0 family bridge interface-mode trunk

set interfaces cbp0 unit 0 family bridge bridge-domain-type bvlan

set interfaces cbp0 unit 0 family bridge isid-list all

set interfaces cbp0 unit 1 family bridge interface-mode trunk

set interfaces cbp0 unit 1 family bridge bridge-domain-type bvlan

set interfaces cbp0 unit 1 family bridge isid-list all

set interfaces irb unit 0 family inet address 10.0.0.1/8

set interfaces irb unit 0 family iso

set interfaces irb unit 0 family mpls

set interfaces lo0 unit 0 family inet address 127.0.0.1/8 primary

set interfaces pip0 unit 0 family bridge interface-mode trunk

set interfaces pip0 unit 0 family bridge bridge-domain-type svlan

set interfaces pip0 unit 0 family bridge isid-list all-service-groups

set interfaces pip0 unit 1 family bridge interface-mode trunk
```



```
set interfaces pip0 unit 1 family bridge bridge-domain-type svlan

set interfaces pip0 unit 1 family bridge isid-list all-service-groups

set routing-options router-id 127.0.0.1

set routing-options autonomous-system 65221

set protocols rsvp interface all

set protocols rsvp interface fxp0.0 disable

set protocols mpls label-switched-path PE1toPE2 from 127.0.0.1

set protocols mpls label-switched-path PE1toPE2 to 127.0.0.2

set protocols mpls label-switched-path PE1toPE3 from 127.0.0.1

set protocols mpls label-switched-path PE1toPE3 to 127.0.0.3

set protocols mpls interface all

set protocols mpls interface fxp0.0 disable

set protocols bgp group ibgp type internal

set protocols bgp group ibgp local-address 127.0.0.1

set protocols bgp group ibgp family evpn signaling

set protocols bgp group ibgp neighbor 127.0.0.2

set protocols bgp group ibgp neighbor 127.0.0.3

set protocols ospf traffic-engineering

set protocols ospf area 0.0.0.0 interface all

set protocols ospf area 0.0.0.0 interface fxp0.0 disable

set routing-instances pbnn1 instance-type virtual-switch
```

```
set routing-instances pbbn1 interface cbp0.0

set routing-instances pbbn1 route-distinguisher 127.0.0.1:100

set routing-instances pbbn1 vrf-target target:100:100

set routing-instances pbbn1 protocols evpn pbb-evpn-core

set routing-instances pbbn1 protocols evpn extended-isid-list 1000

set routing-instances pbbn1 protocols evpn extended-isid-list 2000

set routing-instances pbbn1 bridge-domains bda vlan-id 100

set routing-instances pbbn1 bridge-domains bda isid-list 1000

set routing-instances pbbn1 bridge-domains bda vlan-id-scope-local

set routing-instances pbbn1 bridge-domains bdb vlan-id 200

set routing-instances pbbn1 bridge-domains bdb isid-list 2000

set routing-instances pbbn1 bridge-domains bdb vlan-id-scope-local

set routing-instances pbn1 instance-type virtual-switch

set routing-instances pbn1 interface pip0.0

set routing-instances pbn1 bridge-domains bda domain-type bridge

set routing-instances pbn1 bridge-domains bda vlan-id 10

set routing-instances pbn1 bridge-domains bda interface xe-1/2/2.0

set routing-instances pbn1 bridge-domains bda interface xe-1/0/0.0

set routing-instances pbn1 bridge-domains bdb domain-type bridge

set routing-instances pbn1 bridge-domains bdb vlan-id 20

set routing-instances pbn1 bridge-domains bdb interface xe-1/2/2.1

set routing-instances pbn1 bridge-domains bdb interface xe-1/0/0.1
```

```

set routing-instances pbn1 bridge-domains bdc domain-type bridge

set routing-instances pbn1 bridge-domains bdc vlan-id 11

set routing-instances pbn1 bridge-domains bdc interface xe-1/2/2.2

set routing-instances pbn1 bridge-domains bdd domain-type bridge

set routing-instances pbn1 bridge-domains bdd vlan-id 21

set routing-instances pbn1 bridge-domains bdd interface xe-1/2/2.3

set routing-instances pbn1 pbb-options peer-instance pbbn1

set routing-instances pbn1 service-groups sga service-type elan

set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-
id-list 10

set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-
id-list 11

set routing-instances pbn1 service-groups sga pbb-service-options source-bmac
00:50:50:50:50:50

set routing-instances pbn1 service-groups sgb service-type elan

set routing-instances pbn1 service-groups sgb pbb-service-options isid 2000 vlan-
id-list 20

set routing-instances pbn1 service-groups sgb pbb-service-options isid 2000 vlan-
id-list 21

set bridge-domains bd vlan-id none

set bridge-domains bd interface ae6.0

set bridge-domains bd routing-interface irb.0

```

PE2

```
set chassis aggregated-devices ethernet device-count 3

set chassis network-services enhanced-ip

set interfaces xe-0/0/0 flexible-vlan-tagging

set interfaces xe-0/0/0 encapsulation flexible-ethernet-services

set interfaces xe-0/0/0 unit 0 encapsulation vlan-bridge

set interfaces xe-0/0/0 unit 0 vlan-id 10

set interfaces xe-0/0/0 unit 1 encapsulation vlan-bridge

set interfaces xe-0/0/0 unit 1 vlan-id 20

set interfaces xe-0/0/0 unit 2 encapsulation vlan-bridge

set interfaces xe-0/0/0 unit 2 vlan-id 11

set interfaces xe-0/0/0 unit 2 family bridge

set interfaces xe-0/0/0 unit 3 encapsulation vlan-bridge

set interfaces xe-0/0/0 unit 3 vlan-id 21

set interfaces xe-0/0/0 unit 3 family bridge

set interfaces ge-0/2/1 unit 0 family inet address 172.16.0.1/16

set interfaces ge-0/2/1 unit 0 family mpls

set interfaces ge-0/2/9 gigether-options 802.3ad ae6

set interfaces ae6 encapsulation ethernet-bridge

set interfaces ae6 unit 0 family bridge

set interfaces cbp0 unit 0 family bridge interface-mode trunk
```

```
set interfaces cbp0 unit 0 family bridge bridge-domain-type bvlan

set interfaces cbp0 unit 0 family bridge isid-list all

set interfaces cbp0 unit 1 family bridge interface-mode trunk

set interfaces cbp0 unit 1 family bridge bridge-domain-type bvlan

set interfaces cbp0 unit 1 family bridge isid-list all

set interfaces irb unit 0 family inet address 10.0.0.2/8

set interfaces irb unit 0 family mpls

set interfaces lo0 unit 0 family inet address 127.0.0.2/8 primary

set interfaces pip0 unit 0 family bridge interface-mode trunk

set interfaces pip0 unit 0 family bridge bridge-domain-type svlan

set interfaces pip0 unit 0 family bridge isid-list all-service-groups

set interfaces pip0 unit 1 family bridge interface-mode trunk

set interfaces pip0 unit 1 family bridge bridge-domain-type svlan

set interfaces pip0 unit 1 family bridge isid-list all-service-groups

set routing-options router-id 127.0.0.2

set routing-options autonomous-system 65221

set protocols rsvp interface all

set protocols rsvp interface fxp0.0 disable

set protocols mpls label-switched-path PE2toPE1 from 127.0.0.2

set protocols mpls label-switched-path PE2toPE1 to 127.0.0.1

set protocols mpls label-switched-path PE2toPE3 from 127.0.0.2
```

```
set protocols mpls label-switched-path PE2toPE3 to 127.0.0.3

set protocols mpls interface all

set protocols mpls interface fxp0.0 disable

set protocols bgp group ibgp type internal

set protocols bgp group ibgp local-address 127.0.0.2

set protocols bgp group ibgp family evpn signaling

set protocols bgp group ibgp neighbor 127.0.0.1

set protocols bgp group ibgp neighbor 127.0.0.3

set protocols ospf traffic-engineering

set protocols ospf area 0.0.0.0 interface all

set protocols ospf area 0.0.0.0 interface fxp0.0 disable

set routing-instances pbbn1 instance-type virtual-switch

set routing-instances pbbn1 interface cbp0.0

set routing-instances pbbn1 route-distinguisher 127.0.0.2:100

set routing-instances pbbn1 vrf-target target:100:100

set routing-instances pbbn1 protocols evpn pbb-evpn-core

set routing-instances pbbn1 protocols evpn extended-isid-list 1000

set routing-instances pbbn1 protocols evpn extended-isid-list 2000

set routing-instances pbbn1 bridge-domains bda vlan-id 100

set routing-instances pbbn1 bridge-domains bda isid-list 1000

set routing-instances pbbn1 bridge-domains bda vlan-id-scope-local

set routing-instances pbbn1 bridge-domains bdb vlan-id 200
```

```

set routing-instances pbbn1 bridge-domains bdb isid-list 2000

set routing-instances pbbn1 bridge-domains bdb vlan-id-scope-local

set routing-instances pbn1 instance-type virtual-switch

set routing-instances pbn1 interface pip0.0

set routing-instances pbn1 bridge-domains bda domain-type bridge

set routing-instances pbn1 bridge-domains bda vlan-id 10

set routing-instances pbn1 bridge-domains bda interface xe-0/0/0.0

set routing-instances pbn1 bridge-domains bdb domain-type bridge

set routing-instances pbn1 bridge-domains bdb vlan-id 20

set routing-instances pbn1 bridge-domains bdb interface xe-0/0/0.1

set routing-instances pbn1 bridge-domains bdc domain-type bridge

set routing-instances pbn1 bridge-domains bdc vlan-id 11

set routing-instances pbn1 bridge-domains bdc interface xe-0/0/0.2

set routing-instances pbn1 bridge-domains bdd domain-type bridge

set routing-instances pbn1 bridge-domains bdd vlan-id 21

set routing-instances pbn1 bridge-domains bdd interface xe-0/0/0.3

set routing-instances pbn1 pbb-options peer-instance pbbn1

set routing-instances pbn1 service-groups sga service-type elan

set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-
id-list 10

set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-
id-list 11

```

```

        set routing-instances pbn1 service-groups sga pbb-service-options source-bmac
00:51:51:51:51:51

        set routing-instances pbn1 service-groups sgb service-type elan

        set routing-instances pbn1 service-groups sgb pbb-service-options isid 2000 vlan-
id-list 20

        set routing-instances pbn1 service-groups sgb pbb-service-options isid 2000 vlan-
id-list 21

        set bridge-domains bd vlan-id none

        set bridge-domains bd interface ae6.0

        set bridge-domains bd routing-interface irb.0

```

PE3

```

        set chassis aggregated-devices ethernet device-count 16

        set chassis network-services enhanced-ip

        set interfaces xe-0/0/3 flexible-vlan-tagging

        set interfaces xe-0/0/3 encapsulation flexible-ethernet-services

        set interfaces xe-0/0/3 unit 0 encapsulation vlan-bridge

        set interfaces xe-0/0/3 unit 0 vlan-id 10

        set interfaces xe-0/0/3 unit 1 encapsulation vlan-bridge

        set interfaces xe-0/0/3 unit 1 vlan-id 20

        set interfaces xe-0/0/3 unit 2 encapsulation vlan-bridge

        set interfaces xe-0/0/3 unit 2 vlan-id 11

```



```
set interfaces xe-0/0/3 unit 2 family bridge

set interfaces xe-0/0/3 unit 3 encapsulation vlan-bridge

set interfaces xe-0/0/3 unit 3 vlan-id 21

set interfaces xe-0/0/3 unit 3 family bridge

set interfaces ge-1/0/6 unit 0 family inet address 172.17.0.1/16

set interfaces ge-1/0/6 unit 0 family mpls

set interfaces ge-1/1/4 unit 0 family inet address 172.16.0.2/16

set interfaces ge-1/1/4 unit 0 family mpls

set interfaces cbp0 unit 0 family bridge interface-mode trunk

set interfaces cbp0 unit 0 family bridge bridge-domain-type bvlan

set interfaces cbp0 unit 0 family bridge isid-list all

set interfaces cbp0 unit 1 family bridge interface-mode trunk

set interfaces cbp0 unit 1 family bridge bridge-domain-type bvlan

set interfaces cbp0 unit 1 family bridge isid-list all

set interfaces lo0 unit 0 family inet address 127.0.0.3/8 primary

set interfaces pip0 unit 0 family bridge interface-mode trunk

set interfaces pip0 unit 0 family bridge bridge-domain-type svlan

set interfaces pip0 unit 0 family bridge isid-list all-service-groups

set interfaces pip0 unit 1 family bridge interface-mode trunk

set interfaces pip0 unit 1 family bridge bridge-domain-type svlan

set interfaces pip0 unit 1 family bridge isid-list all-service-groups

set routing-options router-id 127.0.0.3
```

```
set routing-options autonomous-system 65221

set protocols rsvp interface all

set protocols rsvp interface fxp0.0 disable

set protocols mpls label-switched-path PE3toPE1 from 127.0.0.3

set protocols mpls label-switched-path PE3toPE1 to 127.0.0.1

set protocols mpls label-switched-path PE3toPE2 from 127.0.0.3

set protocols mpls label-switched-path PE3toPE2 to 127.0.0.2

set protocols mpls interface all

set protocols mpls interface fxp0.0 disable

set protocols bgp group ibgp type internal

set protocols bgp group ibgp local-address 127.0.0.3

set protocols bgp group ibgp family evpn signaling

set protocols bgp group ibgp neighbor 127.0.0.1

set protocols bgp group ibgp neighbor 127.0.0.2

set protocols ospf traffic-engineering

set protocols ospf area 0.0.0.0 interface all

set protocols ospf area 0.0.0.0 interface fxp0.0 disable

set routing-instances pbbn1 instance-type virtual-switch

set routing-instances pbbn1 interface cbp0.0

set routing-instances pbbn1 route-distinguisher 127.0.0.3:100

set routing-instances pbbn1 vrf-target target:100:100
```

```
set routing-instances pbbn1 protocols evpn pbb-evpn-core

set routing-instances pbbn1 protocols evpn extended-isid-list 1000

set routing-instances pbbn1 protocols evpn extended-isid-list 2000

set routing-instances pbbn1 bridge-domains bda vlan-id 100

set routing-instances pbbn1 bridge-domains bda isid-list 1000

set routing-instances pbbn1 bridge-domains bda vlan-id-scope-local

set routing-instances pbbn1 bridge-domains bdb vlan-id 200

set routing-instances pbbn1 bridge-domains bdb isid-list 2000

set routing-instances pbbn1 bridge-domains bdb vlan-id-scope-local

set routing-instances pbn1 instance-type virtual-switch

set routing-instances pbn1 interface pip0.0

set routing-instances pbn1 bridge-domains bda domain-type bridge

set routing-instances pbn1 bridge-domains bda vlan-id 10

set routing-instances pbn1 bridge-domains bda interface xe-0/0/3.0

set routing-instances pbn1 bridge-domains bdb domain-type bridge

set routing-instances pbn1 bridge-domains bdb vlan-id 20

set routing-instances pbn1 bridge-domains bdb interface xe-0/0/3.1

set routing-instances pbn1 bridge-domains bdc domain-type bridge

set routing-instances pbn1 bridge-domains bdc vlan-id 11

set routing-instances pbn1 bridge-domains bdc interface xe-0/0/3.2

set routing-instances pbn1 bridge-domains bdd domain-type bridge

set routing-instances pbn1 bridge-domains bdd vlan-id 21
```

```

set routing-instances pbn1 bridge-domains bdd interface xe-0/0/3.3

set routing-instances pbn1 pbb-options peer-instance pbbn1

set routing-instances pbn1 service-groups sga service-type elan

set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-
id-list 10

set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-
id-list 11

set routing-instances pbn1 service-groups sga pbb-service-options source-bmac
00:52:52:52:52:52

set routing-instances pbn1 service-groups sgb service-type elan

set routing-instances pbn1 service-groups sgb pbb-service-options isid 2000 vlan-
id-list 20

set routing-instances pbn1 service-groups sgb pbb-service-options isid 2000 vlan-
id-list 21

```

Procedure

Step-by-Step Procedure

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

To configure Device PE1:

1. Set the number of aggregated Ethernet interfaces on Device PE1.

```

[edit chassis]
user@PE1# set aggregated-devices ethernet device-count 16

```

2. Set Device PE1's network services to enhanced Internet Protocol and use enhanced mode capabilities.

```
[edit chassis]
user@PE1# set chassis network-services enhanced-ip
```

3. Configure the CE-facing interfaces of Device PE1.

```
[edit interfaces]
user@PE1# set xe-1/0/0 flexible-vlan-tagging
user@PE1# set xe-1/0/0 encapsulation flexible-ethernet-services
user@PE1# set xe-1/0/0 unit 0 encapsulation vlan-bridge
user@PE1# set xe-1/0/0 unit 0 vlan-id 10
user@PE1# set xe-1/0/0 unit 1 encapsulation vlan-bridge
user@PE1# set xe-1/0/0 unit 1 vlan-id 20
user@PE1# set xe-1/2/2 flexible-vlan-tagging
user@PE1# set xe-1/2/2 encapsulation flexible-ethernet-services
user@PE1# set xe-1/2/2 unit 0 encapsulation vlan-bridge
user@PE1# set xe-1/2/2 unit 0 vlan-id 10
user@PE1# set xe-1/2/2 unit 0 family bridge filter input BRI
user@PE1# set xe-1/2/2 unit 1 encapsulation vlan-bridge
user@PE1# set xe-1/2/2 unit 1 vlan-id 20
user@PE1# set xe-1/2/2 unit 2 encapsulation vlan-bridge
user@PE1# set xe-1/2/2 unit 2 vlan-id 11
user@PE1# set xe-1/2/2 unit 2 family bridge
user@PE1# set xe-1/2/2 unit 3 encapsulation vlan-bridge
user@PE1# set xe-1/2/2 unit 3 vlan-id 21
user@PE1# set xe-1/2/2 unit 3 family bridge
```

4. Configure the interfaces connecting Device PE1 with the other PE devices.

```
[edit interfaces]
user@PE1# set ge-2/1/0 together-options 802.3ad ae6
```

```

user@PE1# set ge-2/1/1 unit 0 family inet address 10.0.0.1/8
user@PE1# set ge-2/1/1 unit 0 family iso
user@PE1# set ge-2/1/1 unit 0 family mpls

```

5. Configure the aggregated Ethernet bundle ae6.

```

[edit interfaces]
user@PE1# set ae6 encapsulation ethernet-bridge
user@PE1# set ae6 unit 0 family bridge

```

6. Configure the loopback interface of Device PE1.

```

[edit interfaces]
user@PE1# set interfaces lo0 unit 0 family inet address 127.0.0.1/8 primary

```

7. Configure the integrated routing and bridging (IRB) interfaces for Device PE1.

```

[edit interfaces]
user@PE1# set interfaces irb unit 0 family inet address 10.0.0.1/8
user@PE1# set interfaces irb unit 0 family iso
user@PE1# set interfaces irb unit 0 family mpls

```

8. Configure the customer backbone port (CBP) interfaces on Device PE1.

```

[edit interfaces]
user@PE1# set cbp0 unit 0 family bridge interface-mode trunk
user@PE1# set cbp0 unit 0 family bridge bridge-domain-type bvlan
user@PE1# set cbp0 unit 0 family bridge isid-list all
user@PE1# set cbp0 unit 1 family bridge interface-mode trunk
user@PE1# set cbp0 unit 1 family bridge bridge-domain-type bvlan
user@PE1# set cbp0 unit 1 family bridge isid-list all

```

9. Configure the Provider Instance Port (PIP) on Device PE1.

```
[edit interfaces]
user@PE1# set pip0 unit 0 family bridge interface-mode trunk
user@PE1# set pip0 unit 0 family bridge bridge-domain-type svlan
user@PE1# set pip0 unit 0 family bridge isid-list all-service-groups
user@PE1# set pip0 unit 1 family bridge interface-mode trunk
user@PE1# set pip0 unit 1 family bridge bridge-domain-type svlan
user@PE1# set pip0 unit 1 family bridge isid-list all-service-groups
```

10. Configure the router ID and autonomous system number for Device PE1.

```
[edit routing-options]
user@PE1# set router-id 127.0.0.1
user@PE1# set autonomous-system 65221
```

11. Configure RSVP on all the interfaces of Device PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set rsvp interface all
user@PE1# set rsvp interface fxp0.0 disable
```

12. Configure MPLS on all the interfaces of Device PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set mpls interface all
user@PE1# set mpls interface fxp0.0 disable
```

13. Configure LSPs from Device PE1 to all other PE devices.

```
[edit protocols]
user@PE1# set mpls label-switched-path PE1toPE2 from 127.0.0.1
user@PE1# set mpls label-switched-path PE1toPE2 to 127.0.0.2
user@PE1# set mpls label-switched-path PE1toPE3 from 127.0.0.1
user@PE1# set mpls label-switched-path PE1toPE3 to 127.0.0.3
```

14. Configure an internal BGP session under family EVPN from Device PE1 to all other PE devices.

```
[edit protocols]
user@PE1# set bgp group ibgp type internal
user@PE1# set bgp group ibgp local-address 127.0.0.1
user@PE1# set bgp group ibgp family evpn signaling
user@PE1# set bgp group ibgp neighbor 127.0.0.2
user@PE1# set bgp group ibgp neighbor 127.0.0.3
```

15. Configure OSPF on all the interfaces of Device of PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set ospf traffic-engineering
user@PE1# set ospf area 0.0.0.0 interface all
user@PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
```

16. Configure a customer routing instance (I-component) on Device PE1 with type virtual switch. Assign the CBP interface, route-distinguisher, and virtual routing and forwarding (VRF) target values to the PBBN routing instance.

```
[edit routing-instances]
user@PE1# set pbbn1 instance-type virtual-switch
user@PE1# set pbbn1 interface cbp0.0
user@PE1# set pbbn1 route-distinguisher 127.0.0.1:100
user@PE1# set pbbn1 vrf-target target:100:100
```


17. Configure PBB-EVPN integration from the customer routing instance. Assign the extended I-SID list and bridge domains to the routing instance.

```
[edit routing-instances]
user@PE1# set pbbn1 protocols evpn pbb-evpn-core
user@PE1# set pbbn1 protocols evpn extended-isid-list 1000
user@PE1# set pbbn1 protocols evpn extended-isid-list 2000
user@PE1# set pbbn1 bridge-domains bda vlan-id 100
user@PE1# set pbbn1 bridge-domains bda isid-list 1000
user@PE1# set pbbn1 bridge-domains bda vlan-id-scope-local
user@PE1# set pbbn1 bridge-domains bdb vlan-id 200
user@PE1# set pbbn1 bridge-domains bdb isid-list 2000
user@PE1# set pbbn1 bridge-domains bdb vlan-id-scope-local
```

18. Configure a provider routing instance on Device PE1 with type virtual switch. Assign the PBP interface and bridge domains to the routing instance.

```
[edit routing-instances]
user@PE1# set pbn1 instance-type virtual-switch
user@PE1# set pbn1 interface pip0.0
user@PE1# set pbn1 bridge-domains bda domain-type bridge
user@PE1# set pbn1 bridge-domains bda vlan-id 10
user@PE1# set pbn1 bridge-domains bda interface xe-1/2/2.0
user@PE1# set pbn1 bridge-domains bda interface xe-1/0/0.0
user@PE1# set pbn1 bridge-domains bdb domain-type bridge
user@PE1# set pbn1 bridge-domains bdb vlan-id 20
user@PE1# set pbn1 bridge-domains bdb interface xe-1/2/2.1
user@PE1# set pbn1 bridge-domains bdb interface xe-1/0/0.1
user@PE1# set pbn1 bridge-domains bdc domain-type bridge
user@PE1# set pbn1 bridge-domains bdc vlan-id 11
user@PE1# set pbn1 bridge-domains bdc interface xe-1/2/2.2
user@PE1# set pbn1 bridge-domains bdd domain-type bridge
user@PE1# set pbn1 bridge-domains bdd vlan-id 21
user@PE1# set pbn1 bridge-domains bdd interface xe-1/2/2.3
```

19. Configure the peer PBBN routing instance in the customer routing instance.

```
[edit routing-instances]
user@PE1# set pbn1 pbb-options peer-instance pbbn1
```

20. Configure the service groups to be supported in the customer routing instance.

```
[edit routing-instances]
user@PE1# set pbn1 service-groups sga service-type elan
user@PE1# set pbn1 service-groups sga pbb-service-options isid 1000 vlan-id-list 10
user@PE1# set pbn1 service-groups sga pbb-service-options isid 1000 vlan-id-list 11
user@PE1# set pbn1 service-groups sga pbb-service-options source-bmac 00:50:50:50:50:50
user@PE1# set pbn1 service-groups sgb service-type elan
user@PE1# set pbn1 service-groups sgb pbb-service-options isid 2000 vlan-id-list 20
user@PE1# set pbn1 service-groups sgb pbb-service-options isid 2000 vlan-id-list 21
```

21. Configure the bridge domains on Device PE1.

```
[edit bridge-domains]
user@PE1# set bd vlan-id none
user@PE1# set bd interface ae6.0
user@PE1# set bd routing-interface irb.0
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `show routing-options`, `show protocols`, `show routing-instances`, 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@PE1# show chassis
aggregated-devices {
    ethernet {
        device-count 16;
    }
}
network-services enhanced-ip;
```

```
user@PE1# show interfaces
xe-1/0/0 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 10;
    }
    unit 1 {
        encapsulation vlan-bridge;
        vlan-id 20;
    }
}
xe-1/2/2 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 10;
        family bridge {
            filter {
                input BRI; ## reference 'BRI' not found
            }
        }
    }
    unit 1 {
        encapsulation vlan-bridge;
        vlan-id 20;
    }
    unit 2 {
```

```

        encapsulation vlan-bridge;
        vlan-id 11;
        family bridge;
    }
    unit 3 {
        encapsulation vlan-bridge;
        vlan-id 21;
        family bridge;
    }
}
ge-2/1/0 {
    together-options {
        802.3ad ae6;
    }
}
ge-2/1/1 {
    unit 0 {
        family inet {
            address 10.0.0.1/8;
        }
        family iso;
        family mpls;
    }
}
ae6 {
    encapsulation ethernet-bridge;
    unit 0 {
        family bridge;
    }
}
cbp0 {
    unit 0 {
        family bridge {
            interface-mode trunk;
            bridge-domain-type bvlan;
            isid-list all;
        }
    }
    unit 1 {
        family bridge {
            interface-mode trunk;
            bridge-domain-type bvlan;
            isid-list all;
        }
    }
}

```

```

    }
  }
}
irb {
  unit 0 {
    family inet {
      address 10.0.0.1/8;
    }
    family iso;
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 127.0.0.1/8 {
        primary;
      }
    }
  }
}
pip0 {
  unit 0 {
    family bridge {
      interface-mode trunk;
      bridge-domain-type svlan;
      isid-list all-service-groups;
    }
  }
  unit 1 {
    family bridge {
      interface-mode trunk;
      bridge-domain-type svlan;
      isid-list all-service-groups;
    }
  }
}

```

```

    }
}

```

```

user@PE1# show routing-options
router-id 127.0.0.1;
autonomous-system 65221;

```

```

user@PE1# show protocols
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
mpls {
    label-switched-path PE1toPE2 {
        from 127.0.0.1;
        to 127.0.0.2;
    }
    label-switched-path PE1toPE3 {
        from 127.0.0.1;
        to 127.0.0.3;
    }
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    group ibgp {
        type internal;
        local-address 127.0.0.1;
        family evpn {
            signaling;
        }
        neighbor 127.0.0.2;
        neighbor 127.0.0.3;
    }
}
ospf {

```

```

traffic-engineering;
area 0.0.0.0 {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
}

```

```

user@PE1# show routing-instances
pbbn1 {
    instance-type virtual-switch;
    interface cbp0.0;
    route-distinguisher 127.0.0.1:100;
    vrf-target target:100:100;
    protocols {
        evpn {
            pbb-evpn-core;
            extended-isid-list [ 1000 2000 ];
        }
    }
    bridge-domains {
        bda {
            vlan-id 100;
            isid-list 1000;
            vlan-id-scope-local;
        }
        bdb {
            vlan-id 200;
            isid-list 2000;
            vlan-id-scope-local;
        }
    }
}
pbn1 {
    instance-type virtual-switch;
    interface pip0.0;
    bridge-domains {
        bda {
            domain-type bridge;
            vlan-id 10;

```

```

        interface xe-1/2/2.0;
        interface xe-1/0/0.0;
    }
    bdb {
        domain-type bridge;
        vlan-id 20;
        interface xe-1/2/2.1;
        interface xe-1/0/0.1;
    }
    bdc {
        domain-type bridge;
        vlan-id 11;
        interface xe-1/2/2.2;
    }
    bdd {
        domain-type bridge;
        vlan-id 21;
        interface xe-1/2/2.3;
    }
}
pbb-options {
    peer-instance pbbn1;
}
service-groups {
    sga {
        service-type elan;
        pbb-service-options {
            isid 1000 vlan-id-list [ 10 11 ];
            source-bmac 00:50:50:50:50:50;
        }
    }
    sgb {
        service-type elan;
        pbb-service-options {
            isid 2000 vlan-id-list [ 20 21 ];
        }
    }
}

```



```
}  
}
```

```
user@PE1# show bridge-domains  
bd {  
    vlan-id none;  
    interface ae6.0;  
    routing-interface irb.0;  
}
```

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

Verification

IN THIS SECTION

- [Verifying BGP Peering Status | 1960](#)
- [Verifying MPLS LSPs | 1961](#)
- [Verifying the EVPN Routing Instance | 1962](#)
- [Verifying Routing Table Entries of the EVPN Routing Instance | 1963](#)
- [Verifying the EVPN Database | 1965](#)
- [Verifying the MAC Table Entries | 1966](#)
- [Verifying the inet.3 Routing Table Entries | 1967](#)

Confirm that the configuration is working properly.

Verifying BGP Peering Status

Purpose

Verify that the BGP session is established between the PE devices.

Action

From operational mode, run the `show bgp summary` command.

```
user@PE1> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State   Pending
bgp.evpn.0
                8          8          0          0          0          0
Peer          AS      InPkt    OutPkt    OutQ    Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
127.0.0.2      65221      9        7        0        0      2:09 Establ
  bgp.evpn.0: 4/4/4/0
  pbbn1.evpn.0: 4/4/4/0
  __default_evpn__.evpn.0: 0/0/0/0
127.0.0.3      65221      7        7        0        0      1:25 Establ
  bgp.evpn.0: 4/4/4/0
  pbbn1.evpn.0: 4/4/4/0
  __default_evpn__.evpn.0: 0/0/0/0
```

Meaning

A BGP session is established between the PE devices.

Verifying MPLS LSPs

Purpose

Verify the MPLS LSP status on Device PE1.

Action

From operational mode, run the `show mpls lsp` command.

```
user@PE1> show mpls lsp
Ingress LSP: 2 sessions
To          From      State Rt P    ActivePath    LSPname
127.0.0.2   127.0.0.1 Up      0 *          PE1toPE2
127.0.0.3   127.0.0.1 Up      0 *          PE1toPE3
Total 2 displayed, Up 2, Down 0
```

Egress LSP: 2 sessions

To	From	State	Rt	Style	Labelin	Labelout	LSPname
127.0.0.1	127.0.0.3	Up	0	1 FF	3	-	PE3toPE1
127.0.0.1	127.0.0.2	Up	0	1 FF	3	-	PE2toPE1

Total 2 displayed, Up 2, Down 0

Transit LSP: 0 sessions

Total 0 displayed, Up 0, Down 0

Verifying the EVPN Routing Instance

Purpose

Verify the EVPN routing instance information.

Action

From operational mode, run the `show evpn instance extensive` command.

```
user@PE1> show evpn instance extensive
```

Instance: __default_evpn__

Route Distinguisher: 127.0.0.1:0

Number of bridge domains: 0

Number of neighbors: 0

Instance: pbbn1

Route Distinguisher: 127.0.0.1:100

Per-instance MAC route label: 16

Per-instance multicast route label: 17

PBB EVPN Core enabled

Control word enabled

MAC database status	Local	Remote
MAC advertisements:	2	4
MAC+IP advertisements:	0	0
Default gateway MAC advertisements:	0	0

Number of local interfaces: 1 (1 up)

Interface name	ESI	Mode	Status	AC-Role
cbp0.0	00:00:00:00:00:00:00:00:00:00	single-homed	Up	Root

Number of IRB interfaces: 0 (0 up)

```

Number of bridge domains: 2
  VLAN  Domain ID  Intfs / up  IRB intf  Mode          MAC sync  IM route label  SG
sync  IM core nexthop
      1000          0   0              Extended    Enabled   17
Disabled
      2000          0   0              Extended    Enabled   17
Disabled
Number of Bundle bridge domains: 0
Number of neighbors: 2
  Address          MAC      MAC+IP      AD      IM      ES Leaf-label
  127.0.0.2        2        0          0        2        0
  127.0.0.3        2        0          0        2        0
Number of ethernet segments: 0

```

Meaning

The output displays the `pbbn1` routing instance information, such as the integration of PBB with EVPN, the single-homed EVPN mode of operation, and the IP address of Devices PE2 and PE3 as the neighbors.

Verifying Routing Table Entries of the EVPN Routing Instance

Purpose

Verify the routing table entries of the EVPN routing instance.

Action

From operational mode, run the `show route table pbbn1.evpn.0` command.

```

user@PE1> show route table pbbn1.evpn.0
pbbn1.evpn.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2:127.0.0.1:100::1000::00:50:50:50:50/304 MAC/IP
      *[EVPN/170] 00:04:20
      Indirect
2:127.0.0.1:100::2000::00:1d:b5:a2:47:b0/304 MAC/IP
      *[EVPN/170] 00:04:20
      Indirect
2:127.0.0.2:100::1000::00:51:51:51:51/304 MAC/IP

```

```

*[BGP/170] 00:02:50, localpref 100, from 127.0.0.2
  AS path: I, validation-state: unverified
  > to 10.0.0.2 via irb.0, label-switched-path PE1toPE2
2:127.0.0.2:100::2000::00:23:9c:5e:a7:b0/304 MAC/IP
*[BGP/170] 00:02:50, localpref 100, from 127.0.0.2
  AS path: I, validation-state: unverified
  > to 10.0.0.2 via irb.0, label-switched-path PE1toPE2
2:127.0.0.3:100::1000::00:52:52:52:52:52/304 MAC/IP
*[BGP/170] 00:02:05, localpref 100, from 127.0.0.3
  AS path: I, validation-state: unverified
  > to 172.17.0.1 via ge-2/1/1.0, label-switched-path PE1toPE3
2:127.0.0.3:100::2000::5c:5e:ab:0d:3a:b8/304 MAC/IP
*[BGP/170] 00:02:05, localpref 100, from 127.0.0.3
  AS path: I, validation-state: unverified
  > to 172.17.0.1 via ge-2/1/1.0, label-switched-path PE1toPE3
3:127.0.0.1:100::1000::127.0.0.1/248 IM
*[EVPN/170] 00:04:20
  Indirect
3:127.0.0.1:100::2000::127.0.0.1/248 IM
*[EVPN/170] 00:04:20
  Indirect
3:127.0.0.2:100::1000::127.0.0.2/248 IM
*[BGP/170] 00:02:50, localpref 100, from 127.0.0.2
  AS path: I, validation-state: unverified
  > to 10.0.0.2 via irb.0, label-switched-path PE1toPE2
3:127.0.0.2:100::2000::127.0.0.2/248 IM
*[BGP/170] 00:02:50, localpref 100, from 127.0.0.2
  AS path: I, validation-state: unverified
  > to 10.0.0.2 via irb.0, label-switched-path PE1toPE2
3:127.0.0.3:100::1000::127.0.0.3/248 IM
*[BGP/170] 00:02:05, localpref 100, from 127.0.0.3
  AS path: I, validation-state: unverified
  > to 172.17.0.1 via ge-2/1/1.0, label-switched-path PE1toPE3
3:127.0.0.3:100::2000::127.0.0.3/248 IM
*[BGP/170] 00:02:05, localpref 100, from 127.0.0.3
  AS path: I, validation-state: unverified
  > to 172.17.0.1 via ge-2/1/1.0, label-switched-path PE1toPE3

```

Meaning

The output displays the use of IRB interfaces for routing the LSPs between the PE devices.

Verifying the EVPN Database

Purpose

Verify the EVPN database information on the PE devices.

Action

From operational mode, run the `show evpn database` command.

```
user@PE1> show evpn database
```

```
Instance: pbbn1
```

VLAN	DomainId	MAC address	Active source	Timestamp	IP address
2000		00:1d:b5:a2:47:b0	Local	Apr 14 13:48:51	
	2000	00:23:9c:5e:a7:b0	127.0.0.2	Apr 14 13:53:04	
2000		5c:5e:ab:0d:3a:b8	127.0.0.3	Apr 14 13:53:38	
1000		00:50:50:50:50:50	Local	Apr 14 13:48:51	
1000		00:51:51:51:51:51	127.0.0.2	Apr 14 13:53:04	
1000		00:52:52:52:52:52	127.0.0.3	Apr 14 13:53:38	

```
user@PE2> show evpn database
```

```
Instance: pbbn1
```

VLAN	DomainId	MAC address	Active source	Timestamp	IP address
2000		00:1d:b5:a2:47:b0	127.0.0.1	Apr 14 13:53:04	
2000		00:23:9c:5e:a7:b0	Local	Apr 14 13:48:46	
2000		5c:5e:ab:0d:3a:b8	127.0.0.3	Apr 14 13:53:37	
1000		00:50:50:50:50:50	127.0.0.1	Apr 14 13:53:04	
1000		00:51:51:51:51:51	Local	Apr 14 13:48:46	
1000		00:52:52:52:52:52	127.0.0.3	Apr 14 13:53:37	

```
user@PE3> show evpn database
```

```
Instance: pbbn1
```

VLAN	DomainId	MAC address	Active source	Timestamp	IP address
1000		00:50:50:50:50:50	127.0.0.1	Apr 14 13:53:34	
1000		00:51:51:51:51:51	127.0.0.2	Apr 14 13:53:27	
1000		00:52:52:52:52:52	Local	Apr 14 13:52:04	
2000		00:1d:b5:a2:47:b0	127.0.0.1	Apr 14 13:53:34	

```

2000      00:23:9c:5e:a7:b0  127.0.0.2      Apr 14 13:53:27
2000      5c:5e:ab:0d:3a:b8  Local

```

Verifying the MAC Table Entries

Purpose

Verify the bridge MAC table entries.

Action

From operational mode, run the `show bridge mac-table` command.

```

user@PE1> show bridge mac-table
MAC flags      (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
               0 -OVSDDB MAC, SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P -Pinned
               MAC)

Routing instance : default-switch
Bridging domain : bd, VLAN : none
  MAC          MAC      Logical      NH      MAC
  address      flags    interface  Index  property
  00:23:9c:5e:a7:f0  D      ae6.0
MAC flags      (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
               0 -OVSDDB MAC, SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P -Pinned
               MAC)

Routing instance : pbbn1
Bridging domain : bda, VLAN : 100
  MAC          MAC      Logical      NH      MAC
  address      flags    interface  Index  property
  00:51:51:51:51:51  DC
  00:52:52:52:52:52  DC
  01:1e:83:00:03:e8  DC
MAC flags      (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
               0 -OVSDDB MAC, SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P -Pinned
               MAC)

Routing instance : pbbn1

```

Bridging domain : bdb, VLAN : 200

MAC address	MAC flags	Logical interface	NH Index	MAC property
00:23:9c:5e:a7:b0	DC		1048576	
01:1e:83:00:07:d0	DC		1048577	
5c:5e:ab:0d:3a:b8	DC		1048581	

MAC flags (S -static MAC, D -dynamic MAC,
SE -Statistics enabled, NM -Non configured MAC, P -Pinned MAC)

Routing instance : pbn1

Bridging domain : bda, ISID : 1000, VLAN : 10

MAC address	MAC flags	Logical interface	Remote BEB address
00:00:00:00:0a:00	D	xe-1/0/0.0	
00:00:00:00:0a:01	D	xe-1/0/0.0	
00:00:00:00:0a:02	D	xe-1/0/0.0	
00:00:00:00:0a:03	D	xe-1/0/0.0	
00:00:00:00:0a:04	D	xe-1/0/0.0	
00:00:00:00:0a:05	D	xe-1/0/0.0	
00:00:00:00:0a:06	D	xe-1/0/0.0	
00:00:00:00:0a:07	D	xe-1/0/0.0	
00:00:00:00:0a:08	D	xe-1/0/0.0	
00:00:00:00:0a:09	D	xe-1/0/0.0	

Meaning

The output displays the MAC addresses associated with the ae6 aggregated Ethernet bundle.

Verifying the inet.3 Routing Table Entries

Purpose

Verify the inet.3 routing table entries on Device PE1.

Action

From operational mode, run the show route table inet.3 command.

```
user@PE1> show route table inet.3
inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
```


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

```
127.0.0.2/8    *[RSVP/7/1] 00:11:15, metric 1
                > to 10.0.0.2 via irb.0, label-switched-path PE1toPE2
127.0.0.3/8    *[RSVP/7/1] 00:09:48, metric 1
                > to 172.17.0.1 via ge-2/1/1.0, label-switched-path PE1toPE3
```

Meaning

The LSPs to Device PE2 and PE3 are routed using the IRB interface.

RELATED DOCUMENTATION

[Provider Backbone Bridging \(PBB\) and EVPN Integration Overview | 1897](#)

[Example: Configuring PBB with Multihomed EVPN | 1968](#)

pbb-evpn-core

Example: Configuring PBB with Multihomed EVPN

IN THIS SECTION

- [Requirements | 1968](#)
- [Overview and Topology | 1969](#)
- [Configuration | 1970](#)
- [Verification | 2002](#)

This example shows how to integrate provider backbone bridging (PBB) with Ethernet VPN (EVPN). With this integration, the control plane operations in the core are simplified, providing faster convergence and scalability enhancements than regular EVPN. The PBB-EVPN applications include Data Center Interconnect (DCI) and carrier Ethernet E-LAN services.

Requirements

This example uses the following hardware and software components:

- Four provider edge (PE) devices connected to two common multihomed customer sites, and each PE device is connected to a host.
- Two multihomed customer edge (CE) devices.
- Junos OS Release 17.2R1 or later running on all the PE routers.

Before you begin:

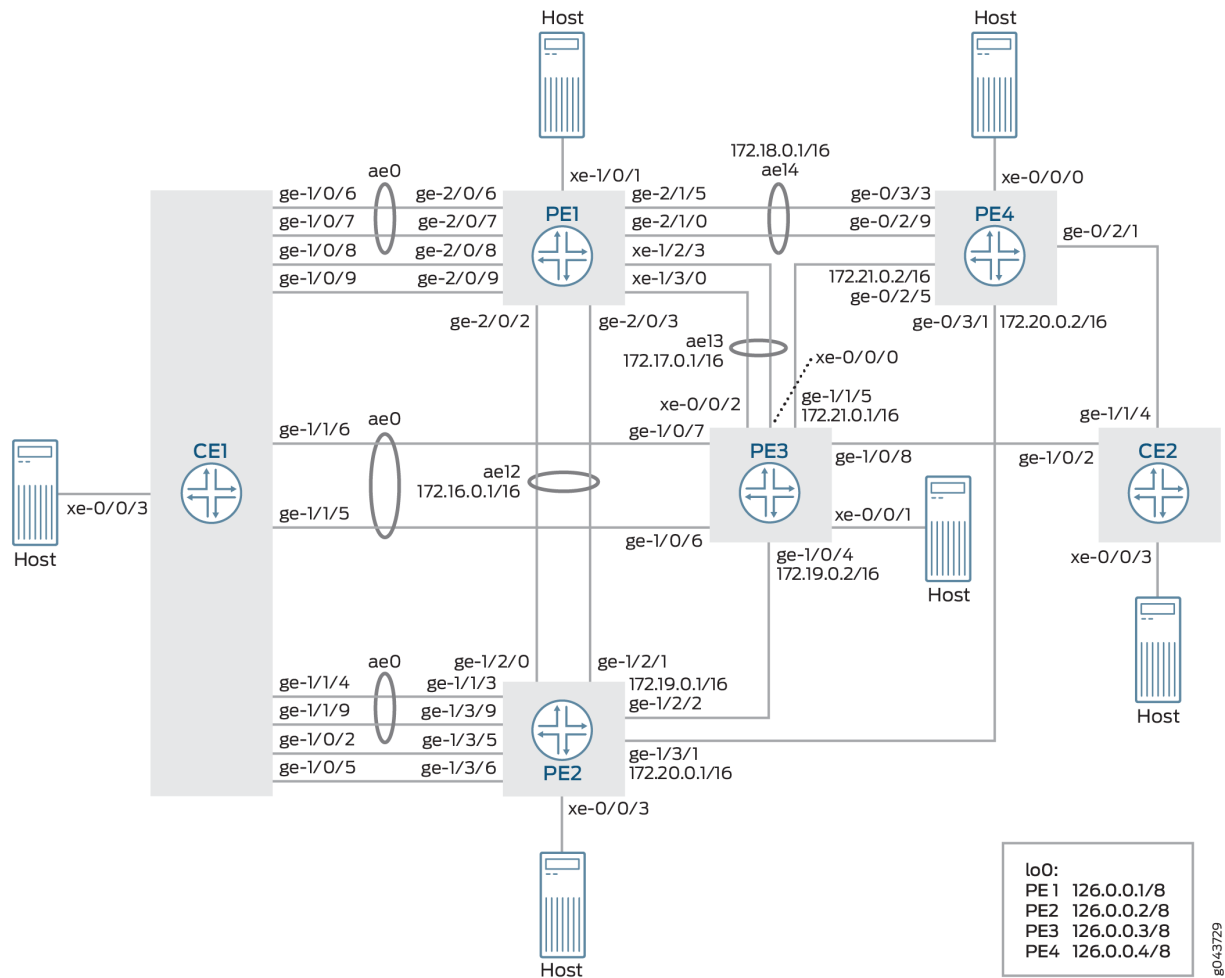
- Configure the device interfaces.
- Configure an IGP, such as OSPF, on all the PE devices.
- Establish an internal BGP session between the PE devices.
- Enable RSVP on the PE devices.
- Configure MPLS and label-switched paths (LSPs) between the PE devices.

Overview and Topology

With the introduction of the Integrating PBB with EVPN feature, PBB became integrated with Ethernet VPN (EVPN) to enable significant reduction in the control plane learning across the core, allowing a huge number of Layer 2 services, such as data center connectivity, to transit the network in a simplified manner. See the [Integrating PBB with EVPN](#) entry in [Feature Explorer](#) for platform and software release support for the Integrating PBB with EVPN feature.

In a PBB-EVPN network, the backbone core bridge (BCB) device in the PBB core is replaced with MPLS, while retaining the service scaling properties of the PBB backbone edge bridge (BEB). The B-component (provider routing instance) is signaled using EVPN BGP signaling and encapsulated inside MPLS using provider edge (PE) and provider (P) devices. Thus, PBB-EVPN combines the vast scaling property of PBB with the simplicity of a traditional basic MPLS core network, resulting in significant reduction in the amount of network-wide state information, as opposed to regular PBB.

Figure 211: PBB with Active/Standby EVPN Multihoming



In [Figure 211 on page 1970](#), PBB is integrated with EVPN, where the CE devices are multihomed in the active/standby mode. Device CE1 is multihomed to Devices PE1, PE2, and PE3, and Device CE2 is multihomed to Device PE3 and PE4.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 1971](#)
- [Configuring Device CE1 | 1983](#)
- [Configuring Device PE1 | 1988](#)

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.

CE1

```
set chassis aggregated-devices ethernet device-count 50
set chassis network-services enhanced-ip
set interfaces xe-0/0/3 flexible-vlan-tagging
set interfaces xe-0/0/3 encapsulation flexible-ethernet-services
set interfaces xe-0/0/3 unit 0 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 0 vlan-id 10
set interfaces xe-0/0/3 unit 1 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 1 vlan-id 20
set interfaces xe-0/0/3 unit 2 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 2 vlan-id 30
set interfaces ge-1/0/2 flexible-vlan-tagging
set interfaces ge-1/0/2 encapsulation flexible-ethernet-services
set interfaces ge-1/0/2 unit 1 encapsulation vlan-bridge
set interfaces ge-1/0/2 unit 1 vlan-id 20
set interfaces ge-1/0/5 flexible-vlan-tagging
set interfaces ge-1/0/5 encapsulation flexible-ethernet-services
set interfaces ge-1/0/5 unit 2 encapsulation vlan-bridge
set interfaces ge-1/0/5 unit 2 vlan-id 30
set interfaces ge-1/0/6 gigether-options 802.3ad ae0
set interfaces ge-1/0/7 gigether-options 802.3ad ae0
set interfaces ge-1/0/8 flexible-vlan-tagging
set interfaces ge-1/0/8 encapsulation flexible-ethernet-services
set interfaces ge-1/0/8 unit 1 encapsulation vlan-bridge
set interfaces ge-1/0/8 unit 1 vlan-id 20
set interfaces ge-1/0/9 flexible-vlan-tagging
set interfaces ge-1/0/9 encapsulation flexible-ethernet-services
set interfaces ge-1/0/9 unit 2 encapsulation vlan-bridge
set interfaces ge-1/0/9 unit 2 vlan-id 30
set interfaces ge-1/1/4 gigether-options 802.3ad ae0
set interfaces ge-1/1/5 gigether-options 802.3ad ae0
set interfaces ge-1/1/6 gigether-options 802.3ad ae0
set interfaces ge-1/1/9 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
```

```

set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set bridge-domains bd10 domain-type bridge
set bridge-domains bd10 vlan-id 10
set bridge-domains bd10 interface ae0.0
set bridge-domains bd10 interface xe-0/0/3.0
set bridge-domains bd20 domain-type bridge
set bridge-domains bd20 vlan-id 20
set bridge-domains bd20 interface xe-0/0/3.1
set bridge-domains bd20 interface ge-1/0/8.1
set bridge-domains bd20 interface ge-1/0/2.1
set bridge-domains bd30 domain-type bridge
set bridge-domains bd30 vlan-id 30
set bridge-domains bd30 interface xe-0/0/3.2
set bridge-domains bd30 interface ge-1/0/9.2
set bridge-domains bd30 interface ge-1/0/5.2

```

CE2

```

set chassis aggregated-devices ethernet device-count 50
set chassis network-services enhanced-ip
set interfaces xe-0/0/3 flexible-vlan-tagging
set interfaces xe-0/0/3 encapsulation flexible-ethernet-services
set interfaces xe-0/0/3 unit 0 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 0 vlan-id 10
set interfaces xe-0/0/3 unit 0 family bridge filter output f_log
set interfaces xe-0/0/3 unit 1 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 1 vlan-id 20
set interfaces xe-0/0/3 unit 2 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 2 vlan-id 30
set interfaces ge-1/0/2 flexible-vlan-tagging
set interfaces ge-1/0/2 encapsulation flexible-ethernet-services
set interfaces ge-1/0/2 unit 0 encapsulation vlan-bridge
set interfaces ge-1/0/2 unit 0 vlan-id 10
set interfaces ge-1/0/2 unit 1 encapsulation vlan-bridge
set interfaces ge-1/0/2 unit 1 vlan-id 20
set interfaces ge-1/0/2 unit 2 encapsulation vlan-bridge
set interfaces ge-1/0/2 unit 2 vlan-id 30
set interfaces ge-1/1/4 flexible-vlan-tagging
set interfaces ge-1/1/4 encapsulation flexible-ethernet-services
set interfaces ge-1/1/4 unit 0 encapsulation vlan-bridge

```

```

set interfaces ge-1/1/4 unit 0 vlan-id 10
set interfaces ge-1/1/4 unit 1 encapsulation vlan-bridge
set interfaces ge-1/1/4 unit 1 vlan-id 20
set interfaces ge-1/1/4 unit 2 encapsulation vlan-bridge
set interfaces ge-1/1/4 unit 2 vlan-id 30
set bridge-domains bd10 domain-type bridge
set bridge-domains bd10 vlan-id 10
set bridge-domains bd10 interface ge-1/1/4.0
set bridge-domains bd10 interface ge-1/0/2.0
set bridge-domains bd10 interface xe-0/0/3.0
set bridge-domains bd20 domain-type bridge
set bridge-domains bd20 vlan-id 20
set bridge-domains bd20 interface ge-1/1/4.1
set bridge-domains bd20 interface ge-1/0/2.1
set bridge-domains bd20 interface xe-0/0/3.1
set bridge-domains bd30 domain-type bridge
set bridge-domains bd30 vlan-id 30
set bridge-domains bd30 interface xe-0/0/3.2
set bridge-domains bd30 interface ge-1/1/4.2
set bridge-domains bd30 interface ge-1/0/2.2

```

PE1

```

set chassis aggregated-devices ethernet device-count 50
set chassis network-services enhanced-ip
set interfaces xe-1/0/1 flexible-vlan-tagging
set interfaces xe-1/0/1 encapsulation flexible-ethernet-services
set interfaces xe-1/0/1 unit 0 encapsulation vlan-bridge
set interfaces xe-1/0/1 unit 0 vlan-id 10
set interfaces xe-1/2/3 gigether-options 802.3ad ae13
set interfaces xe-1/3/0 gigether-options 802.3ad ae13
set interfaces ge-2/0/2 gigether-options 802.3ad ae12
set interfaces ge-2/0/3 gigether-options 802.3ad ae12
set interfaces ge-2/0/6 gigether-options 802.3ad ae0
set interfaces ge-2/0/7 gigether-options 802.3ad ae0
set interfaces ge-2/0/8 flexible-vlan-tagging
set interfaces ge-2/0/8 encapsulation flexible-ethernet-services
set interfaces ge-2/0/8 esi 00:22:22:22:22:22:22:22:22:22
set interfaces ge-2/0/8 esi single-active
set interfaces ge-2/0/8 esi source-bmac 00:22:22:22:22:22
set interfaces ge-2/0/8 unit 0 encapsulation vlan-bridge
set interfaces ge-2/0/8 unit 0 vlan-id 20

```

```

set interfaces ge-2/0/9 flexible-vlan-tagging
set interfaces ge-2/0/9 encapsulation flexible-ethernet-services
set interfaces ge-2/0/9 esi 00:33:33:33:33:33:33:33:33:33
set interfaces ge-2/0/9 esi single-active
set interfaces ge-2/0/9 esi source-bmac 00:33:33:33:33:33
set interfaces ge-2/0/9 unit 0 encapsulation vlan-bridge
set interfaces ge-2/0/9 unit 0 vlan-id 30
set interfaces ge-2/1/0 gigether-options 802.3ad ae14
set interfaces ge-2/1/5 gigether-options 802.3ad ae14
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 esi single-active
set interfaces ae0 esi source-bmac 00:11:11:11:11:11
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set interfaces ae12 flexible-vlan-tagging
set interfaces ae12 encapsulation flexible-ethernet-services
set interfaces ae12 aggregated-ether-options minimum-links 1
set interfaces ae12 unit 0 vlan-id 1200
set interfaces ae12 unit 0 family inet address 172.16.0.1/16
set interfaces ae12 unit 0 family iso
set interfaces ae12 unit 0 family mpls
set interfaces ae13 flexible-vlan-tagging
set interfaces ae13 encapsulation flexible-ethernet-services
set interfaces ae13 aggregated-ether-options minimum-links 1
set interfaces ae13 unit 0 vlan-tags outer 1300
set interfaces ae13 unit 0 vlan-tags inner 13
set interfaces ae13 unit 0 family inet address 172.17.0.1/16
set interfaces ae13 unit 0 family iso
set interfaces ae13 unit 0 family mpls
set interfaces ae14 aggregated-ether-options minimum-links 1
set interfaces ae14 unit 0 family inet address 172.18.0.1/16
set interfaces ae14 unit 0 family iso
set interfaces ae14 unit 0 family mpls
set interfaces cbp0 unit 0 family bridge interface-mode trunk
set interfaces cbp0 unit 0 family bridge bridge-domain-type bvlan
set interfaces cbp0 unit 0 family bridge isid-list all
set interfaces cbp0 unit 1 family bridge interface-mode trunk
set interfaces cbp0 unit 1 family bridge bridge-domain-type bvlan
set interfaces cbp0 unit 1 family bridge isid-list all
set interfaces lo0 unit 0 family inet address 127.0.0.1/8 primary
set interfaces lo0 unit 0 family iso

```

```

set interfaces pip0 unit 0 family bridge interface-mode trunk
set interfaces pip0 unit 0 family bridge bridge-domain-type svlan
set interfaces pip0 unit 0 family bridge isid-list all-service-groups
set interfaces pip0 unit 1 family bridge interface-mode trunk
set interfaces pip0 unit 1 family bridge bridge-domain-type svlan
set interfaces pip0 unit 1 family bridge isid-list all-service-groups
set routing-options router-id 127.0.0.1
set routing-options autonomous-system 65221
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path pe1tope2 from 127.0.0.1
set protocols mpls label-switched-path pe1tope2 to 127.0.0.2
set protocols mpls label-switched-path pe1tope2 primary direct_to_pe2
set protocols mpls label-switched-path pe1tope3 from 127.0.0.1
set protocols mpls label-switched-path pe1tope3 to 127.0.0.3
set protocols mpls label-switched-path pe1tope3 primary direct_to_pe3
set protocols mpls label-switched-path pe1tope4 from 127.0.0.1
set protocols mpls label-switched-path pe1tope4 to 127.0.0.4
set protocols mpls label-switched-path pe1tope4 primary direct_to_pe4
set protocols mpls path direct_to_pe2 100.12.1.2 strict
set protocols mpls path direct_to_pe3 100.13.1.3 strict
set protocols mpls path direct_to_pe4 100.14.1.4 strict
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 127.0.0.1
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 127.0.0.2
set protocols bgp group ibgp neighbor 127.0.0.3
set protocols bgp group ibgp neighbor 127.0.0.4
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances pbbn1 instance-type virtual-switch
set routing-instances pbbn1 interface cbp0.0
set routing-instances pbbn1 route-distinguisher 127.0.0.1:1
set routing-instances pbbn1 vrf-target target:100:1
set routing-instances pbbn1 protocols evpn pbb-evpn-core
set routing-instances pbbn1 protocols evpn extended-isid-list 1000
set routing-instances pbbn1 bridge-domains bda vlan-id 100
set routing-instances pbbn1 bridge-domains bda isid-list 1000
set routing-instances pbbn1 bridge-domains bda vlan-id-scope-local
set routing-instances pbn1 instance-type virtual-switch

```



```

set routing-instances pbn1 interface pip0.0
set routing-instances pbn1 bridge-domains bda domain-type bridge
set routing-instances pbn1 bridge-domains bda vlan-id 10
set routing-instances pbn1 bridge-domains bda interface ae0.0
set routing-instances pbn1 bridge-domains bda interface xe-1/0/1.0
set routing-instances pbn1 bridge-domains bda interface ge-2/0/9.0
set routing-instances pbn1 bridge-domains bda interface ge-2/0/8.0
set routing-instances pbn1 pbb-options peer-instance pbbn1
set routing-instances pbn1 service-groups sga service-type elan
set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-id-list 10

```

PE2

```

set chassis aggregated-devices ethernet device-count 50
set chassis network-services enhanced-ip
set interfaces xe-0/0/3 flexible-vlan-tagging
set interfaces xe-0/0/3 encapsulation flexible-ethernet-services
set interfaces xe-0/0/3 unit 0 encapsulation vlan-bridge
set interfaces xe-0/0/3 unit 0 vlan-id 10
set interfaces ge-1/2/0 gigether-options 802.3ad ae12
set interfaces ge-1/2/1 gigether-options 802.3ad ae12
set interfaces ge-1/2/2 unit 0 family inet address 172.19.0.1/16
set interfaces ge-1/2/2 unit 0 family iso
set interfaces ge-1/2/2 unit 0 family mpls
set interfaces ge-1/3/1 unit 0 family inet address 172.20.0.1/16
set interfaces ge-1/3/1 unit 0 family iso
set interfaces ge-1/3/1 unit 0 family mpls
set interfaces ge-1/3/3 gigether-options 802.3ad ae0
set interfaces ge-1/3/5 flexible-vlan-tagging
set interfaces ge-1/3/5 encapsulation flexible-ethernet-services
set interfaces ge-1/3/5 esi 00:22:22:22:22:22:22:22:22
set interfaces ge-1/3/5 esi single-active
set interfaces ge-1/3/5 esi source-bmac 00:22:22:22:22:23
set interfaces ge-1/3/5 unit 0 encapsulation vlan-bridge
set interfaces ge-1/3/5 unit 0 vlan-id 20
set interfaces ge-1/3/6 flexible-vlan-tagging
set interfaces ge-1/3/6 encapsulation flexible-ethernet-services
set interfaces ge-1/3/6 esi 00:33:33:33:33:33:33:33:33
set interfaces ge-1/3/6 esi single-active
set interfaces ge-1/3/6 esi source-bmac 00:33:33:33:33:34
set interfaces ge-1/3/6 unit 0 encapsulation vlan-bridge
set interfaces ge-1/3/6 unit 0 vlan-id 30

```

```

set interfaces ge-1/3/9 gigether-options 802.3ad ae0
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11
set interfaces ae0 esi single-active
set interfaces ae0 esi source-bmac 00:11:11:11:11:12
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set interfaces ae0 unit 0 family bridge filter output f_log
set interfaces ae12 flexible-vlan-tagging
set interfaces ae12 encapsulation flexible-ethernet-services
set interfaces ae12 aggregated-ether-options minimum-links 1
set interfaces ae12 unit 0 vlan-id 1200
set interfaces ae12 unit 0 family inet address 100.12.1.2/24
set interfaces ae12 unit 0 family iso
set interfaces ae12 unit 0 family mpls
set interfaces cbp0 unit 0 family bridge interface-mode trunk
set interfaces cbp0 unit 0 family bridge bridge-domain-type bvlan
set interfaces cbp0 unit 0 family bridge isid-list all
set interfaces cbp0 unit 1 family bridge interface-mode trunk
set interfaces cbp0 unit 1 family bridge bridge-domain-type bvlan
set interfaces cbp0 unit 1 family bridge isid-list all
set interfaces lo0 unit 0 family inet address 127.0.0.2/8 primary
set interfaces lo0 unit 0 family iso
set interfaces pip0 unit 0 family bridge interface-mode trunk
set interfaces pip0 unit 0 family bridge bridge-domain-type svlan
set interfaces pip0 unit 0 family bridge isid-list all-service-groups
set interfaces pip0 unit 1 family bridge interface-mode trunk
set interfaces pip0 unit 1 family bridge bridge-domain-type svlan
set interfaces pip0 unit 1 family bridge isid-list all-service-groups
set routing-options router-id 127.0.0.2
set routing-options autonomous-system 65221
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path pe2tope1 from 127.0.0.2
set protocols mpls label-switched-path pe2tope1 to 127.0.0.1
set protocols mpls label-switched-path pe2tope1 primary direct_to_pe1
set protocols mpls label-switched-path pe2tope3 from 127.0.0.2
set protocols mpls label-switched-path pe2tope3 to 127.0.0.3
set protocols mpls label-switched-path pe2tope3 primary direct_to_pe3
set protocols mpls label-switched-path pe2tope4 from 127.0.0.2
set protocols mpls label-switched-path pe2tope4 to 127.0.0.4
set protocols mpls label-switched-path pe2tope4 primary direct_to_pe4

```

```

set protocols mpls path direct_to_pe1 172.16.0.1 strict
set protocols mpls path direct_to_pe3 100.23.1.3 strict
set protocols mpls path direct_to_pe4 172.20.0.2 strict
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 127.0.0.2
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 127.0.0.1
set protocols bgp group ibgp neighbor 127.0.0.3
set protocols bgp group ibgp neighbor 127.0.0.4
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 all
set routing-instances pbbn1 instance-type virtual-switch
set routing-instances pbbn1 interface cbp0.0
set routing-instances pbbn1 route-distinguisher 127.0.0.2:1
set routing-instances pbbn1 vrf-target target:100:1
set routing-instances pbbn1 protocols evpn pbb-evpn-core
set routing-instances pbbn1 protocols evpn extended-isid-list 1000
set routing-instances pbbn1 bridge-domains bda vlan-id 100
set routing-instances pbbn1 bridge-domains bda isid-list 1000
set routing-instances pbbn1 bridge-domains bda vlan-id-scope-local
set routing-instances pbn1 instance-type virtual-switch
set routing-instances pbn1 interface pip0.0
set routing-instances pbn1 bridge-domains bda domain-type bridge
set routing-instances pbn1 bridge-domains bda vlan-id 10
set routing-instances pbn1 bridge-domains bda interface ae0.0
set routing-instances pbn1 bridge-domains bda interface xe-0/0/3.0
set routing-instances pbn1 bridge-domains bda interface ge-1/3/6.0
set routing-instances pbn1 bridge-domains bda interface ge-1/3/5.0
set routing-instances pbn1 bridge-domains bda bridge-options mac-statistics
set routing-instances pbn1 pbb-options peer-instance pbbn1
set routing-instances pbn1 service-groups sga service-type elan
set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-id-list 10

```

PE3

```

set chassis aggregated-devices ethernet device-count 50
set chassis network-services enhanced-ip
set interfaces xe-0/0/0 gigether-options 802.3ad ae13
set interfaces xe-0/0/1 flexible-vlan-tagging

```

```

set interfaces xe-0/0/1 encapsulation flexible-ethernet-services
set interfaces xe-0/0/1 unit 0 encapsulation vlan-bridge
set interfaces xe-0/0/1 unit 0 vlan-id 10
set interfaces xe-0/0/2 gigether-options 802.3ad ae13
set interfaces ge-1/0/4 unit 0 family inet address 100.23.1.3/24
set interfaces ge-1/0/4 unit 0 family iso
set interfaces ge-1/0/4 unit 0 family mpls
set interfaces ge-1/0/6 gigether-options 802.3ad ae0
set interfaces ge-1/0/7 gigether-options 802.3ad ae0
set interfaces ge-1/0/8 flexible-vlan-tagging
set interfaces ge-1/0/8 encapsulation flexible-ethernet-services
set interfaces ge-1/0/8 esi 00:44:44:44:44:44:44:44:44:44
set interfaces ge-1/0/8 esi single-active
set interfaces ge-1/0/8 esi source-bmac 00:44:44:44:44:44
set interfaces ge-1/0/8 unit 0 encapsulation vlan-bridge
set interfaces ge-1/0/8 unit 0 vlan-id 10
set interfaces ge-1/1/5 unit 0 family inet address 172.21.0.1/16
set interfaces ge-1/1/5 unit 0 family iso
set interfaces ge-1/1/5 unit 0 family mpls
set interfaces ae0 flexible-vlan-tagging
set interfaces ae0 encapsulation flexible-ethernet-services
set interfaces ae0 esi 00:11:11:11:11:11:11:11:11:11
set interfaces ae0 esi single-active
set interfaces ae0 esi source-bmac 00:11:11:11:11:13
set interfaces ae0 unit 0 encapsulation vlan-bridge
set interfaces ae0 unit 0 vlan-id 10
set interfaces ae13 flexible-vlan-tagging
set interfaces ae13 encapsulation flexible-ethernet-services
set interfaces ae13 aggregated-ether-options minimum-links 1
set interfaces ae13 unit 0 vlan-tags outer 1300
set interfaces ae13 unit 0 vlan-tags inner 13
set interfaces ae13 unit 0 family inet address 100.13.1.3/24
set interfaces ae13 unit 0 family iso
set interfaces ae13 unit 0 family mpls
set interfaces cbp0 unit 0 family bridge interface-mode trunk
set interfaces cbp0 unit 0 family bridge bridge-domain-type bvlan
set interfaces cbp0 unit 0 family bridge isid-list all
set interfaces cbp0 unit 1 family bridge interface-mode trunk
set interfaces cbp0 unit 1 family bridge bridge-domain-type bvlan
set interfaces cbp0 unit 1 family bridge isid-list all
set interfaces lo0 unit 0 family inet address 127.0.0.3/8 primary
set interfaces lo0 unit 0 family iso
set interfaces pip0 unit 0 family bridge interface-mode trunk

```

```

set interfaces pip0 unit 0 family bridge bridge-domain-type svlan
set interfaces pip0 unit 0 family bridge isid-list all-service-groups
set interfaces pip0 unit 1 family bridge interface-mode trunk
set interfaces pip0 unit 1 family bridge bridge-domain-type svlan
set interfaces pip0 unit 1 family bridge isid-list all-service-groups
set routing-options router-id 127.0.0.3
set routing-options autonomous-system 65221
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path pe3tope1 from 127.0.0.3
set protocols mpls label-switched-path pe3tope1 to 127.0.0.1
set protocols mpls label-switched-path pe3tope1 primary direct_to_pe1
set protocols mpls label-switched-path pe3tope2 from 127.0.0.3
set protocols mpls label-switched-path pe3tope2 to 127.0.0.2
set protocols mpls label-switched-path pe3tope2 primary direct_to_pe2
set protocols mpls label-switched-path pe3tope4 from 127.0.0.3
set protocols mpls label-switched-path pe3tope4 to 127.0.0.4
set protocols mpls label-switched-path pe3tope4 primary direct_to_pe4
set protocols mpls path direct_to_pe1 172.17.0.1 strict
set protocols mpls path direct_to_pe2 172.19.0.1 strict
set protocols mpls path direct_to_pe4 172.21.0.2 strict
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 127.0.0.3
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 127.0.0.1
set protocols bgp group ibgp neighbor 127.0.0.2
set protocols bgp group ibgp neighbor 127.0.0.4
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances pbbn1 instance-type virtual-switch
set routing-instances pbbn1 interface cbp0.0
set routing-instances pbbn1 route-distinguisher 127.0.0.3:1
set routing-instances pbbn1 vrf-target target:100:1
set routing-instances pbbn1 protocols evpn traceoptions file pbbevpn.log
set routing-instances pbbn1 protocols evpn traceoptions file size 500m
set routing-instances pbbn1 protocols evpn traceoptions flag all
set routing-instances pbbn1 protocols evpn control-word
set routing-instances pbbn1 protocols evpn pbb-evpn-core
set routing-instances pbbn1 protocols evpn extended-isid-list 1000
set routing-instances pbbn1 bridge-domains bda vlan-id 100

```

```

set routing-instances pbbn1 bridge-domains bda isid-list 1000
set routing-instances pbbn1 bridge-domains bda vlan-id-scope-local
set routing-instances pbn1 instance-type virtual-switch
set routing-instances pbn1 interface pip0.0
set routing-instances pbn1 bridge-domains bda domain-type bridge
set routing-instances pbn1 bridge-domains bda vlan-id 10
set routing-instances pbn1 bridge-domains bda interface ae0.0
set routing-instances pbn1 bridge-domains bda interface xe-0/0/1.0
set routing-instances pbn1 bridge-domains bda interface ge-1/0/8.0
set routing-instances pbn1 bridge-domains bda bridge-options mac-statistics
set routing-instances pbn1 pbb-options peer-instance pbbn1
set routing-instances pbn1 service-groups sga service-type elan
set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-id-list 10

```

PE4

```

set chassis aggregated-devices ethernet device-count 50
set chassis network-services enhanced-ip
set interfaces xe-0/0/0 flexible-vlan-tagging
set interfaces xe-0/0/0 encapsulation flexible-ethernet-services
set interfaces xe-0/0/0 unit 0 encapsulation vlan-bridge
set interfaces xe-0/0/0 unit 0 vlan-id 10
set interfaces ge-0/2/1 flexible-vlan-tagging
set interfaces ge-0/2/1 encapsulation flexible-ethernet-services
set interfaces ge-0/2/1 esi 00:44:44:44:44:44:44:44:44:44
set interfaces ge-0/2/1 esi single-active
set interfaces ge-0/2/1 esi source-bmac 00:44:44:44:44:45
set interfaces ge-0/2/1 unit 0 encapsulation vlan-bridge
set interfaces ge-0/2/1 unit 0 vlan-id 10
set interfaces ge-0/2/5 unit 0 family inet address 172.21.0.2/16
set interfaces ge-0/2/5 unit 0 family iso
set interfaces ge-0/2/5 unit 0 family mpls
set interfaces ge-0/2/9 gigether-options 802.3ad ae14
set interfaces ge-0/3/1 unit 0 family inet address 172.20.0.2/16
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 gigether-options 802.3ad ae14
set interfaces ae14 aggregated-ether-options minimum-links 1
set interfaces ae14 unit 0 family inet address 100.14.1.4/24
set interfaces ae14 unit 0 family iso
set interfaces ae14 unit 0 family mpls
set interfaces cbp0 unit 0 family bridge interface-mode trunk

```

```

set interfaces cbp0 unit 0 family bridge bridge-domain-type bvlan
set interfaces cbp0 unit 0 family bridge isid-list all
set interfaces cbp0 unit 1 family bridge interface-mode trunk
set interfaces cbp0 unit 1 family bridge bridge-domain-type bvlan
set interfaces cbp0 unit 1 family bridge isid-list all
set interfaces lo0 unit 0 family inet address 127.0.0.4/8 primary
set interfaces pip0 unit 0 family bridge interface-mode trunk
set interfaces pip0 unit 0 family bridge bridge-domain-type svlan
set interfaces pip0 unit 0 family bridge isid-list all-service-groups
set interfaces pip0 unit 1 family bridge interface-mode trunk
set interfaces pip0 unit 1 family bridge bridge-domain-type svlan
set interfaces pip0 unit 1 family bridge isid-list all-service-groups
set routing-options router-id 127.0.0.4
set routing-options autonomous-system 65221
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path pe4tope1 from 127.0.0.4
set protocols mpls label-switched-path pe4tope1 to 127.0.0.1
set protocols mpls label-switched-path pe4tope1 primary direct_to_pe1
set protocols mpls label-switched-path pe4tope2 from 127.0.0.4
set protocols mpls label-switched-path pe4tope2 to 127.0.0.2
set protocols mpls label-switched-path pe4tope2 primary direct_to_pe2
set protocols mpls label-switched-path pe4tope3 from 127.0.0.4
set protocols mpls label-switched-path pe4tope3 to 127.0.0.3
set protocols mpls label-switched-path pe4tope3 primary direct_to_pe3
set protocols mpls path direct_to_pe1 172.18.0.1 strict
set protocols mpls path direct_to_pe2 172.20.0.1 strict
set protocols mpls path direct_to_pe3 172.21.0.1 strict
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 127.0.0.4
set protocols bgp group ibgp family evpn signaling
set protocols bgp group ibgp neighbor 127.0.0.1
set protocols bgp group ibgp neighbor 127.0.0.2
set protocols bgp group ibgp neighbor 127.0.0.3
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances pbbn1 instance-type virtual-switch
set routing-instances pbbn1 interface cbp0.0
set routing-instances pbbn1 route-distinguisher 127.0.0.4:1
set routing-instances pbbn1 vrf-target target:100:1

```

```

set routing-instances pbbn1 protocols evpn pbb-evpn-core
set routing-instances pbbn1 protocols evpn extended-isid-list 1000
set routing-instances pbbn1 bridge-domains bda vlan-id 100
set routing-instances pbbn1 bridge-domains bda isid-list 1000
set routing-instances pbbn1 bridge-domains bda vlan-id-scope-local
set routing-instances pbn1 instance-type virtual-switch
set routing-instances pbn1 interface pip0.0
set routing-instances pbn1 bridge-domains bda domain-type bridge
set routing-instances pbn1 bridge-domains bda vlan-id 10
set routing-instances pbn1 bridge-domains bda interface xe-0/0/0.0
set routing-instances pbn1 bridge-domains bda interface ge-0/2/1.0
set routing-instances pbn1 pbb-options peer-instance pbbn1
set routing-instances pbn1 service-groups sga service-type elan
set routing-instances pbn1 service-groups sga pbb-service-options isid 1000 vlan-id-list 10

```

Configuring Device CE1

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 CE1:

1. Set the number of aggregated Ethernet interfaces on Device CE1.

```

[edit chassis]
user@CE1# set aggregated-devices ethernet device-count 50

```

2. Set Device CE1's network services to enhanced Internet Protocol and use enhanced mode capabilities.

```

[edit chassis]
user@CE1# set chassis network-services enhanced-ip

```

3. Configure Device CE1's interfaces.

```

[edit interfaces]
user@CE1# set interfaces xe-0/0/3 flexible-vlan-tagging

```



```

user@CE1# set interfaces xe-0/0/3 encapsulation flexible-ethernet-services
user@CE1# set interfaces xe-0/0/3 unit 0 encapsulation vlan-bridge
user@CE1# set interfaces xe-0/0/3 unit 0 vlan-id 10
user@CE1# set interfaces xe-0/0/3 unit 1 encapsulation vlan-bridge
user@CE1# set interfaces xe-0/0/3 unit 1 vlan-id 20
user@CE1# set interfaces xe-0/0/3 unit 2 encapsulation vlan-bridge
user@CE1# set interfaces xe-0/0/3 unit 2 vlan-id 30
user@CE1# set interfaces ge-1/0/2 flexible-vlan-tagging
user@CE1# set interfaces ge-1/0/2 encapsulation flexible-ethernet-services
user@CE1# set interfaces ge-1/0/2 unit 1 encapsulation vlan-bridge
user@CE1# set interfaces ge-1/0/2 unit 1 vlan-id 20
user@CE1# set interfaces ge-1/0/5 flexible-vlan-tagging
user@CE1# set interfaces ge-1/0/5 encapsulation flexible-ethernet-services
user@CE1# set interfaces ge-1/0/5 unit 2 encapsulation vlan-bridge
user@CE1# set interfaces ge-1/0/5 unit 2 vlan-id 30
user@CE1# set interfaces ge-1/0/6 gigether-options 802.3ad ae0
user@CE1# set interfaces ge-1/0/7 gigether-options 802.3ad ae0
user@CE1# set interfaces ge-1/0/8 flexible-vlan-tagging
user@CE1# set interfaces ge-1/0/8 encapsulation flexible-ethernet-services
user@CE1# set interfaces ge-1/0/8 unit 1 encapsulation vlan-bridge
user@CE1# set interfaces ge-1/0/8 unit 1 vlan-id 20
user@CE1# set interfaces ge-1/0/9 flexible-vlan-tagging
user@CE1# set interfaces ge-1/0/9 encapsulation flexible-ethernet-services
user@CE1# set interfaces ge-1/0/9 unit 2 encapsulation vlan-bridge
user@CE1# set interfaces ge-1/0/9 unit 2 vlan-id 30
user@CE1# set interfaces ge-1/1/4 gigether-options 802.3ad ae0
user@CE1# set interfaces ge-1/1/5 gigether-options 802.3ad ae0
user@CE1# set interfaces ge-1/1/6 gigether-options 802.3ad ae0
user@CE1# set interfaces ge-1/1/9 gigether-options 802.3ad ae0

```

4. Configure the aggregated Ethernet bundle on Device CE1.

```

[edit interfaces]
user@CE1# set interfaces ae0 flexible-vlan-tagging
user@CE1# set interfaces ae0 encapsulation flexible-ethernet-services
user@CE1# set interfaces ae0 aggregated-ether-options minimum-links 1
user@CE1# set interfaces ae0 unit 0 encapsulation vlan-bridge
user@CE1# set interfaces ae0 unit 0 vlan-id 10

```

5. Configure the bridge domains on Device CE1.

```
[edit bridge-domains]
user@CE1# set bd10 domain-type bridge
user@CE1# set bd10 vlan-id 10
user@CE1# set bd10 interface ae0.0
user@CE1# set bd10 interface xe-0/0/3.0
user@CE1# set bd20 domain-type bridge
user@CE1# set bd20 vlan-id 20
user@CE1# set bd20 interface xe-0/0/3.1
user@CE1# set bd20 interface ge-1/0/8.1
user@CE1# set bd20 interface ge-1/0/2.1
user@CE1# set bd30 domain-type bridge
user@CE1# set bd30 vlan-id 30
user@CE1# set bd30 interface xe-0/0/3.2
user@CE1# set bd30 interface ge-1/0/9.2
user@CE1# set bd30 interface ge-1/0/5.2
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `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@CE1# show chassis
aggregated-devices {
  ethernet {
    device-count 50;
  }
}
network-services enhanced-ip;
```

```
user@CE1# show interfaces
xe-0/0/3 {
  flexible-vlan-tagging;
  encapsulation flexible-ethernet-services;
  unit 0 {
    encapsulation vlan-bridge;
    vlan-id 10;
```

```

    }
    unit 1 {
        encapsulation vlan-bridge;
        vlan-id 20;
    }
    unit 2 {
        encapsulation vlan-bridge;
        vlan-id 30;
    }
}
ge-1/0/2 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 1 {
        encapsulation vlan-bridge;
        vlan-id 20;
    }
}
ge-1/0/5 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 2 {
        encapsulation vlan-bridge;
        vlan-id 30;
    }
}
ge-1/0/6 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/0/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/0/8 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 1 {
        encapsulation vlan-bridge;
        vlan-id 20;
    }
}

```

```

}
ge-1/0/9 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 2 {
        encapsulation vlan-bridge;
        vlan-id 30;
    }
}
ge-1/1/4 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/1/5 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/1/6 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/1/9 {
    gigether-options {
        802.3ad ae0;
    }
}
ae0 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    aggregated-ether-options {
        minimum-links 1;
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 10;
    }
}

```

```

    }
}

```

```

user@CE1# show bridge-domains
bd10 {
    domain-type bridge;
    vlan-id 10;
    interface ae0.0;
    interface xe-0/0/3.0;
}
bd20 {
    domain-type bridge;
    vlan-id 20;
    interface xe-0/0/3.1;
    interface ge-1/0/8.1;
    interface ge-1/0/2.1;
}
bd30 {
    domain-type bridge;
    vlan-id 30;
    interface xe-0/0/3.2;
    interface ge-1/0/9.2;
    interface ge-1/0/5.2;
}

```

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

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:

1. Set the number of aggregated Ethernet interfaces on Device PE1.

```
[edit chassis]
user@PE1# set aggregated-devices ethernet device-count 50
```

2. Set Device PE1's network services to enhanced Internet Protocol and use enhanced mode capabilities.

```
[edit chassis]
user@PE1# set network-services enhanced-ip
```

3. Configure the CE-facing interfaces of Device PE1.

```
[edit interfaces]
user@PE1# set ge-2/0/6 gigether-options 802.3ad ae0
user@PE1# set ge-2/0/7 gigether-options 802.3ad ae0
user@PE1# set ge-2/0/8 flexible-vlan-tagging
user@PE1# set ge-2/0/8 encapsulation flexible-ethernet-services
user@PE1# set ge-2/0/8 unit 0 encapsulation vlan-bridge
user@PE1# set ge-2/0/8 unit 0 vlan-id 20
user@PE1# set ge-2/0/9 flexible-vlan-tagging
user@PE1# set ge-2/0/9 encapsulation flexible-ethernet-services
user@PE1# set ge-2/0/9 unit 0 encapsulation vlan-bridge
user@PE1# set ge-2/0/9 unit 0 vlan-id 30
```

4. Configure the aggregate Ethernet bundle on Device PE1 that connects to Device CE1.

```
[edit interfaces]
user@PE1# set ae0 flexible-vlan-tagging
user@PE1# set ae0 encapsulation flexible-ethernet-services
user@PE1# set ae0 unit 0 encapsulation vlan-bridge
user@PE1# set ae0 unit 0 vlan-id 10
```

5. Configure the EVPN multihoming parameters for the CE-facing interfaces and aggregate Ethernet bundle that connect to the multihomed customer site.

```
[edit interfaces]
user@PE1# set ge-2/0/8 esi 00:22:22:22:22:22:22:22:22
```

```

user@PE1# set ge-2/0/8 esi single-active
user@PE1# set ge-2/0/8 esi source-bmac 00:22:22:22:22:22
user@PE1# set ge-2/0/9 esi 00:33:33:33:33:33:33:33:33:33
user@PE1# set ge-2/0/9 esi single-active
user@PE1# set ge-2/0/9 esi source-bmac 00:33:33:33:33:33
user@PE1# set ae0 esi 00:11:11:11:11:11:11:11:11:11
user@PE1# set ae0 esi single-active
user@PE1# set ae0 esi source-bmac 00:11:11:11:11:11

```



NOTE: In this example, the EVPN multihoming is operating in the active/standby mode. To configure active/active EVPN multihoming, include the active-active statement at the [edit interfaces *interface-name* esi] hierarchy level instead of the single-active statement.

6. Configure the interface of Device PE1 that connects to Devices PE2, PE3, and PE4.

```

[edit interfaces]
user@PE1# set ge-2/0/2 gigether-options 802.3ad ae12
user@PE1# set ge-2/0/3 gigether-options 802.3ad ae12
user@PE1# set xe-1/2/3 gigether-options 802.3ad ae13
user@PE1# set xe-1/3/0 gigether-options 802.3ad ae13
user@PE1# set ge-2/1/0 gigether-options 802.3ad ae14
user@PE1# set ge-2/1/5 gigether-options 802.3ad ae14

```

7. Configure the aggregate Ethernet bundle on Device PE1 that connect to Devices PE2, PE3, and PE4.

```

[edit interfaces]
user@PE1# set ae12 flexible-vlan-tagging
user@PE1# set ae12 encapsulation flexible-ethernet-services
user@PE1# set ae12 aggregated-ether-options minimum-links 1
user@PE1# set ae12 unit 0 vlan-id 1200
user@PE1# set ae12 unit 0 family inet address 172.16.0.1/16
user@PE1# set ae12 unit 0 family iso
user@PE1# set ae12 unit 0 family mpls
user@PE1# set ae13 flexible-vlan-tagging
user@PE1# set ae13 encapsulation flexible-ethernet-services
user@PE1# set ae13 aggregated-ether-options minimum-links 1
user@PE1# set ae13 unit 0 vlan-tags outer 1300

```

```

user@PE1# set ae13 unit 0 vlan-tags inner 13
user@PE1# set ae13 unit 0 family inet address 172.17.0.1/16
user@PE1# set ae13 unit 0 family iso
user@PE1# set ae13 unit 0 family mpls
user@PE1# set ae14 aggregated-ether-options minimum-links 1
user@PE1# set ae14 unit 0 family inet address 172.18.0.1/16
user@PE1# set ae14 unit 0 family iso
user@PE1# set ae14 unit 0 family mpls

```

8. Configure the interface of Device PE1 that connects to the host.

```

[edit interfaces]
user@PE1# set xe-1/0/1 flexible-vlan-tagging
user@PE1# set xe-1/0/1 encapsulation flexible-ethernet-services
user@PE1# set xe-1/0/1 unit 0 encapsulation vlan-bridge
user@PE1# set xe-1/0/1 unit 0 vlan-id 10

```

9. Configure the loopback interface of Device PE1.

```

[edit interfaces]
user@PE1# set lo0 unit 0 family inet address 127.0.0.1/8 primary
user@PE1# set lo0 unit 0 family iso

```

10. Configure the customer backbone port (CBP) interfaces on Device PE1.

```

[edit interfaces]
user@PE1# set cbp0 unit 0 family bridge interface-mode trunk
user@PE1# set cbp0 unit 0 family bridge bridge-domain-type bvlan
user@PE1# set cbp0 unit 0 family bridge isid-list all
user@PE1# set cbp0 unit 1 family bridge interface-mode trunk
user@PE1# set cbp0 unit 1 family bridge bridge-domain-type bvlan
user@PE1# set cbp0 unit 1 family bridge isid-list all

```

11. Configure the Provider Instance Port (PIP) on Device PE1.

```

[edit interfaces]
user@PE1# set pip0 unit 0 family bridge interface-mode trunk
user@PE1# set pip0 unit 0 family bridge bridge-domain-type svlan
user@PE1# set pip0 unit 0 family bridge isid-list all-service-groups

```



```

user@PE1# set pip0 unit 1 family bridge interface-mode trunk
user@PE1# set pip0 unit 1 family bridge bridge-domain-type svlan
user@PE1# set pip0 unit 1 family bridge isid-list all-service-groups

```

12. Configure the router ID and autonomous system number for Device PE1.

```

[edit routing-options]
user@PE1# set router-id 127.0.0.1
user@PE1# set autonomous-system 65221

```

13. Configure RSVP on all the interfaces of Device PE1, excluding the management interface.

```

[edit protocols]
user@PE1# set rsvp interface all
user@PE1# set rsvp interface fxp0.0 disable

```

14. Configure MPLS on all the interfaces of Device PE1, excluding the management interface.

```

[edit protocols]
user@PE1# set mpls interface all
user@PE1# set mpls interface fxp0.0 disable

```

15. Configure LSPs from Device PE1 to all other PE devices.

```

[edit protocols]
user@PE1# set mpls label-switched-path pe1tope2 from 127.0.0.1
user@PE1# set mpls label-switched-path pe1tope2 to 127.0.0.2
user@PE1# set mpls label-switched-path pe1tope2 primary direct_to_pe2
user@PE1# set mpls label-switched-path pe1tope3 from 127.0.0.1
user@PE1# set mpls label-switched-path pe1tope3 to 127.0.0.3
user@PE1# set mpls label-switched-path pe1tope3 primary direct_to_pe3
user@PE1# set mpls label-switched-path pe1tope4 from 127.0.0.1
user@PE1# set mpls label-switched-path pe1tope4 to 127.0.0.4
user@PE1# set mpls label-switched-path pe1tope4 primary direct_to_pe4

```

16. Configure MPLS paths from Device PE1 to all other PE devices.

```
[edit protocols]
user@PE1# set mpls path direct_to_pe2 100.12.1.2 strict
user@PE1# set mpls path direct_to_pe3 100.13.1.3 strict
user@PE1# set mpls path direct_to_pe4 100.14.1.4 strict
```

17. Configure an internal BGP session under family EVPN from Device PE1 to all other PE devices.

```
[edit protocols]
user@PE1# set bgp group ibgp type internal
user@PE1# set bgp group ibgp local-address 127.0.0.1
user@PE1# set bgp group ibgp family evpn signaling
user@PE1# set bgp group ibgp neighbor 127.0.0.2
user@PE1# set bgp group ibgp neighbor 127.0.0.3
user@PE1# set bgp group ibgp neighbor 127.0.0.4
```

18. Configure OSPF on all the interfaces of Device of PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set ospf traffic-engineering
user@PE1# set ospf area 0.0.0.0 interface all
user@PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
```

19. Configure a customer routing instance (I-component) on Device PE1 with type virtual switch. Assign the CBP interface, route-distinguisher, and virtual routing and forwarding (VRF) target values to the PBBN routing instance.

```
[edit routing-instances]
user@PE1# set pbbn1 instance-type virtual-switch
user@PE1# set pbbn1 interface cbp0.0
user@PE1# set pbbn1 route-distinguisher 127.0.0.1:1
user@PE1# set pbbn1 vrf-target target:100:1
```

20. Configure PBB-EVPN integration from the customer routing instance. Assign the extended I-SID list and bridge domains to the routing instance.

```
[edit routing-instances]
user@PE1# set pbbn1 protocols evpn pbb-evpn-core
user@PE1# set pbbn1 protocols evpn extended-isid-list 1000
user@PE1# set pbbn1 bridge-domains bda vlan-id 100
user@PE1# set pbbn1 bridge-domains bda isid-list 1000
user@PE1# set pbbn1 bridge-domains bda vlan-id-scope-local
```

21. Configure a provider routing instance on Device PE1 with type virtual switch. Assign the PBP interface and bridge domains to the routing instance.

```
[edit routing-instances]
user@PE1# set pbn1 instance-type virtual-switch
user@PE1# set pbn1 interface pip0.0
user@PE1# set pbn1 bridge-domains bda domain-type bridge
user@PE1# set pbn1 bridge-domains bda vlan-id 10
user@PE1# set pbn1 bridge-domains bda interface ae0.0
user@PE1# set pbn1 bridge-domains bda interface xe-1/0/1.0
user@PE1# set pbn1 bridge-domains bda interface ge-2/0/9.0
user@PE1# set pbn1 bridge-domains bda interface ge-2/0/8.0
```

22. Configure the peer PBBN routing instance in the customer routing instance.

```
[edit routing-instances]
user@PE1# set pbn1 pbb-options peer-instance pbbn1
```

23. Configure the service groups to be supported in the customer routing instance.

```
[edit routing-instances]
user@PE1# set pbn1 service-groups sga service-type elan
user@PE1# set pbn1 service-groups sga pbb-service-options isid 1000 vlan-id-list 10
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `show interfaces`, `routing-options`, `show protocols` 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 chassis
aggregated-devices {
  ethernet {
    device-count 50;
  }
}
network-services enhanced-ip;
```

```
user@PE1# show interfaces
xe-1/0/1 {
  flexible-vlan-tagging;
  encapsulation flexible-ethernet-services;
  unit 0 {
    encapsulation vlan-bridge;
    vlan-id 10;
  }
}
xe-1/2/3 {
  gigether-options {
    802.3ad ae13;
  }
}
xe-1/3/0 {
  gigether-options {
    802.3ad ae13;
  }
}
ge-2/0/2 {
  gigether-options {
    802.3ad ae12;
  }
}
ge-2/0/3 {
  gigether-options {
    802.3ad ae12;
```

```

    }
}
ge-2/0/6 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-2/0/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-2/0/8 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
        00:22:22:22:22:22:22:22:22;
        single-active;
        source-bmac 00:22:22:22:22:22;
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 20;
    }
}
ge-2/0/9 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
        00:33:33:33:33:33:33:33:33;
        single-active;
        source-bmac 00:33:33:33:33:33;
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 30;
    }
}
ge-2/1/0 {
    gigether-options {
        802.3ad ae14;
    }
}

```

```

ge-2/1/5 {
    gigeother-options {
        802.3ad ae14;
    }
}
ae0 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    esi {
        00:11:11:11:11:11:11:11:11:11;
        single-active;
        source-bmac 00:11:11:11:11:11;
    }
    unit 0 {
        encapsulation vlan-bridge;
        vlan-id 10;
    }
}
ae12 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    aggregated-ether-options {
        minimum-links 1;
    }
    unit 0 {
        vlan-id 1200;
        family inet {
            address 172.16.0.1/16;
        }
        family iso;
        family mpls;
    }
}
ae13 {
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    aggregated-ether-options {
        minimum-links 1;
    }
    unit 0 {
        vlan-tags outer 1300 inner 13;
        family inet {
            address 172.17.0.1/16;
        }
    }
}

```

```

    }
    family iso;
    family mpls;
}
}
ae14 {
    aggregated-ether-options {
        minimum-links 1;
    }
    unit 0 {
        family inet {
            address 172.18.0.1/16;
        }
        family iso;
        family mpls;
    }
}
cbp0 {
    unit 0 {
        family bridge {
            interface-mode trunk;
            bridge-domain-type bvlan;
            isid-list all;
        }
    }
    unit 1 {
        family bridge {
            interface-mode trunk;
            bridge-domain-type bvlan;
            isid-list all;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 127.0.0.1/8 {
                primary;
            }
        }
        family iso;
    }
}
}

```

```

pip0 {
    unit 0 {
        family bridge {
            interface-mode trunk;
            bridge-domain-type svlan;
            isid-list all-service-groups;
        }
    }
    unit 1 {
        family bridge {
            interface-mode trunk;
            bridge-domain-type svlan;
            isid-list all-service-groups;
        }
    }
}

```

```

user@PE1# show routing-options
router-id 127.0.0.1;
autonomous-system 65221;

```

```

user@PE1# show protocols
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
mpls {
    label-switched-path pe1tope2 {
        from 127.0.0.1;
        to 127.0.0.2;
        primary direct_to_pe2;
    }
    label-switched-path pe1tope3 {
        from 127.0.0.1;
        to 127.0.0.3;
        primary direct_to_pe3;
    }
    label-switched-path pe1tope4 {

```



```
        from 127.0.0.1;
        to 127.0.0.4;
        primary direct_to_pe4;
    }
    path direct_to_pe2 {
        100.12.1.2 strict;
    }
    path direct_to_pe3 {
        100.13.1.3 strict;
    }
    path direct_to_pe4 {
        100.14.1.4 strict;
    }
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    group ibgp {
        type internal;
        local-address 127.0.0.1;
        family evpn {
            signaling;
        }
        neighbor 127.0.0.2;
        neighbor 127.0.0.3;
        neighbor 127.0.0.4;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

```

    }
}

```

```
user@PE1# show routing-instances
```

```

pbbn1 {
    instance-type virtual-switch;
    interface cbp0.0;
    route-distinguisher 127.0.0.1:1;
    vrf-target target:100:1;
    protocols {
        evpn {
            pbb-evpn-core;
            extended-isid-list 1000;
        }
    }
    bridge-domains {
        bda {
            vlan-id 100;
            isid-list 1000;
            vlan-id-scope-local;
        }
    }
}

pbn1 {
    instance-type virtual-switch;
    interface pip0.0;
    bridge-domains {
        bda {
            domain-type bridge;
            vlan-id 10;
            interface ae0.0;
            interface xe-1/0/1.0;
            interface ge-2/0/9.0;
            interface ge-2/0/8.0;
        }
    }
    pbb-options {
        peer-instance pbbn1;
    }
    service-groups {
        sga {

```

```
        service-type elan;
        pbb-service-options {
            isid 1000 vlan-id-list 10;
        }
    }
}
}
```

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

Verification

IN THIS SECTION

[Verifying BGP Peering Status | 2002](#)

[Verify MAC Table Entries | 2003](#)

[Verifying the EVPN Database | 2004](#)

[Verifying EVPN Routing Instances | 2004](#)

Confirm that the configuration is working properly.

Verifying BGP Peering Status

Purpose

Verify that the BGP session is established between the PE devices.

Action

From operational mode, run the `show bgp summary` command.

```
user@PE1> show bgp summary
Groups: 1 Peers: 3 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State   Pending
bgp.evpn.0
              13         13         0           0         0         0
Peer          AS      InPkt   OutPkt   OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
```

```

127.0.0.2      65221      10      8      0      0      42 Establ
  bgp.evpn.0: 6/6/6/0
  pbbn1.evpn.0: 3/3/3/0
  __default_evpn__.evpn.0: 3/3/3/0
127.0.0.3      65221      8      8      0      0      40 Establ
  bgp.evpn.0: 4/4/4/0
  pbbn1.evpn.0: 3/3/3/0
  __default_evpn__.evpn.0: 1/1/1/0
127.0.0.4      65221      9      11     0      0      1:04 Establ
  bgp.evpn.0: 3/3/3/0
  pbbn1.evpn.0: 3/3/3/0
  __default_evpn__.evpn.0: 0/0/0/0

```

Meaning

A BGP session is established between the PE devices.

Verify MAC Table Entries

Purpose

Verify the number of rbeb interfaces learned in the MAC table on all PEs.

Action

From operational mode, run the `show bgp summary` command.

```

user@PE1> show bridge mac-table count instance pbn1 bridge-domain bda
10 MAC address learned in routing instance pbn1 bridge domain bda

```

MAC address count per interface within routing instance:

Logical interface	MAC count
ae0.0:10	0
ge-2/0/8.0:10	10
ge-2/0/9.0:10	0
rbeb.32772	0
rbeb.32771	0
xe-1/0/0.0:10	0

MAC address count per learn VLAN within routing instance:

Learn VLAN ID	MAC count
10	10

Meaning

For VLAN ID 10, 10 MAC addresses have been learned in the MAC table of Device PE1.

Verifying the EVPN Database

Purpose

Verify the EVPN database information on Device PE4.

Action

From operational mode, run the `show evpn database` command.

```
user@PE1> show evpn database
Instance: pbbn1
```

VLAN	DomainId	MAC address	Active source	Timestamp	IP address
1000		00:11:11:11:11:11	127.0.0.1	Jan 26 12:09:29	
1000		00:11:11:11:11:12	127.0.0.2	Jan 26 12:09:38	
1000		00:1d:b5:a2:47:b0	127.0.0.1	Jan 26 12:09:10	
1000		00:22:22:22:22:22	127.0.0.1	Jan 26 12:09:29	
1000		00:23:9c:5e:a7:b0	Local	Jan 26 12:08:02	
1000		00:23:9c:f0:f9:b0	127.0.0.3	Jan 26 12:09:30	
1000		00:33:33:33:33:33	127.0.0.1	Jan 26 12:09:29	
1000		00:44:44:44:44:44	127.0.0.3	Jan 26 12:09:34	
1000		00:44:44:44:44:45	Local	Jan 26 12:09:28	
1000		80:71:1f:c1:ed:b0	127.0.0.2	Jan 26 12:09:31	

Verifying EVPN Routing Instances

Purpose

Verify the EVPN routing instance information on all the PE devices.

Action

From operational mode, run the `show evpn routing-instance` command.

```

user@PE1> show evpn routing-instance
Instance: pbbn1
  Route Distinguisher: 127.0.0.1:1
  Per-instance MAC route label: 300080
  Per-instance multicast route label: 300096
PBB EVPN Core enabled
  Control word enabled
  MAC database status
    Local Remote
  MAC advertisements:          4      6
  MAC+IP advertisements:      0      0
  Default gateway MAC advertisements: 0      0
  Number of local interfaces: 4 (4 up)
    Interface name  ESI                               Mode      Status    AC-Role
  ae0.0            00:11:11:11:11:11:11:11:11:11:11 single-active Up        Root
  cbp0.0           00:00:00:00:00:00:00:00:00:00 single-homed Up        Root
  ge-2/0/8.0       00:22:22:22:22:22:22:22:22:22 single-active Up        Root
  ge-2/0/9.0       00:33:33:33:33:33:33:33:33:33 single-active Up        Root
  Number of IRB interfaces: 0 (0 up)
  Number of bridge domains: 1
    VLAN Domain ID  Intfs / up  IRB intf  Mode      MAC sync  IM route label  SG
  sync  IM core nexthop
    1000          0    0          Extended  Enabled    300096
  Disabled
  Number of Bundle bridge domains: 0
  Number of neighbors: 3
    Address      MAC    MAC+IP    AD    IM    ES Leaf-label
  127.0.0.2     2      0        0     1     0
  127.0.0.3     2      0        0     1     0
  127.0.0.4     2      0        0     1     0
  Number of ethernet segments: 3
  ESI: 00:11:11:11:11:11:11:11:11:11
    Status: Resolved by IFL ae0.0
    Local interface: ae0.0, Status: Up/Blocking
    Designated forwarder: 127.0.0.2
    Backup forwarder: 127.0.0.1
    Backup forwarder: 127.0.0.3
    Last designated forwarder update: Jan 26 12:43:42
    Minimum ISID: 1000

```

```

ESI: 00:22:22:22:22:22:22:22:22
  Status: Resolved by IFL ge-2/0/8.0
  Local interface: ge-2/0/8.0, Status: Up/Forwarding
  Designated forwarder: 127.0.0.1
  Backup forwarder: 127.0.0.2
  Last designated forwarder update: Jan 26 12:43:42
  Minimum ISID: 1000
ESI: 00:33:33:33:33:33:33:33:33
  Status: Resolved by IFL ge-2/0/9.0
  Local interface: ge-2/0/9.0, Status: Up/Forwarding
  Designated forwarder: 127.0.0.1
  Backup forwarder: 127.0.0.2
  Last designated forwarder update: Jan 26 12:43:42
  Minimum ISID: 1000

```

```

user@PE2> show evpn routing-instance
Instance: pbbn1
  Route Distinguisher: 127.0.0.2:1
  Per-instance MAC route label: 338721
  Per-instance multicast route label: 338737
  PBB EVPN Core enabled
  Control word enabled
  MAC database status
    Local Remote
  MAC advertisements:          2      8
  MAC+IP advertisements:      0      0
  Default gateway MAC advertisements: 0      0
  Number of local interfaces: 4 (4 up)
    Interface name  ESI                               Mode          Status    AC-Role
  ae0.0            00:11:11:11:11:11:11:11:11 single-active  Up        Root
  cbp0.0           00:00:00:00:00:00:00:00:00 single-homed   Up        Root
  ge-1/3/5.0       00:22:22:22:22:22:22:22:22 single-active  Up        Root
  ge-1/3/6.0       00:33:33:33:33:33:33:33:33 single-active  Up        Root
  Number of IRB interfaces: 0 (0 up)
  Number of bridge domains: 1
    VLAN  Domain ID  Intfs / up  IRB intf  Mode          MAC sync  IM route label  SG
  sync  IM core nexthop
    1000          0    0          Extended  Enabled    338737
  Disabled
  Number of Bundle bridge domains: 0
  Number of neighbors: 3
    Address          MAC    MAC+IP    AD    IM    ES Leaf-label

```

127.0.0.1	4	0	0	1	0
127.0.0.3	2	0	0	1	0
127.0.0.4	2	0	0	1	0

Number of ethernet segments: 3

ESI: 00:11:11:11:11:11:11:11:11

Status: Resolved by IFL ae0.0

Local interface: ae0.0, Status: **Up/Forwarding**

Designated forwarder: 127.0.0.2

Backup forwarder: 127.0.0.1

Backup forwarder: 127.0.0.3

Last designated forwarder update: Jan 26 12:09:37

Minimum ISID: 1000

ESI: 00:22:22:22:22:22:22:22:22

Status: Resolved by IFL ge-1/3/5.0

Local interface: ge-1/3/5.0, Status: **Up/Blocking**

Designated forwarder: 127.0.0.1

Backup forwarder: 127.0.0.2

Last designated forwarder update: Jan 26 12:09:33

Minimum ISID: 1000

ESI: 00:33:33:33:33:33:33:33:33

Status: Resolved by IFL ge-1/3/6.0

Local interface: ge-1/3/6.0, Status: **Up/Blocking**

Designated forwarder: 127.0.0.1

Backup forwarder: 127.0.0.2

Last designated forwarder update: Jan 26 12:09:33

Minimum ISID: 1000

user@PE3> show evpn routing-instance

Instance: pbbn1

Route Distinguisher: 127.0.0.3:1

Per-instance MAC route label: 338833

Per-instance multicast route label: 338849

PBB EVPN Core enabled

Control word enabled

MAC database status

	Local	Remote
MAC advertisements:	2	8
MAC+IP advertisements:	0	0
Default gateway MAC advertisements:	0	0

Number of local interfaces: 3 (3 up)

Interface name	ESI	Mode	Status	AC-Role
ae0.0	00:11:11:11:11:11:11:11:11	single-active	Up	Root


```

    cbp0.0      00:00:00:00:00:00:00:00:00 single-homed Up      Root
    ge-1/0/8.0  00:44:44:44:44:44:44:44:44 single-active Up      Root
Number of IRB interfaces: 0 (0 up)
Number of bridge domains: 1
  VLAN Domain ID  Intfs / up  IRB intf  Mode          MAC sync  IM route label  SG
sync  IM core nexthop
      1000        0    0              Extended      Enabled    338849
Disabled
Number of Bundle bridge domains: 0
Number of neighbors: 3
  Address          MAC      MAC+IP      AD      IM      ES Leaf-label
  127.0.0.1        4        0          0        1        0
  127.0.0.2        2        0          0        1        0
  127.0.0.4        2        0          0        1        0
Number of ethernet segments: 2
ESI: 00:11:11:11:11:11:11:11:11
  Status: Resolved by IFL ae0.0
  Local interface: ae0.0, Status: Up/Blocking
  Designated forwarder: 127.0.0.2
  Backup forwarder: 127.0.0.1
  Backup forwarder: 127.0.0.3
  Last designated forwarder update: Jan 26 12:09:46
  Minimum ISID: 1000
ESI: 00:44:44:44:44:44:44:44:44
  Status: Resolved by IFL ge-1/0/8.0
  Local interface: ge-1/0/8.0, Status: Up/Forwarding
  Designated forwarder: 127.0.0.3
  Backup forwarder: 127.0.0.4
  Last designated forwarder update: Jan 26 12:09:31
  Minimum ISID: 1000

```

```

user@PE4> show evpn routing-instance
Instance: pbbn1
Route Distinguisher: 127.0.0.4:1
Per-instance MAC route label: 301520
Per-instance multicast route label: 301536
PBB EVPN Core enabled
Control word enabled
MAC database status
MAC advertisements:
MAC+IP advertisements:

```

	Local	Remote
MAC advertisements:	2	8
MAC+IP advertisements:	0	0

```

Default gateway MAC advertisements:      0      0
Number of local interfaces: 2 (2 up)
Interface name  ESI                               Mode      Status    AC-Role
cbp0.0          00:00:00:00:00:00:00:00:00:00 single-homed Up         Root
ge-0/2/1.0      00:44:44:44:44:44:44:44:44:44 single-active Up         Root
Number of IRB interfaces: 0 (0 up)
Number of bridge domains: 1
VLAN  Domain ID  Intfs / up  IRB intf  Mode      MAC sync  IM route label  SG
sync  IM core nexthop
      1000        0    0                Extended    Enabled    301536
Disabled
Number of Bundle bridge domains: 0
Number of neighbors: 3
Address          MAC    MAC+IP      AD      IM      ES Leaf-label
127.0.0.1        4      0          0        1        0
127.0.0.2        2      0          0        1        0
127.0.0.3        2      0          0        1        0
Number of ethernet segments: 1
ESI: 00:44:44:44:44:44:44:44:44:44
Status: Resolved by IFL ge-0/2/1.0
Local interface: ge-0/2/1.0, Status: Up/Blocking
Designated forwarder: 127.0.0.3
Backup forwarder: 127.0.0.4
Last designated forwarder update: Jan 26 12:43:52
Minimum ISID: 1000

```

Meaning

The command output displays PBB-EVPN integration, where EVPN is in the active/standby multihoming mode. If EVPN multihoming was configured in the active/active mode, the status of all the ESIs would be Up/Forwarding. In the active/standby multihoming mode, one ESI is in the Up/Forwarding state and all other ESIs remain in the Up/Blocking state.

RELATED DOCUMENTATION

[Provider Backbone Bridging \(PBB\) and EVPN Integration Overview | 1897](#)

[Example: Configuring PBB with Single-Homed EVPN | 1931](#)

pbb-evpn-core

Configuring MAC Pinning for PBB-EVPNs

IN THIS CHAPTER

- [PBB-EVPN MAC Pinning Overview | 2010](#)
- [Configuring PBB-EVPN MAC Pinning | 2011](#)

PBB-EVPN MAC Pinning Overview

In current Junos OS releases, 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.

The MAC pinning feature is used to avoid loops in a network and is also used for MAC security restriction by avoiding MAC move on duplicate MAC detection. When MAC pinning is enabled, the dynamically learned MAC addresses are not allowed to move to any other interface in a bridge domain until it is aged out and traffic received with the same source MAC address on other bridge interfaces are discarded. This feature is an advantage over blocking of the complete interface on duplicate MAC detection or loop, as MAC pinning works at the MAC label. This feature is local to a provider edge (PE) device and does not require any interoperability.

PBB has I-component and B-Component, where I-component (customer routing instance) is responsible for mapping the CE port traffic to the instance source ID (I-SID), and the B-component learns and forwards traffic on the backbone port. Traffic received from the MPLS core or from the PBB port is classified and based on the I-SID and PBB MAC, and gets mapped to the correct I-component. Remote customer MAC addresses are learned over remote backbone edge port (BEB) interface in the I-component bridge domain. This interface is created dynamically on PBB neighbor detection. MAC addresses learned over the remote BEB interface in I-component are pinned when MAC pinning is enabled for PBB-EVPN.

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 bridge domain over CE interfaces, as well as PBB-MPLS core interfaces are pinned.

When configuring the PBB-EVPN MAC pinning feature, take the following into consideration:

- PBB-EVPN MAC pinning is supported on MX Series routers with MPC and MIC interfaces only.
- PBB-EVPN MAC pinning is supported on Ethernet Layer 2 bridge interfaces only.
- When there is a MAC move between the I-component and an access interface, the MAC address is learned locally over the PBB-EVPN MPLS core over a remote BEB interface in the I-component bridge domain. The MAC moves between the CE or core interfaces for this MAC is not allowed.
- In MAC pinning for PBB with EVPN active-active and single-active multihoming, MAC pinning must be enabled or disabled on all the multihomed PE devices in the broadcast domain. This is because MAC pinning at a multihomed PE device is local to the PE, and it is possible that a MAC address that is pinned towards a multihomed CE device and PE device is also pinned toward a single-homed customer site or toward any other Ethernet segment identifier (ESI) at another multihomed PE device.
- A next hop bridge domain is created in PBB-EVPN I-component bridge domain toward the B-component when there is an unresolved source MAC notification when the first remote MAC address is received. As a result, the first MAC address learned over PBB back bone core interface can be delayed on pinning, and may result moving to other single-homed or ESI interface if the same MAC traffic is received.
- Static MAC addresses are given preference over dynamic pin MACs.
- MAC pinning is enabled for all neighbors of a PBB routing instance and cannot be enabled for a specific neighbor.
- PBB-EVPN MAC pin discard notification is not generated for a remote BEB interface when traffic is discarded due to MAC pinning until a MAC is learned locally over the remote BEB interface.

RELATED DOCUMENTATION

Understanding MAC Pinning

[Configuring PBB-EVPN MAC Pinning | 2011](#)

mac-pinning

pbb-evpn-core

Configuring PBB-EVPN MAC Pinning

In the Junos OS releases that support MAC pinning for PBB-EVPN, the MAC pinning feature is enabled for provider backbone bridging (PBB) and for Ethernet VPN (EVPN) integration, including for customer edge (CE) interfaces and for EVPN over PBB core in both all-active or single-active mode. See the [MAC](#)

[pinning support for PBB-EVPN](#) feature in [Feature Explorer](#) for platform and release support information for the MAC pinning for PBB-EVPN feature.

When MAC pinning is enabled, the dynamically learned MAC addresses are not allowed to move to any other interface in a bridge domain until it is aged out and traffic received with the same source MAC address on other bridge interfaces are discarded. This feature is an advantage over blocking of the complete interface on duplicate MAC detection or loop detection, as MAC pinning works at the MAC label. This feature is local to a provider edge (PE) device and does not require any interoperability.

Before you begin:

- Configure the device interfaces, including the customer backbone port (CBP) interface, the provider instance port (PIP) interfaces, and the loopback interface. Assign the bridge family to the interfaces.
- Assign the router ID and autonomous system ID to the device.
- Configure an internal BGP group with EVPN signaling.
- Enable the following protocols on the device:
 - MPLS
 - LDP
 - OSPF

To enable MAC pinning on PBB-EVPN:

1. Configure the B-component routing instance.

Assign the virtual switch instance type and the CBP interface to it. Configure other routing instance attributes like route distinguisher and virtual routing and forwarding (VRF) target to the routing instance.



NOTE: Configure B-component routing instances for other CBP interface units on the device, and assign different VLAN IDs and I-SID lists for the different interface units.

```
[edit routing-instances]
user@R1# set pbbn instance-type virtual-switch
user@R1# set pbbn interface cbp-interface
user@R1# set pbbn route-distinguisher route-distinguisher-value
user@R1# set pbbn vrf-target vrf-target
```

2. Enable PBB- EVPN integration for the B-component routing instance.

```
[edit routing-instances]
user@R1# set pbbn protocols evpn pbb-evpn-core
```

3. Enable MAC pinning for the B-component routing instance.

```
[edit routing-instances]
user@R1# set pbbn protocols evpn mac-pinning
```

4. Assign instance source IDs (I-SID) list to the B-component routing instance.

```
[edit routing-instances]
user@R1# set pbbn protocols evpn extended-isid-list extended-isid-list
```

5. Configure a bridge domain for the B-component routing instance and assign a VLAN and and I-SID list to the bridge domain.

```
[edit routing-instances]
user@R1# set pbbn bridge-domains bridge-domain vlan-id vlan-id
user@R1# set pbbn bridge-domains bridge-domain isid-list isid-list
```

6. Configure the I-component routing instance.
Assign the virtual switch instance type and the PIP interface to it.



NOTE: Configure I-component routing instances for other PIP interface units on the device, and assign different bridge domains, VLAN IDs and I-SID lists for the different interface units.

```
[edit routing-instances]
user@R1# set pbn instance-type virtual-switch
user@R1# set pbn interface pip-interface
```

7. Configure bridge domain for the I-component routing instance and assign interfaces and VLANs to the bridge domain.

```
[edit routing-instances]
user@R1# set pbn bridge-domains bridge-domain domain-type bridge
```

```
user@R1# set pbn bridge-domains bridge-domain vlan-id vlan-id
user@R1# set pbn bridge-domains bridge-domain interface interface-name
```

8. Enable MAC pinning for the interface in the I-component routing instance.

```
[edit routing-instances]
user@R1# set pbn bridge-domains bridge-domain bridge-options interface interface-name mac-
pinning
```

9. Configure peering between the B-component and the I-component routing instances.

```
[edit routing-instances]
user@R1# set pbn bridge-domains bridge-domain pbb-options peer-instance pbbn
```

10. Configure PBB service group and assign I-SID and VLAN ID list.

```
[edit routing-instances]
user@R1# set pbn service-groups service-group pbb-service-options isid isid vlan-id-list
valn-id-list
```

RELATED DOCUMENTATION

Understanding MAC Pinning

[PBB-EVPN MAC Pinning Overview | 2010](#)

mac-pinning

pbb-evpn-core

9

PART

EVPN Standards

- [Supported EVPN Standards | 2016](#)
-

Supported EVPN Standards

IN THIS CHAPTER

- [Supported EVPN Standards | 2016](#)

Supported EVPN Standards

RFCs and Internet drafts that define standards for EVPNs:

- RFC 3704, *Ingress Filtering for Multihomed Networks*
- RFC 4364, *BGP/MPLS IP Virtual Private Networks (VPNs)*
- RFC 4761, *Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling*
- RFC 7209, *Requirements for Ethernet VPN (EVPN)*
- RFC 7432, *BGP MPLS-Based Ethernet VPN*
- RFC 7623, *Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)*
- RFC 8214, *Virtual Private Wire Service Support in Ethernet VPN*
- RFC 8317, *Ethernet-Tree (E-Tree) Support in Ethernet VPN (EVPN) and Provider Backbone Bridging EVPN (PBB-EVPN)*
- RFC 8365, *A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)*
- RFC 8402, *Segment Routing Architecture*
- RFC 8560, *Seamless Integration of Ethernet VPN (EVPN) with Virtual Private LAN Service (VPLS) and Their Provider Backbone Bridge (PBB) Equivalents*
- RFC 8584, *Framework for Ethernet VPN Designated Forwarder Election Extensibility*
- RFC 8660, *Segment Routing with MPLS data plane*
- RFC 8667, *IS-IS Extensions for Segment Routing*

- RFC 9047, *Propagation of ARP/ND Flags in an Ethernet Virtual Private Network (EVPN)*
- RFC 9062, *EVPN Operations, Administration and Maintenance Requirements and Framework*
- RFC 9135, *Integrated Routing and Bridging in Ethernet VPN (EVPN)*
- RFC 9136, *IP Prefix Advertisement in Ethernet VPN (EVPN)*
- RFC 9161, *Operational Aspects of Proxy-ARP/ND in Ethernet Virtual Private Networks*
- RFC 9251, *Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Proxies for Ethernet VPN (EVPN)*
- RFC 9574, *Optimized Ingress Replication Solution for Ethernet VPN (EVPN)*
- RFC 9625, *EVPN Optimized Inter-Subnet Multicast (OISM) Forwarding*
- RFC 9744, *EVPN VPWS Flexible Cross-Connect Service*
- RFC 9784, *Virtual Ethernet Segments for EVPN and Provider Backbone Bridge EVPN*
- RFC 9785, *Preference-based EVPN Designated Forwarder (DF) Election*
- RFC 9786, *EVPN Port-Active Redundancy Mode*
- Internet draft draft-ietf-bess-evpn-yang, *Yang Data Model for EVPN*
- Internet draft draft-wsv-bess-extended-evpn-optimized-ir, *Extended Procedures for EVPN Optimized Ingress Replication*

RELATED DOCUMENTATION

[EVPN Overview | 1429](#)

[Accessing Standards Documents on the Internet](#)

10

PART

VXLAN-Only Features

- Flexible VXLAN Tunnels | **2019**
 - Static VXLAN | **2026**
-

Flexible VXLAN Tunnels

IN THIS CHAPTER

- [Understanding Programmable Flexible VXLAN Tunnels | 2019](#)

Understanding Programmable Flexible VXLAN Tunnels

IN THIS SECTION

- [Benefits of Programmable Flexible Tunnels | 2020](#)
- [Programmable Flexible Tunnels, Flexible Routes, and Tunnel Profiles | 2021](#)
- [Routes and Routing Tables | 2022](#)
- [Flexible Tunnel Traffic Flows | 2023](#)
- [Preparing the Gateway Devices | 2023](#)
- [Understanding Flexible Tunnel Behavior | 2024](#)
- [Flexible Tunnel Limitations | 2024](#)

Starting in Junos OS Release 19.1R1, we support flexible tunnels in the following data center environment:

- Implements one or more controllers.
- Uses Virtual Extensible LAN (VXLAN) as the overlay encapsulation protocol.



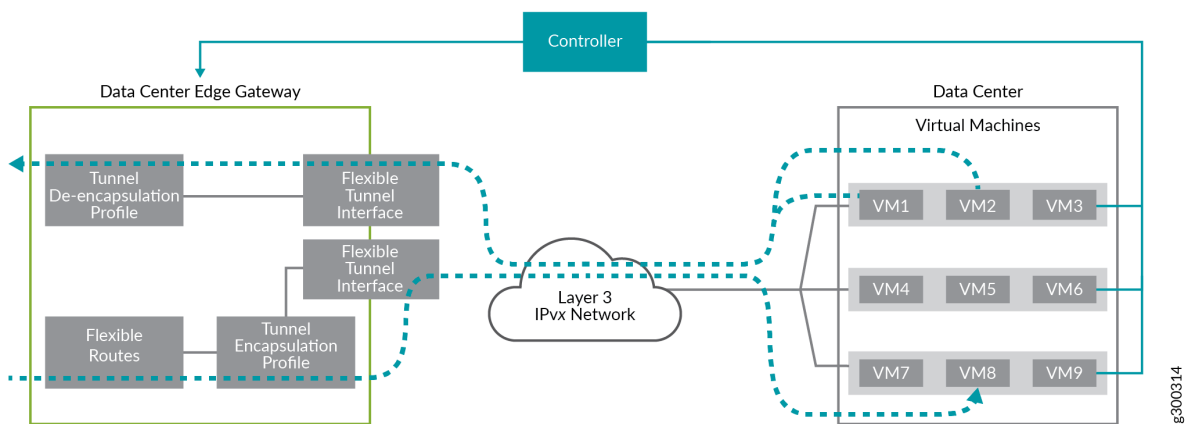
NOTE: The flexible tunnel feature supports IPv4 and IPv6 VXLAN encapsulation and de-encapsulation.

- Controllers in your data center environment and the JET APIs enable you to program a large number of flexible tunnels into the gateway devices. This new method enables edge gateways to communicate with a large number of hosts in a data center.
- The tunnel encapsulation and de-encapsulation profiles for a particular flexible tunnel are typically separate and do not need to be symmetrical. A static route for a particular data center host is mapped to an encapsulation profile, and traffic destined for that host is encapsulated in a distinct tunnel. However, de-encapsulation is not associated with a particular static route. As a result, traffic from one or more data center hosts that match attributes in a de-encapsulation profile can be aggregated into a single tunnel. This mechanism results in fewer de-encapsulation profiles and more efficient use of the existing profiles and flexible tunnels.

Programmable Flexible Tunnels, Flexible Routes, and Tunnel Profiles

Figure 213 on page 2021 shows a sample topology in which we support flexible tunnels. In this topology, a data center edge gateway interfaces with virtual machines (VMs) in a data center in which VXLAN is used. When tunneling Layer 3 traffic over the intervening IPv4 or IPv6 network, the gateway device and the hypervisors or other specialized devices that manage the VMs also function as virtual tunnel endpoints (VTEPs). The VTEPs encapsulate the packets with a VXLAN header, remove, or de-encapsulate, the header from the packets, then forward the packets.

Figure 213: Flexible Tunnel Components and Traffic Flows



To support a flexible tunnel, the following components are required:

- A flexible tunnel interface, which is a Layer 3 logical interface that supports both IPv4 and IPv6 families, on the gateway device. To configure a flexible tunnel interface, you must use the `tunnel` and `family` stanzas at the `[edit interfaces unit logical-unit-number]` hierarchy level. For more information, see [Flexible Tunnel Interfaces Overview](#).
- A flexible route, which is comprised of the following:
 - A static route that specifies, among other attributes, a tunnel encapsulation profile.
 - A tunnel encapsulation profile that specifies, among other attributes, a flexible tunnel interface.
- A tunnel de-encapsulation profile, which is typically separate from the encapsulation profile and specifies a subset of the attributes required for the encapsulation profile.

You can program a static route and map an encapsulation profile to the route using the `rib.service.proto` API file and fully define the encapsulation profile using the `flexible_tunnel_profile.proto` API file.

You can fully define de-encapsulation profiles using the `flexible_tunnel_profile.proto` API file and use the `flexible_tunnel_service.proto` API file to perform the bulk addition, modification, and deletion of

these profiles. The **flexible_tunnel_service.proto** API file also includes parameters that enable you to view a specific de-encapsulation profile.

For complete information about the JET API files, see the [JET API Guide](#).

To display information about tunnel encapsulation and de-encapsulation profiles on the Juniper Networks gateway devices, you can use the *show route detail* and *show route extensive* commands. To display information about de-encapsulation profiles, you can use the *show flexible-tunnels profiles* command.

Routes and Routing Tables

The routes associated with tunnel encapsulation and de-encapsulation profiles have different purposes, and are therefore, stored in different routing tables on the gateway device. [Table 99 on page 2022](#) provides a summary of the tunnel profiles, the routes associated with the profiles, and routing tables.

Table 99: Summary of Tunnel Profiles, Routes, and Routing Tables

Tunnel Profiles	Routes	Route Purpose	Routing Tables	Notes
Tunnel encapsulation profile	Flexible route (static route and a mapped tunnel encapsulation profile)	Used to forward traffic to hosts in the data center.	Routing information base (RIB) or routing table determined by the API configuration.	–
Tunnel de-encapsulation profile	Automatically generated internal route	Used to associate the de-encapsulation profile to a route so that it can be downloaded to an internal routing table.	Internal routing table named __flexible_tunnel_profiles__.inet.0	Upon receipt of an encapsulated packet from a data center host, this routing table provides a means of implementing a lookup of de-encapsulation profiles and matching a particular packet with a de-encapsulation profile.

If performing detailed troubleshooting, it is helpful to know the mapping between a de-encapsulation profile and an internal route. The `show flexible-tunnels profiles` command displays the internal route in the Route prefix field.

Flexible Tunnel Traffic Flows

The flexible tunnel feature supports the following general IPv4 and IPv6 traffic flows:

- When the gateway device receives a packet with a destination address that matches a flexible route in the routing table, the device functions as a source VTEP and takes the following action:
 - Encapsulates the packet according to parameters specified in the encapsulation profile. Encapsulation parameters include but are not limited to encapsulation type, source prefix, destination address, flexible tunnel interface, and VXLAN network identifier (VNI).
 - Forwards the packet toward the destination VM through the flexible tunnel interface specified in the encapsulation profile.
 - Implements the output features—for example, statistics, sampling, mirroring, and filters—configured on the flexible tunnel interface.
- When the gateway device receives an encapsulated packet with content that matches a de-encapsulation profile, the device functions as a destination VTEP and takes the following actions:
 - De-encapsulates the packet.
 - Implements the input features configured on the flexible tunnel interface.
 - Forwards the packet toward its destination through the flexible tunnel interface specified in the de-encapsulation profile.

Preparing the Gateway Devices

Before programming flexible tunnels using the JET APIs, you must enable gRPC Remote Procedure Calls (gRPC) on the gateway devices using the following command:

```
set system services extension-service request-response grpc ssl
```

After enabling gRPC, the gateway device is ready to receive route and tunnel service requests over a secure connection from JET applications.

For complete information about preparing the gateway devices to interact with JET APIs, see the [JET API Guide](#).

Understanding Flexible Tunnel Behavior

Keep the following in mind about the flexible tunnel feature:

- If a static route is mapped to a flexible tunnel interface that is not yet configured, the route remains inactive until the interface is configured.
- If one or more flexible routes are mapped to a flexible tunnel interface that was deleted, the flexible routes become inactive and are removed from the Packet Forwarding Engine.
- If a flexible tunnel interface goes down, is not yet configured, or is offline, traffic to be encapsulated or de-encapsulated is discarded gracefully.
- Each static route must be mapped to one tunnel encapsulation profile. In other words, there should be a one-to-one correspondence between a static route and an encapsulation profile. We do not support the mapping of two or more static routes to the same encapsulation profile.
- If adding or changing a tunnel de-encapsulation profile using the **flexible_tunnel_service.proto** API file and a conflict with another de-encapsulation profile arises, the operation fails, and an error message is provided. A conflict between two de-encapsulation profiles occurs when one or more values of the following parameters are the same:
 - UDP destination port
 - Source prefix
 - Source prefix length
 - VNI

Flexible Tunnel Limitations

The flexible tunnel feature has the following limitations:

- We do not support source lookup features, including but not limited to reverse-path-forwarding (RPF) and MAC address validation, on routing tables in which flexible routes are stored.
- Each flexible route has a tunnel encapsulation profile as a next hop instead of an IP address. As a result, we do not support next-hop features, including but not limited to load balancing, across encapsulation profiles.
- Flexible tunnels do not have control plane functionality. That is, protocols and other control plane features cannot run over these tunnels. Instead, the forwarding plane handles the flexible tunnels.

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
19.1R1	Starting in Junos OS Release 19.1R1, we support flexible tunnels in the following data center environment:

Static VXLAN

IN THIS CHAPTER

- [Static VXLAN | 2026](#)

Static VXLAN

IN THIS SECTION

- [Understanding Static VXLAN | 2026](#)
- [Q-in-Q VLAN Tunnels in a Spine-and-Leaf Network with Static VXLAN | 2031](#)
- [Configure Static VXLAN at the Global Level | 2033](#)
- [Configure Static VXLAN at the VLAN or Bridge Domain Level on L2 VXLAN Gateway Devices | 2036](#)

Juniper Networks supports the static Virtual Extensible LAN (VXLAN) feature in a small multichassis link aggregation group (MC-LAG) network and in small networks on Layer 2 (L2) VXLAN gateway devices.

Understanding Static VXLAN

IN THIS SECTION

- [Benefits of Static VXLAN | 2027](#)
- [How Static VXLAN Works | 2027](#)
- [Small MC-LAG Network Static VXLAN Use Case | 2028](#)
- [VLAN or Bridge Domain Level Static VXLAN Use Case | 2030](#)

Static Virtual Extensible LAN (VXLAN), also known as unicast VXLAN, enables you to statically configure source and destination virtual tunnel endpoints (VTEPs) for a particular traffic flow. A source VTEP encapsulates and a destination VTEP de-encapsulates L2 packets with a VXLAN header, thereby tunneling the packets through an underlying Layer 3 IP network.



NOTE: Although this feature is also known as unicast VXLAN, static VXLAN tunnels can support both unicast and broadcast, unknown unicast, and multicast (BUM) traffic. We support BUM traffic replication and forwarding using ingress replication.

Starting in Junos OS Release 14.1X53-D40, Juniper Networks supports static VXLAN on devices that act as L2 VXLAN gateway devices. Here we show sample environments with a small MC-LAG network or leaf in a classic spine and leaf fabric.

Benefits of Static VXLAN

- Instead of using an Ethernet VPN (EVPN) control plane to learn the MAC addresses of hosts, static VXLAN uses a flooding-and-learning technique in the VXLAN data plane. Therefore, using static VXLAN reduces complexity in the control plane.
- For a small MC-LAG network, EVPN might be overly complex to configure and maintain. Static VXLAN provides the benefits of VXLAN and is relatively easy to design and configure.

How Static VXLAN Works

To enable static VXLAN on a Juniper Networks device that functions as a VTEP, you must configure:

- A list that includes one or more remote VTEPs with which the local VTEP can form a VXLAN tunnel.
- Ingress node replication.
- The VTEP's loopback interface (lo0):
 - Configure an anycast IP address as the primary interface.
 - Specify this interface as the source interface for a VXLAN tunnel.

When a VTEP receives a BUM packet, the VTEP uses ingress node replication to replicate and flood the packet to the statically defined remote VTEPs on your list. The remote VTEPs in turn flood the packet to the hosts in each VXLAN of which the VTEPs are aware.

The VTEPs learn the MAC addresses of:

- Remote hosts from the VTEPs on the remote VTEP list.
- Local hosts from the local access interfaces

Upon learning a MAC address of a host, the device adds the MAC address to the Ethernet switching table.

You can view information about remote static VXLAN VTEP interfaces using the following CLI show commands:

- `show ethernet-switching vxlan-tunnel-end-point remote`—Shows the remote VTEPs on the device, including their IP addresses and VTEP interface names. See the **RVTEP-IP** and **Interface** output fields.
- `show interfaces vtep-interface-name` —Displays status of a VTEP interface. The VTEP interfaces you configure as static VXLAN VTEPs display the value **Configured-remote** in the **VXLAN Endpoint type** output field of this command.

For example, with static VXLAN remote VTEPs configured for VLANs 200 and 300 in an L2 routing instance named `ri1`:

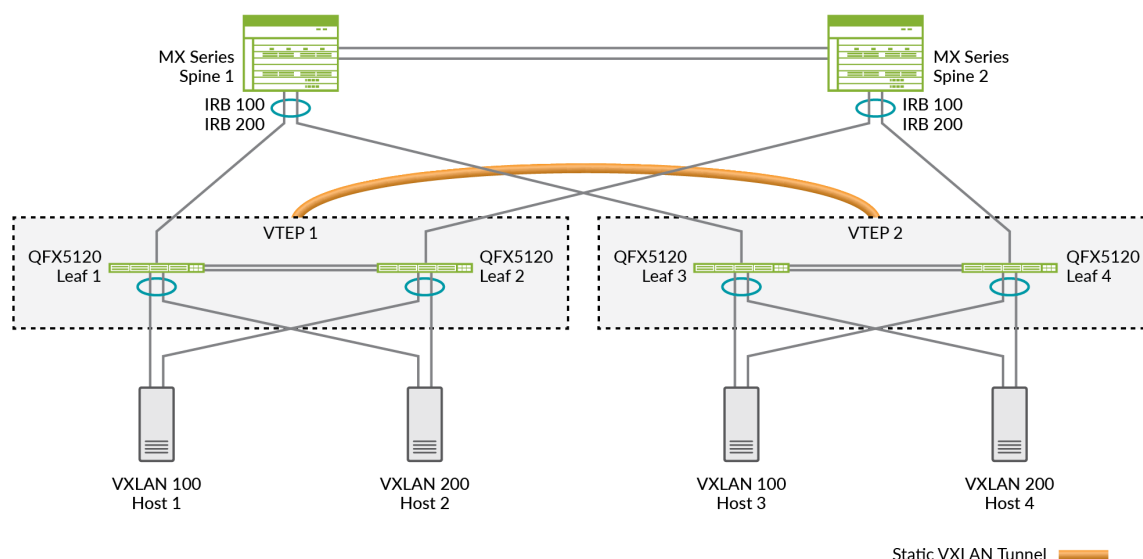
```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point remote
Logical System Name      Id  SVTEP-IP      IFL  L3-Idx  SVTEP-Mode  ELP-SVTEP-IP
<default>                0   192.168.1.1   lo0.0  0
RVTEP-IP      IFL-Idx  Interface  NH-Id  RVTEP-Mode  ELP-IP      Flags
192.168.2.1    19307    vtep.32771  53035  RNVE
RVTEP-IP      L2-RTT                IFL-Idx  Interface  NH-Id  RVTEP-Mode  ELP-IP
Flags
192.168.2.1    ri1                671522819 vtep-53.32771  53035  RNVE
  VNID      MC-Group-IP
  200        0.0.0.0
  300        0.0.0.0

user@leaf-1> show interfaces vtep.32771
Logical interface vtep.32771 (Index 20108) (SNMP ifIndex 537)
  Flags: Up Encapsulation: Unspecified
  VXLAN Endpoint Type: Configured-remote, VXLAN Endpoint Address: 192.168.2.1, L3 Routing
Instance: default
  Input packets : 0
  Output packets: 0
  Protocol ethernet-switching, MTU: Unlimited
  Flags: None
```

Small MC-LAG Network Static VXLAN Use Case

Here we show a static VXLAN at the global level in a small network that uses MC-LAGs.

Figure 214: Sample MC-LAG Network with Static VXLAN



In the MC-LAG network in [Figure 214 on page 2029](#):

- The bottom layer includes hosts (Host 1 through Host 4), each of which is multihomed to leaf devices and is a member of a VXLAN.
- The middle layer includes leaf devices (Leaf 1 through Leaf 4), each of which functions as an L2 VXLAN gateway. In addition, Leaf 1 and Leaf 2 form an MC-LAG and together, both leaf devices function as VTEP 1. Leaf 3 and Leaf 4 form another MC-LAG and function as VTEP 2.
- The top layer includes spine devices (Spine 1 and Spine 2), which form an MC-LAG. These devices function as IP routers.

Note the following about the configuration of Leaf 1 through Leaf 4 on which static VXLAN is configured:

- Instead of EVPN-VXLAN, here we enable static VXLAN.
- Each leaf device is an MC-LAG member and functions along with its MC-LAG peer as a VTEP. That is, two physical leaf devices make up each virtual VTEP. This situation impacts the configuration in the following ways:
 - On each leaf device that forms a VTEP, you must configure the same loopback address.
 - On each leaf device that forms a VTEP, you must configure the same remote VTEP list.

For a sample configuration of this use case, see ["Configure Static VXLAN at the Global Level" on page 2033](#).

- Instead of an IP multicast feature, we enable ingress node replication.

VLAN or Bridge Domain Level Static VXLAN Use Case

On some platforms, you can set up an L2 VXLAN gateway without EVPN at the VLAN or bridge domain level. In this case, you statically configure a list of one or more remote VTEPs at the global level, and map the list to a particular VLAN or bridge domain.



NOTE: You don't configure EVPN as part of a static VXLAN configuration. However, you can have EVPN-VXLAN and static VXLAN configurations on the same device, as long as the two configurations use different sets of remote peer VTEPs.

First, statically configure a global list of remote VTEPs using the `remote-vtep-list [...]` statement. Then map the same list of remote VTEPs at the VLAN or bridge domain level using the `static-remote-vtep-list [...]` statement. Each VLAN must also have a VXLAN network identifier (VNI) mapping.



NOTE: When you configure static VXLAN in EVPN-VXLAN deployments on the PTX10000 line of routers running Junos OS Evolved, you must also enable the `tunnel-termination option` at the `[edit forwarding-options]` hierarchy level. This enables the termination of tunnels on all interfaces at the global level.

We support configuring static VXLAN at the global level, VLAN level, and bridge domain level as follows:

Table 100: Supported Static VXLAN Configurations

Static VXLAN Configuration	Supported Platforms
In the default switch instance	MX Series QFX Series
In a virtual-switch routing instance	MX Series PTX10000 line of routers running Junos OS Evolved
At the global level	MX Series QFX Series PTX10000 line of routers running Junos OS Evolved

Table 100: Supported Static VXLAN Configurations (Continued)

Static VXLAN Configuration	Supported Platforms
At the VLAN level	MX Series QFX Series PTX10000 line of routers running Junos OS Evolved
At the bridge domain level	MX Series

When you specify the remote VTEPs at the VLAN or bridge domain level:

- In the default switch instance:

You must also specify the same VTEPs at the global level in the default switch instance at the [edit switch-options] hierarchy level.

- In a virtual-switch routing instance:

You must also specify the same VTEPs at the global level in the same routing instance at the [edit routing-instances *name*] hierarchy level.

To replicate and flood BUM traffic over static VXLAN tunnels at the VLAN or bridge domain level for VXLANs, you must also configure ingress node replication mode (ingress-node-replication option) at that level. This mode restricts the BUM traffic flood domain to only those VTEPs mapped to the particular VLAN or bridge domain.

For a sample configuration of this use case, see ["Configure Static VXLAN at the VLAN or Bridge Domain Level on L2 VXLAN Gateway Devices" on page 2036](#).

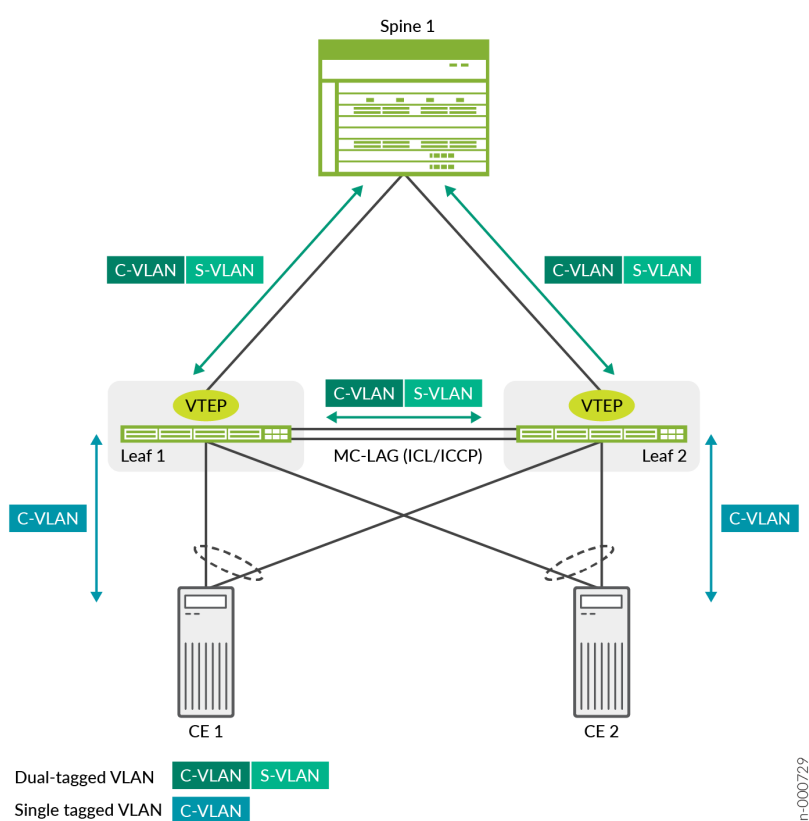
Q-in-Q VLAN Tunnels in a Spine-and-Leaf Network with Static VXLAN

Starting in Junos OS Release 23.4R1, Junos OS supports Q-in-Q tunnels in an MC-LAG spine-and-leaf network with static VXLAN tunnels. You must use service provider style configuration to configure Q-in-Q tunnels.

Service providers use Q-in-Q tunnels (VLAN translation) to segregate or bundle customer traffic into fewer or different VLANs by adding another layer of 802.1Q tags. In Q-in-Q tunneling, as a packet travels from a customer VLAN (C-VLAN) to a service provider's VLAN, Junos OS adds a service-provider defined VLAN (S-VLAN) tag to the packet. This additional 802.1Q tag allows service providers to create an L2 Ethernet connection between two customer sites in different geographic locations. As the packets exits the service provider's VLAN, Junos OS removes the extra 802.1Q (S-VLAN) tag.

Figure 215 on page 2032 shows how Junos devices handle Q-in-Q VLAN traffic in a spine-and-leaf topology. Leaf 1 and Leaf 2 are configured as static VXLAN gateway devices. Leaf 1 and Leaf 2 are also configured as MC-LAG peers to provide redundancy and load balancing. The VTEPs on the leaf devices are responsible for pushing the S-VLAN tag to the C-VLAN packet when it sends traffic to the spine and for popping the S-VLAN tag when it receives packets from the spine. In the event of a link failure between a leaf device and CE device, the packet is sent to the other leaf device. In which case, the leaf devices hold onto the S-VLAN tag.

Figure 215: VLANs in a Spine-and-Leaf Network



For more information about configuring the different elements in this features, see:

- ["Configure Static VXLAN at the VLAN or Bridge Domain Level on L2 VXLAN Gateway Devices" on page 2036](#)
- *Understanding Multichassis Link Aggregation Groups*
- *MC-LAG Examples*
- *Configuring Q-in-Q Tunneling and VLAN Q-in-Q Tunneling and VLAN Translation*

Configure Static VXLAN at the Global Level

You can implement VXLAN using the static VXLAN feature on L2 VXLAN gateway devices. Here we show using a static VXLAN in a small multichassis link aggregation group (MC-LAG) network. In the MC-LAG network, Juniper Networks devices function as source and destination virtual tunnel endpoints (VTEPs). In this environment, static VXLAN serves two purposes:

- To learn the MAC addresses of hosts in a VXLAN. To accomplish this task, static VXLAN uses the ingress node replication feature to flood BUM packets throughout a VXLAN. The VTEPs learn the MAC addresses of remote hosts from the VTEPs on the remote VTEP list and the MAC addresses of local hosts from the local access interfaces. Upon learning of a host's MAC address, the MAC address is then added to the Ethernet switching table.
- To encapsulate L2 packets with a VXLAN header and later de-encapsulate the packets, thereby enabling them to be tunneled through an underlying Layer 3 IP network. For this task to be accomplished, on each VTEP, you configure a list of statically defined remote VTEPs with which the local VTEP can form a VXLAN tunnel.

This sample configuration shows the steps and a sample configuration with the default switch instance rather than an explicitly configured L2 routing instance.



NOTE: See ["Configure Static VXLAN at the VLAN or Bridge Domain Level on L2 VXLAN Gateway Devices" on page 2036](#) for a sample configuration with a different use case—static VXLAN at the VLAN or bridge domain level. That configuration shows configuration steps and a sample configuration with an L2 routing instance of type virtual-switch.

Before You Begin

- Verify that Ethernet VPN (EVPN) is not enabled on the Juniper Networks devices that function as leaf devices.
- Verify that IP multicast is not enabled on the leaf devices.

In the sample MC-LAG network shown in [Figure 216 on page 2034](#), note that each VTEP (VTEP 1 and VTEP 2) comprises two physical leaf devices. That is, the two physical leaf devices function as a single virtual entity. As a result, for both leaf devices that compose a VTEP, you must configure the following:

- The same loopback anycast IP address.
- The same remote VTEP list.

Figure 216: Sample MC-LAG Network with Static VXLAN

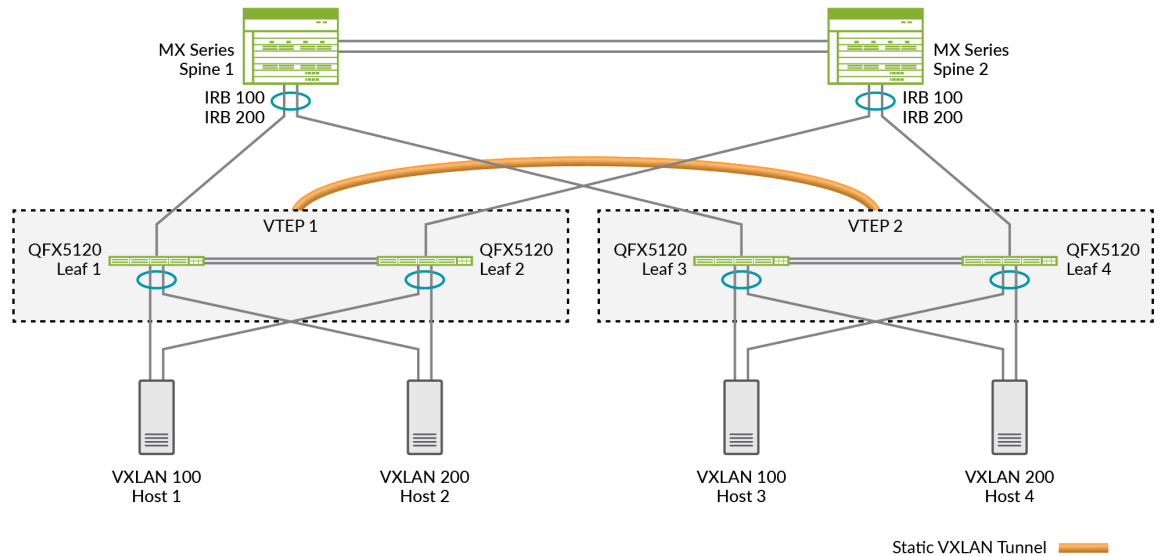


Table 101 on page 2034 shows the sample loopback address and remote VTEP list configured on each leaf device in the network.

Table 101: Sample Static VXLAN Configurations

Static VXLAN Parameters	VTEP 1: Leaf 1 Configuration	VTEP 1: Leaf 2 Configuration	VTEP 2: Leaf 3 Configuration	VTEP 2: Leaf 4 Configuration
Loopback anycast IP address	192.168.99.110/32	192.168.99.110/32	192.168.99.120/32	192.168.99.120/32
Remote VTEP list	192.168.99.120	192.168.99.120	192.168.99.110	192.168.99.110



NOTE: This procedure focuses on configuring the static VXLAN feature. It does not show how to configure peripheral but related entities such as interfaces, VLANs, and so on. However, the sample configuration that follows the procedure includes a more comprehensive configuration, including the related entities.

To enable static VXLAN on a leaf device:

1. Configure the loopback interface (lo0).

- a. Specify an anycast IP address as the primary address for the loopback interface.

```
[edit]user@switch# set interfaces lo0 unit logical-unit-number family inet address anycast-ip-address primary
```

- b. Specify the loopback interface as the source interface for VXLAN tunnels.

```
[edit]user@switch# set switch-options vtep-source-interface lo0.logical-unit-number
```

2. Create a list of statically defined remote VTEPs.

```
[edit]user@switch# set switch-options remote-vtep-list remote-vtep-loopback address(es)
```

3. For each VXLAN, enable ingress node replication.

```
[edit]user@switch# set vlans vlan-name vxlan ingress-node-replication
```

Sample Static VXLAN Configuration

These show configuration output snippets display a static VXLAN sample configuration for leafs 1 and 2 (VTEP 1) and include the parameters outlined in [Table 101 on page 2034](#).

```
interfaces {
  . . .
  ae0 {
    unit 0 {
      family ethernet-switching {
        interface-mode trunk;
        vlan {
          members 100 200;
        }
      }
    }
  }
  . . .
  lo0 {
    unit 0 {
      family inet {
        address 192.168.99.110/32 {
```

```

        primary;
    }
}
}
}
}
. . .
switch-options {
    vtep-source-interface lo0.0;
    remote-vtep-list 192.168.99.120;
}
. . .
vlans {
    vlan100 {
        vlan-id 100;
        vxlan {
            vni 1100;
            ingress-node-replication;
        }
    }
    vlan200 {
        vlan-id 200;
        vxlan {
            vni 2200;
            ingress-node-replication;
        }
    }
}
}

```

Configure Static VXLAN at the VLAN or Bridge Domain Level on L2 VXLAN Gateway Devices

In small spine and leaf networks, you can enable the static VXLAN feature on L2 VXLAN gateway devices that function as source and destination virtual tunnel endpoints (VTEPs). The L2 VXLAN gateway devices tunnel L2 packets over a VXLAN tunnel through an underlying Layer 3 IP network by:

- Encapsulating L2 packets with a VXLAN header when sending the packets to remote VTEPs.
- De-encapsulating the packets when receiving them from the tunnel to send to the destination hosts.

In this case, for each VTEP on the L2 VXLAN gateway devices, at the global level, statically configure the list of remote VTEPs with which the local VTEP can form a VXLAN tunnel. Then map the VTEP list to a particular VLAN or bridge domain in an L2 routing instance.

1. Statically configure a global list of remote VTEPs using the `remote-vtep-list` statement as follows:

In the default switch instance:

```
set switch-options remote-vtep-list [vtep-ip-addr1 <vtep-ip-addr2 ...>]
```

In a virtual-switch routing instance:

```
set <routing-instances name> remote-vtep-list [vtep-ip-addr1 <vtep-ip-addr2 ...>]
```

2. Map the remote VTEPs list to a particular VLAN or bridge domain as follows:

- Configure the VLAN with a VXLAN VNI mapping
- Map the remote VTEPs list to the VLAN using the `static-remote-vtep-list` statement:

In the default switch instance:

```
set vlans vlan-name vxlan static-remote-vtep-list [vtep-ip-addr1 <vtep-ip-addr2 ...>]
```

In a virtual-switch routing instance:

```
set <routing-instances name> vlans vlan-name vxlan static-remote-vtep-list [vtep-ip-addr1  
<vtep-ip-addr2 ...>]
```

See ["VLAN or Bridge Domain Level Static VXLAN Use Case" on page 2030](#) for an overview of how static VXLAN works in this case. See [Table 100 on page 2030](#) for a summary of the supported static VXLAN configurations.



NOTE: If you configure a global list of remote VTEPs in a routing instance using the `remote-vtep-list` statement, but you don't also configure the `static-remote-vtep-list` statement for a VLAN in the routing instance, then that VLAN inherits all the globally configured remote VTEPs.

Here we include the configuration steps and a simple sample configuration on two L2 VXLAN gateway devices to set up static VXLAN tunnels between them at the VLAN level. [Table 102 on page 2038](#) shows the parameters for this configuration. We use L2 routing instances of type `virtual-switch`, so you configure most elements in the routing instance at the `[edit routing-instances name]` hierarchy level. Each step notes the statement hierarchy differences for the default switch instance instead.

Table 102: Sample Static VXLAN Configuration at the VLAN Level

Device	lo0 IP Address	L2 Instance Name	Associated VLAN	VLAN to VNI mapping	Remote VTEP IP Addresses
VXLAN-GW1	192.168.4.1	rt1	v100 (VLAN ID 100)	100	192.168.5.1
			v200 (VLAN ID 200)	200	
VXLAN-GW2	192.168.5.1	rt1	v100 (VLAN ID 100)	100	192.168.4.1
			v200 (VLAN ID 200)	200	

1. Configure the loopback interface (lo0).
 - a. Specify an anycast IP address address for the loopback interface.

For example, for VXLAN-GW1 in [Table 102 on page 2038](#):

```
set interfaces lo0 unit 0 family inet address 192.168.4.1/32 primary
set interfaces lo0 unit 0 family inet address 192.168.4.1/32 preferred
```

- b. Specify the loopback interface as the source interface for VXLAN tunnels in the routing instance.
- For example, in L2 routing instance rt1:

```
set routing-instances rt1 vtep-source-interface lo0.0
```

If your configuration uses the default switch instance, set lo0.0 as the vtep-source-interface at the [edit switch-options] hierarchy level instead of in a routing instance.

2. Configure the VLANs in the L2 routing instance (or the default switch instance) with an associated interface. Map each VLAN to a VNI for the VXLAN tunnel.

For example, for VXLAN-GW1 in [Table 102 on page 2038](#):

```
set interfaces et-0/0/7 unit 0 family ethernet-switching interface-mode trunk
set interfaces et-0/0/7 unit 0 family ethernet-switching vlan members 100
set interfaces et-0/0/7 unit 0 family ethernet-switching vlan members 200
set routing-instances rt1 interface et-0/0/7.0
```

```
set routing-instances rt1 vlans v100 vlan-id 100
set routing-instances rt1 vlans v100 vxlan vni 100
set routing-instances rt1 vlans v200 vlan-id 200
set routing-instances rt1 vlans v200 vxlan vni 200
```

If your configuration uses the default switch instance, set the VLAN parameters above globally at the [edit vlans] hierarchy level instead of in a routing instance.

3. Create the list of statically defined remote VTEPs in the L2 routing instance at the global level.

For example, [Table 102 on page 2038](#) shows VXLAN-GW1 has VXLAN-GW2 as its remote VTEP:

```
set routing-instances rt1 remote-vtep-list 192.168.5.1
```

If your configuration uses the default switch instance, set remote VTEPs list at the [edit switch-options] hierarchy level instead of in a routing instance.

4. Statically map the remote VTEPs to particular VLANs in the L2 routing instance.

For example, on VXLAN-GW1, use the static-remote-vtep-list statement at the VLAN level in the routing instance rt1 to map the same remote VTEP from the global list to VLANs v100 and v200:

```
set routing-instances rt1 vlans v100 vxlan static-remote-vtep-list 192.168.5.1
set routing-instances rt1 vlans v200 vxlan static-remote-vtep-list 192.168.5.1
```

If your configuration uses the default switch instance, set the VLAN level remote VTEPs list at the [edit vlans *vlan-name* vxlan] hierarchy level instead of in a routing instance.

5. For each VXLAN, enable ingress node replication to replicate and flood BUM traffic only to those VTEPs you map to the associated VLAN(s):

```
set routing-instances rt1 vlans v100 vxlan ingress-node-replication
```



NOTE: On supported QFX5100 and QFX5120 switches, to ensure that BUM traffic is flooded to only those VTEPs mapped to a particular VLAN, we strongly recommend you configure the static VTEP lists on all of those switches symmetrically. We recommend these settings to avoid situations where, for example:

- Switch 1's list includes switches 2 and 3,
- Switch 2's list includes switches 1 and 3, and

- Switch 3's list includes only switch 2.

Because of this asymmetric configuration, when switch 1 sends BUM traffic to switch 2 and switch 3 on its list, Switch 3 forwards the BUM traffic to its local ports in the VLAN.

If your configuration uses the default switch instance, set the `vxlan ingress-node-replication` option at the `[edit vlans name]` hierarchy level instead of in a routing instance.

See the following sample set command configuration blocks that show complementary configurations on VXLAN-GW1 and VXLAN-GW2. Using the parameters in [Table 102 on page 2038](#), these commands set up a static VXLAN tunnel between the two devices in L2 routing instance `rt1` at the VLAN level for VLANs 100 and 200.

VXLAN-GW1:

```
set interfaces et-0/0/3 unit 0 family inet address 10.1.1.1/24
set interfaces et-0/0/7 unit 0 family ethernet-switching interface-mode trunk
set interfaces et-0/0/7 unit 0 family ethernet-switching vlan members 100
set interfaces et-0/0/7 unit 0 family ethernet-switching vlan members 200
set interfaces lo0 unit 0 family inet address 192.168.4.1/32 primary
set interfaces lo0 unit 0 family inet address 192.168.4.1/32 preferred
set routing-instances rt1 instance-type virtual-switch
set routing-instances rt1 vtep-source-interface lo0.0
set routing-instances rt1 remote-vtep-list 192.168.5.1
set routing-instances rt1 interface et-0/0/7.0
set routing-instances rt1 vlans v100 vlan-id 100
set routing-instances rt1 vlans v100 vxlan vni 100
set routing-instances rt1 vlans v100 vxlan ingress-node-replication
set routing-instances rt1 vlans v100 vxlan static-remote-vtep-list 192.168.5.1
set routing-instances rt1 vlans v200 vlan-id 200
set routing-instances rt1 vlans v200 vxlan vni 200
set routing-instances rt1 vlans v200 vxlan ingress-node-replication
set routing-instances rt1 vlans v200 vxlan static-remote-vtep-list 192.168.5.1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface et-0/0/3.0
```

VXLAN-GW2:

```
set interfaces ge-0/0/1 unit 0 family inet address 10.1.1.2/24
set interfaces ge-0/0/3 unit 0 family ethernet-switching interface-mode trunk
set interfaces ge-0/0/3 unit 0 family ethernet-switching vlan members 100
set interfaces ge-0/0/3 unit 0 family ethernet-switching vlan members 200
```

```
set interfaces lo0 unit 0 family inet address 192.168.5.1/32 primary
set interfaces lo0 unit 0 family inet address 192.168.5.1/32 preferred
set routing-instances rt1 instance-type virtual-switch
set routing-instances rt1 vtep-source-interface lo0.0
set routing-instances rt1 remote-vtep-list 192.168.4.1
set routing-instances rt1 interface ge-0/0/3.0
set routing-instances rt1 vlans v100 vlan-id 100
set routing-instances rt1 vlans v100 vxlan vni 100
set routing-instances rt1 vlans v100 vxlan ingress-node-replication
set routing-instances rt1 vlans v100 vxlan static-remote-vtep-list 192.168.4.1
set routing-instances rt1 vlans v200 vlan-id 200
set routing-instances rt1 vlans v200 vxlan vni 200
set routing-instances rt1 vlans v200 vxlan ingress-node-replication
set routing-instances rt1 vlans v200 vxlan static-remote-vtep-list 192.168.4.1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
```

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
20.4R1	Starting in Junos OS Release 20.4R1, Juniper Networks supports configuring static VXLAN VTEPs at the VLAN or bridge domain level as well as at the previously supported global level.
14.1X53-D40	Starting in Junos OS Release 14.1X53-D40, Juniper Networks supports static VXLAN in small MC-LAG networks.

11

PART

Configuration Statements and Operational Commands

-
- [Junos CLI Reference Overview](#) | 2043
-

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](#)