L2VPN and Ethernet Services Configuration Guide for Cisco NCS 560 Series Routers, IOS XR Release 7.8.x

First Published: 2022-11-30

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2022 Cisco Systems, Inc. All rights reserved.

CONTENTS

CHAPTER 1 **Configure Gigabit Ethernet for Layer 2 VPNs** 1

Introduction to Layer 2 Virtual Private Networks 1

Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces 1

Configure Gigabit Ethernet Interfaces for Layer 2 Transport 3

Ethernet Data Plane Loopback 4

Configure Ethernet Data Plane Loopback 6

Running Configuration 7

Verification 8

CHAPTER 2 **Configure Layer 2 Access Control Lists** 9

Layer 2 Access Control Lists 9

Prerequisites for Configuring Layer 2 Access Control Lists 9

Layer 2 Access Control Lists Feature Highlights 10

Purpose of Layer 2 Access Control Lists 10

How a Layer 2 Access Control List Works 10

Layer 2 Access Control List Process and Rules 10

Create Layer 2 Access Control List 11

Restrictions for Configuring Layer 2 Access Control Lists 11

Configuration 11

Running Configuration 12

Verification 12

CHAPTER 3 **Configure Virtual LANs in Layer 2 VPNs** 13

Configure VLAN Subinterfaces 15

Introduction to Ethernet Flow Point 19

Identify Frames of an EFP 19

Apply Features 21
Define Data-Forwarding Behavior 21
Ethernet Flow Points Visibility 22
Configuring EFP Visibility 22
Configure VLAN Header Rewrite 24
Valid Ingress Rewrite Actions 27
Valid Ingress-Egress Rewrite Combinations 28
Rewrite of Priority Tag 36
Configure Rewrite of Priority Tag 36

CHAPTER 4 **L2CP Tunneling MEF 39**
L2CP Tunneling 39
L2CP Protocol Support on Cisco NCS Series Router 40

CHAPTER 5 **Configure Link Bundles for Layer 2 VPNs 43**
Configure Gigabit Ethernet Link Bundle 43
Configure VLAN Bundle 46
References for Configuring Link Bundles 47
Characteristics of Link Bundles 48
Methods of Forming Bundles of Ethernet Interfaces 48
Link Aggregation Through LACP 49

CHAPTER 6 **Configure Multipoint Layer 2 Services 51**
Prerequisites for Implementing Multipoint Layer 2 Services 51
Information About Implementing Multipoint Layer 2 Services 52
Multipoint Layer 2 Services Overview 52
Restrictions for VPLS 52
Bridge Domain 53
Bridge Domain and BVI Scale 53
Pseudowires 54
Access pseudowires 54
Virtual Forwarding Instance 56
Information About Implementing Multipoint Layer 2 Services 57
VPLS for an MPLS-based Provider Core 57

VPLS for Layer 2 Switching 57
Storm Control on Bridge Domain 58
Interoperability Between Cisco IOS XR and Cisco IOS on VPLS LDP Signaling 64
VPLS VFI with BVI as Routed Interface 64
MAC Address-related Parameters 67
How to Implement Services 72
Configuring a Bridge Domain 72
Creating a Bridge Domain 72
Associating Members with a Bridge Domain 73
Configuring Bridge Domain Parameters 74
Disabling a Bridge Domain 76
Flooding Disable 77
Configure Flooding Disable 77
Configuring a Layer 2 Virtual Forwarding Instance 79
Creating the Virtual Forwarding Instance 79
Associating Pseudowires with the Virtual Forwarding Instance 80
Associating a Virtual Forwarding Instance to a Bridge Domain 81
Attaching Pseudowire Classes to Pseudowires 83
Configuring Pseudowires Using Static Labels 84
Disabling a Virtual Forwarding Instance 86
Configuring the MAC Address-related Parameters 87
Configuring the MAC Address Source-based Learning 88
Configuring the MAC Address Aging 89
Disabling MAC Flush at the Bridge Port Level 91
MAC Address Withdrawal 92
Configure MAC Address Withdrawal 93
MAC Loop Prevention 95
Configuration Examples for Multipoint Layer 2 Services 99
Multipoint Layer 2 Services Configuration for Provider Edge-to-Provider Edge: Example 99
Multipoint Layer 2 Services Configuration for Provider Edge-to-Customer Edge: Example 100
Displaying MAC Address Withdrawal Fields: Example 100
Bridging on IOS XR Trunk Interfaces: Example 102
Bridging on Ethernet Flow Points: Example 105
LDP-Based VPLS and VPWS FAT Pseudowire 108

Configure LDP-Based VPLS and VPWS FAT Pseudowire 109
PPPoE traffic-based load balancing 113
Configure Load balancing for PPPoE Traffic 115

CHAPTER 7 **Configure Point-to-Point Layer 2 Services** 117
Ethernet over MPLS 118
Ethernet Port Mode 119
VLAN Mode 119
QinQ Mode 120
Configure Local Switching Between Attachment Circuits 121
Configure Static Point-to-Point Connections Using Cross-Connect Circuits 125
Flexible Cross-Connect Service 127
Flexible Cross-Connect Service - Single-Homed 128
Flexible Cross-Connect Service - Multi-Homed 128
Flexible Cross-Connect Service Supported Modes 129
VLAN Unaware 129
Configure Single-Homed Flexible Cross-Connect Service using VLAN Unaware 129
Configure Multi-Homed Flexible Cross-Connect Service using VLAN Unaware 131
VLAN Aware 134
Configure Single-Homed Flexible Cross-Connect using VLAN Aware 135
Configure Multi-Homed Flexible Cross-Connect Service using VLAN Aware 136
Local Switching 140
Configure Multi-Homed Flexible Cross-Connect Service using Local Switching 141
Configure Preferred Tunnel Path 143
Multisegment Pseudowire 144
Multisegment Pseudowire Redundancy 146
Configure Multisegment Pseudowire 146
Split Horizon Groups 150
Configure Split Horizon Group 2 151
G.8032 Ethernet Ring Protection 153
Configure G.8032 Ethernet Ring Protection 157
Configure ERP Profile 158
Configuring an ERP Instance 158
Configuring G.8032 Ethernet Ring Protection: Example 160

Configuring Interconnection Node: Example 161
Configuring the Node of an Open Ring: Example 162
Pseudowire Redundancy 163
Configure Pseudowire Redundancy 164
Running Configuration 164
Verification 165
Configure Pseudowire Redundancy 166
Access Pseudowire Redundancy 167
Configure Access Pseudowire Redundancy 168
Virtual Circuit Connection Verification on L2VPN 169
GTP Load Balancing 169
VPLS over SR-TE and RSVP-TE 171

CHAPTER 8 **Information About Multiple Spanning Tree Protocol 173**
Spanning Tree Protocol Overview 173
STP Protocol Operation 174
Topology Changes 174
Variants of STP 174
Multiple Spanning Tree Protocol 175
MSTP Regions 175
MSTP Port Fast 176
MSTP Root Guard 177
MSTP Topology Change Guard 177
MSTP Supported Features 178
BPDU Guard 178
Flush Containment 178
Bringup Delay 179
Restrictions 179
Access Gateway 180
Overview of Access Gateway 181
Topology Change Propagation 182
Preempt Delay 183
Supported Access Gateway Protocols 183
MSTAG Edge Mode 183

PVSTAG on Bundle Interfaces 185
Per-VLAN Rapid Spanning Tree 185
Multiple VLAN Registration Protocol 186
How to Implement Multiple Spanning Tree Protocol 186
Configuring MSTP 187
Enabling MSTP 187
Configuring MSTP parameters 187
Verifying MSTP 192
Configuring MSTAG or REPAG 192
Configuring an untagged subinterface 192
Enabling MSTAG 193
Configuring MSTAG parameters 193
Configuring MSTAG Topology Change Propagation 198
Verifying MSTAG 198
MSTAG Uplink Tracking 199
Benefits 200
Prerequisites 200
Restrictions 200
Configure MSTAG Uplink Tracking 200
Configuring PVSTAG or PVRSTAG 202
Enabling PVSTAG 202
Configuring PVSTAG parameters 202
Configuring Subinterfaces 206
Verifying PVSTAG 206
Configuring PVRST 207
Configuring MVRP-lite 208
Configuring MVRP-lite 208
Configuring MVRP-lite parameters 208
Verifying MVRP-lite 210
Configuration Examples for Implementing MSTP 210
Configuring MSTP: Examples 210
Configuring MSTAG: Examples 215
Configuring PVSTAG: Examples 217
Configuring PVRST: Example 218

Configuring MVRP-Lite: Examples 218

CHAPTER 9 EVPN Features 221

EVPN Overview 222
EVPN Concepts 223
EVPN Operation 224
EVPN Route Types 225
EVPN Timers 226
Configure EVPN L2 Bridging Service 226
Running Configuration 227
EVPN Single-Active Multi-Homing Mode 228
EVPN Software MAC Learning 234
Configure EVPN Software MAC Learning 235
Supported Modes for EVPN Software MAC Learning 236
Single Home Device or Single Home Network Mode 236
Configure EVPN in Single Home Device or Single Home Network Mode 236
Dual Home Device—All-Active Load Balancing Mode 238
Configure EVPN Software MAC Learning in Dual Home Device—All-Active Mode 238
Verify EVPN Software MAC Learning 240
EVPN Out of Service 242
Configure EVPN Out of Service 243
Running Configuration 244
CFM Support for EVPN 246
Control Word Support for ELAN 246
EVPN Multiple Services per Ethernet Segment 247
Configure EVPN Multiple Services per Ethernet Segment 247
Configuration Example 247
Running Configuration 250
Associated Commands 253
EVPN Single-Flow-Active Multihoming Load-Balancing Mode 253
EVPN Convergence Using NTP Synchronization 258
EVPN MPLS Seamless Integration with VPLS 261
Migrate VPLS Network to EVPN Network through Seamless Integration 261
Configure EVPN on the Existing VPLS Network 262

Configure L2 EVPN Address-Family 262
Configure EVI and Corresponding BGP Route Target under EVPN Configuration Mode 263
Configure EVI under a Bridge Domain 264
EVI Configuration Under L2VPN Bridge-Domain 264
Verify EVPN Configuration 265
Clear Forwarding Table 269
Hierarchical EVPN Access Pseudowire 269
Configure Hierarchical EVPN Access Pseudowire 270
EVPN Seamless Integration with VPWS 271
Network Convergence using Core Isolation Protection 277
Configure EVPN Convergence using Core Isolation Protection 279
Configurable Recovery Time for EVPN Core Isolation Group 283
Configurable Recovery Time for EVPN Core Isolation Group 284
Conditional Advertisement of Default-Originate 290
Configure Conditional Advertisement of Default-Originate 291
EVPN Single-Active Multihoming for Anycast Gateway IRB 294
Configure EVPN Single-Active Multihoming 294
Configure EVPN Ethernet Segment 295
Configure EVPN Service Instance (EVI) Parameters 295
Configure Layer 2 Interface 295
Configure a Bridge Domain 296
EVPN Core Isolation Protection 296
Configure EVPN Core Isolation Protection 297
Restrictions 297
Running Configuration 298
Verification 298
EVPN Routing Policy 299
EVPN Route Types 299
EVPN RPL Attribute 304
EVPN RPL Attribute Set 306
Configure EVPN RPL Feature 307
Running Configuration 308
CFM on EVPN ELAN 314
Configure CFM on EVPN ELAN 316

EVPN Bridging and VPWS Services over BGP-LU Underlay 322
Configure EVPN Bridging and VPWS Services over BGP-LU Underlay 324
Support for DHCPv4 and DHCPv6 Client over BVI 334
Configure DHCPv4 and DHCPv6 Client over BVI 334
Layer 2 fast reroute 339
Restrictions for layer 2 fast reroute 341
Configure layer 2 fast reroute for E-LAN service 341
EVPN Preferred Nexthop 344
Configure EVPN Preferred Nexthop 345
EVPN Access-Driven DF Election 346
Configure EVPN Access-Driven DF Election 353

CHAPTER 10 **Configure EVPN IRB, Distributed Anycast Gateway and E-tree** 359
EVPN IRB 359
EVPN Single-Homing Access Gateway 361
EVPN Multihoming All-Active 362
EVPN Single-Active Multihoming for Anycast Gateway IRB 363
Configure EVPN Single-Active Multihoming 363
Configure EVPN IRB with Host Routing 364
Configure EVPN Ethernet Segment 364
Configure EVPN Service Instance (EVI) Parameters 365
Configure Layer 2 Interface 365
Configure a Bridge Domain 366
Configure VRF 366
Enable Auto-BGP RT with Manual ESI Configuration 367
Supported EVPN IRB Scenarios 367
Distributed Anycast Gateway 367
EVPN IRB with All-Active Multi-Homing without Subnet Stretch or Host-Routing across the Fabric 368
EVPN IRB with All-Active Multihoming with Subnet Stretch or Host-Routing across the Fabric 368
MAC and IP Unicast Control Plane 369
Intra-subnet Unicast Data Plane 370
Inter-subnet Unicast Data Plane 370
BVI-Coupled Mode 370

VM Mobility Support 372
MAC and MAC-IP Sequence Numbers 372
Synchronized MAC and MAC-IP Sequence Numbers 372
Local Sequence Number Updates 372
Best Route Selection after Host Movement 373
Stale Route Deletion after a Host Movement 373
Host Movement Detection through GARP 373
Host Move Detection with Silent Host 373
Host Move Detection without GARP with Data Packet 373
Duplicate MAC Detection 373
Configuring EVPN IRB 374
Running Configuration for EVPN IRB 375
Verify EVPN IRB 376
EVPN IPv6 Hosts with Mobility 386
Configure EVPN IPv6 Hosts with Mobility 387
Running Configuration 391
Verification 395
Duplicate IP Address Detection 397
Configure Duplicate IP Address Detection 398
Configuration Example 398
Running Configuration 398
Verification 399
EVPN Automatic Unfreezing of MAC and IP Addresses 399
Configure EVPN Automatic Unfreezing of MAC or IP Address 400
EVPN E-Tree 402
Configure EVPN E-Tree 406
Configuration Example 406
Running Configuration 407
Verification 408
EVPN E-Tree Using RT Constraints 410
CE with Multihoming Active-Active and CE with Multihoming Active-Active 411
EVPN E-Tree Per-PE (Scenario 1b) 425
Configure EVPN E-Tree Per-PE (Scenario1b) 427
DHCPv4 Relay on IRB 431

Configure DHCPv4 Relay on IRB 436
Configuration Example 436
Running Configuration 437
DHCPv4 Relay Synchronization for All-Active Multihoming 438
DHCPv6 Relay IAPD on IRB 439
Configure DHCPv6 Relay IAPD on IRB 440
Configuration Example 440
Running Configuration 441
DHCPv6 PD Synchronization for All-Active Multihoming using Session Redundancy 442
Configure DHCPv6 PD Synchronization 443
Configuration Example 443
Running Configuration 443
IAPD Route Distribution and Withdrawal in DHCPv6 Relay 445

CHAPTER 11 **EVPN Virtual Private Wire Service (VPWS) 447**
Supported Modes for EVPN-VPWS 447
Benefits of EVPN-VPWS 447
Prerequisites for EVPN-VPWS 448
Restrictions for EVPN-VPWS 448
EVPN-VPWS Single Homed 448
Configure EVPN-VPWS Single Homed 449
Running Configuration 450
EVPN-VPWS Multi-Homed 450
Configure EVPN-VPWS All-Active Multi-Homed 451
Running Configuration 452
Flow Label Support for EVPN VPWS 453
Configure Flow Label for EVPN VPWS 455

CHAPTER 12 **L2VPN Preferred path 457**
L2VPN Services over Segment Routing for Traffic Engineering Policy 457
EVPN VPWS Preferred Path over SR-TE Policy 458
Configure EVPN VPWS Preferred Path over SR-TE Policy 459
Configure Prefix-SID in ISIS 459
Configure Adjacency-SID in ISIS 461

Configure Segment-list 463
Configure SR-TE Policy 464
Configure EVPN VPWS over SR-TE Policy 465
Running Configuration 465
Verify EVPN VPWS Preferred Path over SR-TE Policy Configuration 470
Associated Commands 471
Related Topics 471
L2VPN VPWS Preferred Path over SR-TE Policy 471
Configure L2VPN VPWS Preferred Path over SR-TE Policy 471
Configure Prefix-SID in IS-IS 472
Configure Adjacency-SID in IS-IS 473
Configure Segment-list 475
Configure SR-TE Policy 476
Configure VPWS over SR-TE Policy 477
Running Configuration 478
Verify L2VPN VPWS Preferred Path over SR-TE Policy Configuration 481
Associated Commands 483
Related Topics 484
EVPN VPWS On-Demand Next Hop with SR-TE 484
Configure EVPN VPWS On Demand Next Hop with SR-TE 485
Topology 485
Configure Prefix-SID in ISIS 485
Configure SR-TE 487
Configure PCE and PCC 488
Configure SR Color 488
Configure EVPN Route Policy 489
Configure BGP 490
Configure EVPN VPWS 490
Configure Flexible Cross-connect Service (FXC) VLAN-unaware 491
Running Configuration 491
Related Topics 498
Overview of Segment Routing 498
How Segment Routing Works 499
Segment Routing Global Block 500

CHAPTER 13 **Configure BPDU Transparency with MACsec** 501

Layer 2 Control Plane Tunneling in MACsec 501

MACsec and MKA Overview 501

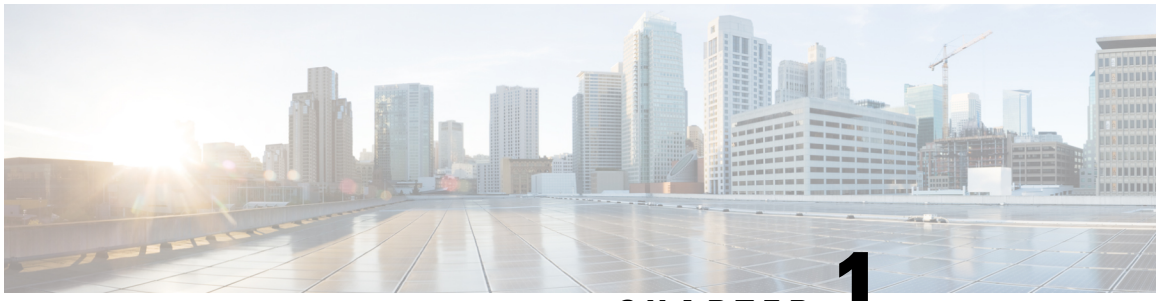
L2CP Tunneling 502

L2CP Tunneling in MACsec 502

Configuration 502

Running Configuration 504

Verification 505



CHAPTER 1

Configure Gigabit Ethernet for Layer 2 VPNs

This chapter introduces you to Layer 2 features and standards, and describes how you can configure L2VPN features.

The distributed Gigabit Ethernet (including 10-Gigabit and 100-Gigabit) architecture and features deliver network scalability and performance, while enabling service providers to offer high-density, high-bandwidth networking solutions designed to interconnect the router with other systems in POPs, including core and edge routers and Layer 2 and Layer 3 switches.

- Introduction to Layer 2 Virtual Private Networks, on page 1
- Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces, on page 1
- Configure Gigabit Ethernet Interfaces for Layer 2 Transport, on page 3
- Ethernet Data Plane Loopback, on page 4

Introduction to Layer 2 Virtual Private Networks

A Layer 2 Virtual Private Network (VPN) emulates a physical sub-network in an IP or MPLS network, by creating private connections between two points. Building a L2VPN network requires coordination between the service provider and customer. The service provider establishes Layer 2 connectivity. The customer builds a network by using the data link resources obtained from the service provider. In a L2VPN service, the service provider does not require information about the customer's network topology and other information. This helps maintain customer privacy, while using the service provider resources to establish the network.

The service provider requires Provider Edge (PE) routers with the following capabilities:

- Encapsulation of L2 protocol data units (PDU) into Layer 3 (L3) packets.
- Interconnection of any-to-any L2 transports.
- Support for MPLS tunneling mechanism.
- Process databases that include all information related to circuits and their connections.

This section introduces Layer 2 Virtual Private Networks (VPNs) and the corresponding Gigabit Ethernet services.

Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces

A L2VPN network enables service providers (SPs) to provide L2 services to geographically disparate customer sites. Typically, a SP uses an access network to connect the customer to the core network. This access network may use a mixture of L2 technologies, such as Ethernet and Frame Relay. The connection between the customer

site and the nearby SP edge router is known as an attachment circuit (AC). Traffic from the customer travels over this link to the edge of the SP core network. The traffic then tunnels through a pseudowire over the SP core network to another edge router. The edge router sends the traffic down another AC to the customer's remote site.

The L2VPN feature enables the connection between different types of L2 attachment circuits and pseudowires, allowing users to implement different types of end-to-end services.

Note BOOTP traffic (dst UDP 68) over any type of pseudowire is unsupported.

Cisco IOS XR software supports a point-to-point end-to-end service, where two Ethernet circuits are connected together. An L2VPN Ethernet port can operate in one of two modes:

- Port Mode—In this mode, all packets reaching the port are sent over the pseudowire, regardless of any VLAN tags that are present on the packets. In Port mode, the configuration is performed under the l2transport configuration mode.
- VLAN Mode—Each VLAN on a CE (customer edge) or access network to PE (provider edge) link can be configured as a separate L2VPN connection (using either VC type 4 or VC type 5). To configure L2VPN on VLANs, see *The Carrier Ethernet Model* chapter in this manual. In VLAN mode, the configuration is performed under the individual sub-interface.

Switching can take place in the following ways:

- AC-to-PW—Traffic reaching the PE is tunneled over a PW (pseudowire) (and conversely, traffic arriving over the PW is sent out over the AC). This is the most common scenario.
- Local switching—Traffic arriving on one AC is immediately sent out of another AC without passing through a pseudowire.
- PW stitching—Traffic arriving on a PW is not sent to an AC, but is sent back into the core over another PW.

Note

- If your network requires that packets are transported transparently, you may need to modify the packet's destination MAC (Media Access Control) address at the edge of the Service Provider (SP) network. This prevents the packet from being consumed by the devices in the SP network.
- The **encapsulation dot1ad** *vlan-id* and **encapsulation dot1ad** *vlan-id* **dot1q any** commands cannot co-exist on the same physical interface or bundle interface. Similarly, the **encapsulation dot1q** *vlan-id* and **encap dot1q** *vlan-id* **second-dot1q any** commands cannot co-exist on the same physical interface or bundle interface. If there is a need to co-exist, it is recommended to use the exact keyword in the single tag encapsulation. For example, **encap dot1ad** *vlan-id* **exact** or **encap dot1q** *vlan-id* **exact**.
- In an interface which already has QinQ configuration, you cannot configure the QinQ Range sub-interface where outer VLAN range of QinQ Range overlaps with outer VLAN of QinQ. Attempting this configuration results in the splitting of the existing QinQ and QinQ Range interfaces. However, the system can be recovered by deleting a recently configured QinQ Range interface.
- In an interface which already has QinQ Range configuration, you cannot configure the QinQ Range sub-interface where outer VLAN range of QinQ Range overlaps with inner VLAN of QinQ Range. Attempting this configuration results in the splitting of the existing QinQ and QinQ Range interfaces. However, the system can be recovered by deleting a recently configured QinQ Range interface.
- The inner VLAN ranges of sub-interfaces configured cannot have overlapping values. In such overlapping inner VLAN range cases, the system can be recovered by reloading the LC on Cisco IOS XR Release 6.5.x.

You can use the **show interfaces** command to display AC and pseudowire information.

Configure Gigabit Ethernet Interfaces for Layer 2 Transport

This section describes how you can configure Gigabit ethernet interfaces for Layer 2 transport.

Configuration Example

```
RP/0/RP0/CPU0(config)#interface TenGigE 0/0/0/10

/* Configure the ethertype for the 802.1q encapsulation (optional) */
/* For VLANs, the default ethertype is 0x8100. In this example, we configure a value of
0x9100.
/* The other assignable value is 0x9200 */
/* When ethertype is configured on a physical interface, it is applied to all sub-interfaces
 created on this interface */

RP/0/RP0/CPU0:router(config-if)#dot1q tunneling ethertype 0x9100

/* Configure Layer 2 transport on the interface, and commit your configuration */
RP/0/RP0/CPU0:router(config-if)#l2transport
RP/0/RP0/CPU0:router(config-if-l2)#commit
Sat May  2 19:50:36.799 UTC
RP/0/RP0/CPU0:router(config-if-l2)#exit
RP/0/RP0/CPU0:router(config-if)#no shutdown
RP/0/RP0/CPU0:router(config-if)#exit
RP/0/RP0/CPU0:router(config)#
```

Running Configuration

```
configure
 interface TenGigE 0/0/0/10
  dot1q tunneling ethertype 0x9100
  l2transport
 !
```

Verification

Verify that the Ten-Gigabit Ethernet interface is up and operational.

```
router# show interfaces TenGigE 0/0/0/10

...
TenGigE0/0/0/10 is up, line protocol is up
  Interface state transitions: 1
  Hardware is TenGigE, address is 0011.1aac.a05a (bia 0011.1aac.a05a)
  Layer 1 Transport Mode is LAN
  Layer 2 Transport Mode
  MTU 1514 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 10000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  ...
```

Associated Commands

- l2transport (Ethernet)

Ethernet Data Plane Loopback

The Ethernet Data Plane Loopback function allows you to run loopback tests to test the connectivity and quality of connections through a Layer 2 cloud. You can run this test on:

- Main interface or sub-interfaces
- Bundle or its sub-interfaces
- Multiple hops through the underlying network

You can use this feature to test the throughput of an Ethernet port remotely. You can verify the maximum rate of frame transmission with no frame loss.

This feature allows for bidirectional or unidirectional throughput measurement, and on-demand or out-of-service (intrusive) operation during service turn-up.

Two types of Ethernet loopback are supported:

- External loopback - Traffic loopback occurs at the Ingress interface. Traffic does not flow into the router for loopback.

- Internal loopback - Traffic loopback occurs at the Egress interface. Traffic loopback occurs after the traffic flows into the router to the other interface.

Ethernet data traffic can be looped back on per port basis. This feature supports a maximum of 100 concurrent Ethernet data plane loopback sessions per system. Filters based on frame header can be used for initiating the loopback session. This ensures that only a subset of traffic that is received on an interface is looped back. You can use Source MAC, Destination MAC, and VLAN Priority (COS bits) as filters.

Ethernet Data Plane Loopback Configuration Restrictions

These configuration restrictions are applicable for Ethernet Data Plane Loopback:

- The maximum supported Ethernet data plane loopback session at system level is 100.
- CFM UP MEP is not supported with Ethernet data plane loopback.
- Ethernet data plane loopback is not supported on L3 interfaces or L3 sub-interfaces.
- The following filters are not supported:
  - Outer VLAN or range of outer VLAN
  - Inner VLAN or range of inner VLAN
  - Ether type
- Only the following combinations of filters are supported for external loopback:
  - Source MAC
  - Source MAC and Destination MAC
  - Source MAC, Destination MAC, and VLAN priority
  - Destination MAC
  - Destination MAC and VLAN priority
- The rewrite modification on the loopback traffic is not supported.

Note Ensure that no rewrite should be configured on subinterface.

- Ethernet data plane loopback is not supported on BVI interface.
- Only one Ethernet loopback session, either internal or external, can be active on the same interface at any given instance.
- This feature supports a maximum throughput of 10Gbps for internal loopback over all the sessions. For external loopback, there is no throughput limit.
- Dropping of packets that are received in the non-loopback direction is not supported.
- Ethernet data plane loopback is not supported on packets having destination as multicast and broadcast MAC address.
- External and internal Ethernet data plane loopback is not supported over bridge domain.

- The Cisco NCS Routers do not support Ethernet loopback (external and internal) on Layer2 VPN bridge-domain.

Configure Ethernet Data Plane Loopback

This section describes how you can configure Ethernet Data Plane Loopback on physical interface and sub-interface. Configuring Ethernet Data Plane Loopback involves these steps:

- Configuring Ethernet Data Plane External Loopback
- Starting an Ethernet Data Plane Loopback Session

Configuration Example

```
/* Configuring Ethernet Data Plane External Loopback */

/* On physical interface */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface tenGigE 0/0/0/0 l2transport
RP/0/RSP0/CPU0:router((config-if-l2)# ethernet loopback permit external

/* Starting an Ethernet Data Plane Loopback Session  */

RP/0/RSP0/CPU0:router# ethernet loopback start local interface tenGigE 0/0/0/0 external
source mac-address 0000.0000.0001 destination mac-address 0000.0000.0002 cos 5 timeout none


/* On physical sub-interface */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface tenGigE 0/2/0/0/0.1 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router((config-if-l2)# ethernet loopback permit external

/* Starting an Ethernet Data Plane Loopback Session  */

RP/0/RSP0/CPU0:router# ethernet loopback start local interface tenGigE 0/2/0/0/0.1 external
 source mac-address 0000.0000.0001 destination mac-address 0000.0000.0002 cos 5 timeout
none

/* Configuring Ethernet Data Plane Internal Loopback */

/* On physical interface

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface tenGigE 0/0/0/1 l2transport
RP/0/RSP0/CPU0:router((config-if-l2)# ethernet loopback permit internal

/* Starting an Ethernet Data Plane Loopback Session  */

RP/0/RSP0/CPU0:router# ethernet loopback start local interface tenGigE 0/0/0/1 internal
source mac-address 0000.0000.0002 destination mac-address 0000.0000.0003 cos 5 timeout none


/* On physical sub-interface */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface tenGigE 0/2/0/0/0.1 l2transport
```

```
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router(config-if-l2)# ethernet loopback permit internal

/* Starting an Ethernet Data Plane Loopback Session  */

RP/0/RSP0/CPU0:router# ethernet loopback start local interface tenGigE 0/2/0/0/0.1 internal
 source mac-address 0000.0000.0002 destination mac-address 0000.0000.0003 cos 5 timeout
none



/* Stopping an Ethernet Data Plane Loopback Session */

RP/0/RSP0/CPU0:router# ethernet loopback stop local interface tenGigE 0/0/0/0 id 1
RP/0/RSP0/CPU0:router# ethernet loopback stop local interface tenGigE 0/0/0/1 id 2
RP/0/RSP0/CPU0:router# ethernet loopback stop local interface tenGigE 0/2/0/0/0.1 id 1
```

Similarly, you can configure the Ethernet Data Plane Loopback session for bundle interface and bundle sub-interface.

Ethernet loopback works even after SSO.

Running Configuration

This section shows Ethernet Data Plane Loopback running configuration.

```
/* External Loopback */

/* On physical interface */

configure
 interface interface tenGigE 0/0/0/0 l2transport
  ethernet loopback permit external
 !

/* On physical sub-interface */

configure
 interface interface tenGigE 0/2/0/0/0.1 l2transport
  encapsulation dot1q 100
  ethernet loopback permit external
 !



/* Internal Loopback */

/* On physical interface */

configure
 interface interface tenGigE 0/0/0/1 l2transport
  ethernet loopback permit internal
 !

/* On physical sub-interface */

configure
 interface interface tenGigE 0/2/0/0/0.1 l2transport
  encapsulation dot1q 100
  ethernet loopback permit internal
 !
```

Verification

The following example displays the loopback capabilities per interface. The output shows internal loopback has been permitted on Ten Gigabit Ethernet 0/0/0/1 interface and external loopback has been permitted on Ten Gigabit Ethernet 0/0/0/0 interface.

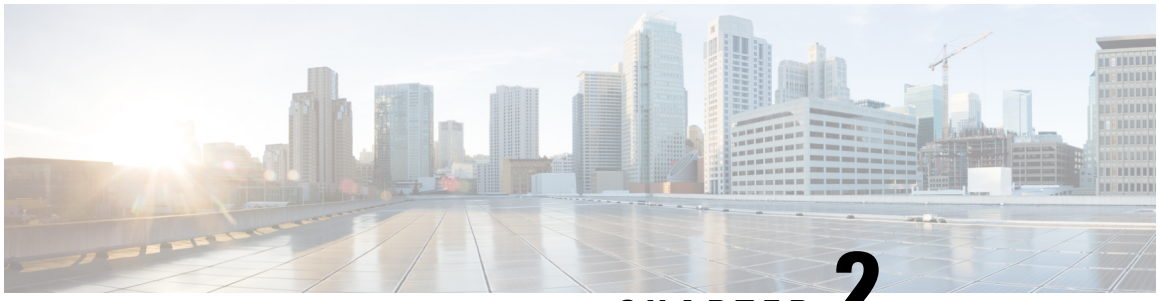
```
RP/0/RSP0/CPU0:router# show ethernet loopback permitted

-------------------------------------------------------------------------------
Interface                              Dot1q(s)                       Direction
-------------------------------------------------------------------------------
tenGigE 0/0/0/1.1                       100                            Internal
tenGigE 0/0/0/0.1                       100                            External
-------------------------------------------------------------------------------


/* This example shows all active sessions on the router */

RP/0/RSP0/CPU0:router# show ethernet loopback active
Thu Jul 20 11:00:57.864 UTC
Local: TenGigE0/0/0/0.1, ID 1
============================================
Direction:                          External
Time out:                               None
Time left:                                 -
Status:                               Active
Filters:
  Dot1Q:                                 Any
  Second-dot1Q:                          Any
  Source MAC Address:                    Any
  Destination MAC Address:               Any
  Class of Service:                      Any
Local: TenGigE0/0/0/0.1, ID 2
============================================
Direction:                          External
Time out:                               None
Time left:                                 -
Status:                               Active
Filters:
  Dot1Q:                                 Any
  Second-dot1Q:                          Any
  Source MAC Address:         0000.0000.0001
  Destination MAC Address:    0000.0000.0002
  Class of Service:                        5
```

Related Topics

- Ethernet Data Plane Loopback, on page 4

Associated Commands

- ethernet loopback
- show ethernet loopback



CHAPTER 2

Configure Layer 2 Access Control Lists

This chapter introduces you to Layer 2 Access Control Lists and describe how you can configure the Layer 2 access control lists.

- Layer 2 Access Control Lists, on page 9
- Prerequisites for Configuring Layer 2 Access Control Lists, on page 9
- Layer 2 Access Control Lists Feature Highlights, on page 10
- Purpose of Layer 2 Access Control Lists, on page 10
- How a Layer 2 Access Control List Works, on page 10
- Layer 2 Access Control List Process and Rules, on page 10
- Create Layer 2 Access Control List, on page 11
- Restrictions for Configuring Layer 2 Access Control Lists, on page 11
- Configuration, on page 11

Layer 2 Access Control Lists

An Ethernet services access control lists (ACLs) consist of one or more access control entries (ACE) that collectively define the Layer 2 network traffic profile. This profile can then be referenced by Cisco IOS XR software features. Each Ethernet services ACL includes an action element (permit or deny) based on criteria such as source and destination address, Class of Service (CoS), ether-type, or 802.1ad DEI.

Layer 2 ACLs are supported on ingress traffic only. Layer 2 ACLs are not supported on egress traffic.

Layer 2 access control lists are also known as Ethernet services control access lists.

Prerequisites for Configuring Layer 2 Access Control Lists

This prerequisite applies to configuring the access control lists and prefix lists:

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Layer 2 Access Control Lists Feature Highlights

Layer 2 access control lists have these feature highlights:

- The ability to clear counters for an access list using a specific sequence number.
- The ability to copy the contents of an existing access list to another access list.
- Allows users to apply sequence numbers to permit or deny statements.
- Layer 2 ACLs can be applied on interfaces, VLAN subinterfaces, bundle-Ethernet interfaces, bundle subinterfaces with L2 transport. Atomic replacement of Layer 2 ACLs is supported on these physical and bundle interfaces.

Purpose of Layer 2 Access Control Lists

Layer 2 access control lists perform packet filtering to control which packets move through the network and where. Such controls help to limit incoming and outgoing network traffic and restrict the access of users and devices to the network at the port level.

How a Layer 2 Access Control List Works

A Layer 2 access control list is a sequential list consisting of permit and deny statements that apply to Layer 2 configurations. The access list has a name by which it is referenced.

An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control Layer 2 traffic arriving at the router, but not traffic originating at the router and leaving the router.

Layer 2 Access Control List Process and Rules

Use this process and rules when configuring Layer 2 access control list:

- The software tests the source or destination address of each packet being filtered against the conditions in the access list, one condition (permit or deny statement) at a time.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the remaining statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies the address or protocol, the software discards the packet.
- If no conditions match, the software drops the packet because each access list ends with an unwritten or implicit deny statement. That is, if the packet has not been permitted or denied by the time it was tested against each statement, it is denied.

- The access list should contain at least one permit statement or else all packets are denied.
- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same permit or deny statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.
- Inbound access lists process packets arriving at the router. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, permit means continue to process the packet after receiving it on an inbound interface; deny means discard the packet.
- An access list can not be removed if that access list is being applied by an access group in use. To remove an access list, remove the access group that is referencing the access list and then remove the access list.
- An access list must exist before you can use the **ethernet-services access-group** command.

Create Layer 2 Access Control List

Consider these when creating a Layer 2 access control list:

- Create the access list before applying it to an interface.
- Organize your access list so that more specific references appear before more general ones.

Restrictions for Configuring Layer 2 Access Control Lists

These restrictions apply to configuring Layer 2 access control lists:

- Layer 2 access control list is not supported, if the destination address is a BVI MAC address.
- Layer 2 access control lists are not supported over management interfaces.
- NetIO (software slow path) is not supported for Layer 2 access control lists.
- Layer 2 access control lists attachment is possible only in ingress direction on an interface.
- Layer 2 access control lists are supported only for the field's L2 source and destination address, Ether Type, Outer VLAN ID, Class of Service (COS), and VLAN Discard Eligibility Indication (DEI). VLAN range is not supported.

Configuration

This section describes how you can configure Layer 2 access control lists.

```
Router# configure
Router(config)# ethernet-services access-list es_acl_1
Router(config-es-acl)# deny 00ff.eedd.0010 ff00.0000.00ff 0000.0100.0001 0000.0000.ffff
Router(config-es-acl)# permit host 000a.000b.000c host 00aa.ab99.1122 cos 1 dei
Router(config-es-acl)# deny host 000a.000b.000c host 00aa.dc11.ba99 cos 7 dei
Router(config-es-acl)# commit
```

```
Router(config)# interface tengige0/0/0/4
Router(config-if)# l2transport
Router(config-if-l2)# commit
Router(config-if-l2)# exit
Router(config-if)# ethernet-services access-group es_acl_1 ingress
Router(config-if)# commit
```

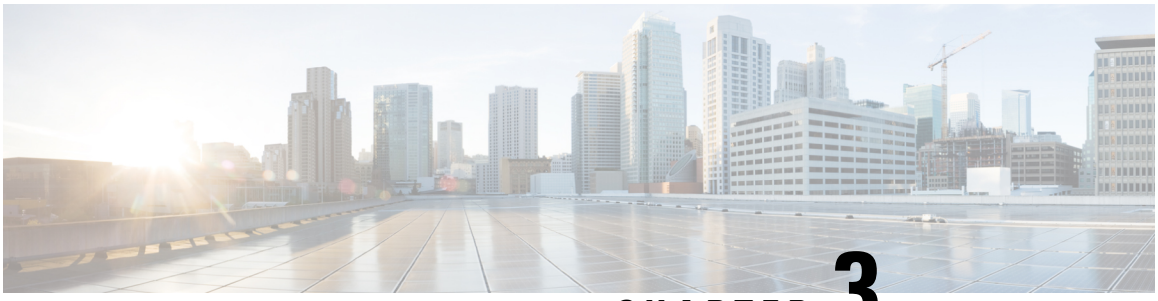
Running Configuration

```
!
Configure
ethernet-services access-list es_acl_1
10 deny 00ff.eedd.0000 ff00.0000.00ff 0000.0100.0000 0000.0000.ffff
20 permit host 000a.000b.000c host 00aa.ab99.1122 cos 1 dei
30 deny host 000a.000b.000c host 00aa.dc11.ba99 cos 7 dei
!
```

Verification

Verify that you have configured Layer 2 access control lists.

```
/* Verify the Layer 2 access control lists configuration */
Router# show access-lists ethernet-services es_acl_1  hardware ingress location 0/0/CPU0
Fri Oct 21 09:39:52.904 UTC
ethernet-services access-list es_acl_1
10 deny 00ff.eedd.0000 ff00.0000.00ff 0000.0100.0000 0000.0000.ffff (2051 matches)
20 permit host 000a.000b.000c host 00aa.ab99.1122 cos 1 dei
30 deny host 000a.000b.000c host 00aa.dc11.ba99 cos 7 dei (2050 matches)
```



CHAPTER 3

Configure Virtual LANs in Layer 2 VPNs

The Layer 2 Virtual Private Network (L2VPN) feature enables Service Providers (SPs) to provide L2 services to geographically disparate customer sites.

A virtual local area network (VLAN) is a group of devices on one or more LANs that are configured so that they can communicate as if they were attached to the same wire, when in fact they are located on a number of different LAN segments. The IEEE's 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames.

VLANs are very useful for user and host management, bandwidth allocation, and resource optimization. Using VLANs addresses the problem of breaking large networks into smaller parts so that broadcast and multicast traffic does not consume more bandwidth than necessary. VLANs also provide a higher level of security between segments of internal networks.

The 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames. Cisco IOS XR software supports VLAN sub-interface configuration on Gigabit Ethernet and 10-Gigabit Ethernet interfaces.

The configuration model for configuring VLAN Attachment Circuits (ACs) is similar to the model used for configuring basic VLANs, where the user first creates a VLAN sub-interface, and then configures that VLAN in sub-interface configuration mode. To create an Attachment Circuit, you need to include the **l2transport** keyword in the **interface** command string to specify that the interface is a L2 interface.

VLAN ACs support the following modes of L2VPN operation:

- Basic Dot1Q Attachment Circuit—The Attachment Circuit covers all frames that are received and sent with a specific VLAN tag.
- QinQ Attachment Circuit—The Attachment Circuit covers all frames received and sent with a specific outer VLAN tag and a specific inner VLAN tag. QinQ is an extension to Dot1Q that uses a stack of two tags.

Encapsulation

Encapsulation defines the matching criteria that maps a VLAN, a range of VLANs. Different types of encapsulations are default, dot1q, dot1ad. The following are the supported encapsulation types:

- **encapsulation default**: Configures the default service instance on a port.
- **encapsulation dot1q vlan-id** : Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance.

- **encapsulation dot1ad vlan-id** : Defines the matching criteria to map 802.1ad frames ingress on an interface to the appropriate service instance.
- **encapsulation dot1q second-dot1q**: Defines the matching criteria to map Q-in-Q ingress frames on an interface to the appropriate service instance.
- **encapsulation dot1ad dot1q**: Defines the matching criteria to be used in order to map single-tagged 802.1ad frames ingress on an interface to the appropriate service instance.

Restrictions and Limitations

To configure VLANs for Layer 2 VPNs, the following restrictions are applicable.

- In a point-to-point connection, the two Attachment Circuits do not have to be of the same type. For example, a port mode Ethernet Attachment Circuit can be connected to a Dot1Q Ethernet Attachment Circuit.
- Pseudowires can run in VLAN mode or in port mode. A pseudowire running in VLAN mode always carries Dot1Q or Dot1ad tag(s), while a pseudowire running in port mode may or may NOT carry tags. To connect these different types of circuits, popping, pushing, and rewriting tags is required.
- The Attachment Circuits on either side of an MPLS pseudowire can be of different types. In this case, the appropriate conversion is carried out at one or both ends of the Attachment Circuit to pseudowire connection.
- When receiving single or double Dot1Q tagged traffic on an L2VPN pseudowire, the egress rewrite action Push 1 configured in an attachment circuit is not supported. The egress rewrite action Push 1 configured in an attachment circuit is supported only for untagged traffic received on an L2VPN pseudowire.

- Configure VLAN Subinterfaces, on page 15
- Introduction to Ethernet Flow Point, on page 19
- Configure VLAN Header Rewrite, on page 24
- Rewrite of Priority Tag, on page 36

Configure VLAN Subinterfaces

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Increased VLAN-IDs per VLAN list	Release 7.8.1	From this release, you can configure up to 64 VLAN-IDs per VLAN list. Previously, the number of VLAN-IDs supported were only up to 9, per VLAN list. The enhanced VLAN-IDs help to add more number of customers in an Ethernet network. Use the encapsulation list-extended dot1q command, to configure up to 64 VLAN-IDs.
VLAN List	Release 7.4.1	VLANs separated by a comma are called VLAN lists. This feature allows you to configure a VLAN list on the L2 subinterface. VLAN-IDs of up to 9 are supported, per VLAN list. This feature overrides any limit set on the number of customers that can be supported in an Ethernet network.

Subinterfaces are logical interfaces created on a hardware interface. These software-defined interfaces allow for segregation of traffic into separate logical channels on a single hardware interface as well as allowing for better utilization of the available bandwidth on the physical interface.

Subinterfaces are distinguished from one another by adding an extension on the end of the interface name and designation. For instance, the Ethernet subinterface 23 on the physical interface designated TenGigE 0/1/0/0 would be indicated by TenGigE 0/1/0/0.23.

Before a subinterface is allowed to pass traffic, it must have a valid tagging protocol encapsulation and VLAN identifier assigned. All Ethernet subinterfaces always default to the 802.1Q VLAN encapsulation. However, the VLAN identifier must be explicitly defined.

The subinterface Maximum Transmission Unit (MTU) is inherited from the physical interface with 4 bytes allowed for the 802.1Q VLAN tag.

The following modes of VLAN subinterface configuration are supported:

- Basic dot1q Attachment Circuit
- Basic dot1ad Attachment Circuit
- Q-in-Q Attachment Circuit

To configure a basic dot1q Attachment Circuit, use this encapsulation mode:

encapsulation dot1q *vlan extra-id*

From Release 7.8.1, use **encapsulation list-extended dot1q** command to extend the number of VLAN IDs or VLAN ranges, to configure up to 64 VLAN IDs or VLAN ranges per VLAN list. The VLAN list is supported for both inner and outer VLAN IDs.

encapsulation list-extended dot1q *vlan-id*

To configure a basic dot1ad Attachment Circuit, use this encapsulation mode:

encapsulation dot1ad *vlan-id*

To configure a Q-in-Q Attachment Circuit, use the following encapsulation modes:

- **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id*
- **encapsulation dot1ad** *vlan-id* **dot1q** *vlan-id*

From Release 7.4.1, VLAN list is supported for both inner and outer VLAN IDs. The following is the example to show the supported VLAN lists on L2 subinterface:

```
Router#configure
Router(config)#interface TenGigE 0/0/0/1.101 l2transport
Router(config-subif)#encapsulation dot1q 10,11,12,13,14,15,16,17,untagged
```

From Release 7.8.1, use **encapsulation list-extended dot1q** command to extend the number of VLAN IDs or VLAN ranges, to configure up to 64 VLAN IDs or VLAN ranges per VLAN list. The following is the example to show the supported VLAN lists on an L2 subinterface:

```
Router#configure
Router(config)#interface TenGigE 0/0/0/1.102 l2transport
Router(config-subif)#encapsulation list-extended dot1q 10,11,12,13,14,15,16,17,18,19,20,21
```

If you're moving from any old commands like, **encapsulation dot1q**, or **encapsulation dot1q priority-tagged**, or **encapsulation default**, or from any other old commands, to the **encapsulation list-extended** command, then **no encapsulation** command should precede the **encapsulation list-extended** command as shown in the following example.

If you're moving from the **encapsulation list-extended** command to any of the old commands, then **no encapsulation list-extended** command should precede the old command as shown in the following example.

```
Router(config-subif)#no encapsulation list-extended
Router(config-subif)#encapsulation default
Router(config-subif)#commit
```

- BVI with Double-Tagged AC—You can configure the attachment circuit (AC) with double-VLAN tag encapsulation on the bridge-group virtual interface (BVI). You must specify the rewrite ingress pop 2 symmetric option when you configure the AC on the BVI with double-VLAN tag encapsulation.

Restrictions and Limitations

To configure VLAN subinterface, the following restrictions are applicable.

- At least 64 VLAN-IDs in a VLAN list is required to overcome the limitation of only 9 VLAN ranges per NPU.
- If 0x9100/0x9200 is configured as tunneling ether-type, then dot1ad (0x88a8) encapsulation isn't supported.
- If any subinterface is already configured under a main interface, modifying the tunneling ether-type isn't supported.
- Following limitations are applicable to both outer and inner VLAN ranges:
  - 32 unique VLAN ranges are supported per NPU.
  - The overlap between outer VLAN ranges on subinterfaces of the same physical or bundle main interface isn't supported. A subinterface with a single VLAN tag that falls into a range configured on another subinterface of the same physical or bundle main interface is also considered an overlap.
  - The overlap between inner VLAN ranges on subinterfaces of the same NPU isn't supported.
  - Range 'any' doesn't result in explicit programming of a VLAN range in hardware and therefore doesn't count against the configured ranges.

Configuration Example

Configuring a VLAN subinterface involves:

- Creating a Ten Gigabit Ethernet subinterface
- Enabling L2 transport mode on the interface
- Defining the matching criteria (encapsulation mode) to be used in order to map ingress frames on an interface to the appropriate service instance.

Configuration of Basic dot1q Attachment Circuit

```
Router# configure
Router(config)# interface TenGigE 0/0/0/10.1 l2transport
Router(config-if)# encapsulation dot1q 10 exact
Router(config-if)# no shutdown

Router# configure
Router(config)#interface TenGigE 0/0/0/1.101 l2transport
Router(config-subif)#encapsulation list-extended dot1q
66-67,68-69,70-71,118-119,120-121,122-123,229,230,231
```

Running Configuration

```
configure
 interface TenGigE 0/0/0/10.1
  l2transport
  encapsulation dot1q 10 exact
 !
!

Configure
```

```
 interface TenGigE 0/0/0/1.101
  l2transport
  encapsulation list-extended dot1q 66-67,68-69,70-71,118-119,120-121,122-123,229,230,231
```

Verification

Verify that the VLAN subinterface is active:

```
Router# show interfaces TenGigE 0/0/0/10.1

...
TenGigE0/0/0/10.1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 0011.1aac.a05a
  Layer 2 Transport Mode
  MTU 1518 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation 802.1Q Virtual LAN,
    Outer Match: Dot1Q VLAN 10
    Ethertype Any, MAC Match src any, dest any
  loopback not set,
...


Router#show interfaces TenGigE 0/0/0/1.101
TenGigabitEthernet0/0/0/1.101 is down, line protocol is down
   Interface state transitions: 0
   Hardware is VLAN sub-interface(s), address is 008a.9678.0c04
   Layer 2 Transport Mode
   MTU 1518 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
      reliability Unknown, txload Unknown, rxload Unknown
   Encapsulation 802.1Q Virtual LAN,
      Outer Match: Dot1Q VLAN 66-67,68-69,70-71,118-119,120-121,122-123,229,230,231
      Ethertype Any, MAC Match src any, dest any
   loopback not set,
   Last input never, output never
   Last clearing of "show interface" counters never
      0 packets input, 0 bytes
      0 input drops, 0 queue drops, 0 input errors
      0 packets output, 0 bytes
      0 output drops, 0 queue drops, 0 output errors
```

Associated Commands

- encapsulation dot1ad dot1q
- encapsulation dot1q
- encapsulation dot1q second-dot1q
- l2transport (Ethernet)
- encapsulation dot1ad

Introduction to Ethernet Flow Point

An Ethernet Flow Point (EFP) is a Layer 2 logical sub-interface used to classify traffic under a physical or a bundle interface. An EFP is defined by a set of filters (a set of entries) that are applied to all the ingress traffic to classify the frames that belong to a particular EFP. Each entry usually contains 0, 1 or 2 VLAN tags. You can specify a VLAN or QinQ tagging to match against on ingress. A packet that starts with the same tags as an entry in the filter is said to match the filter; if the start of the packet does not correspond to any entry in the filter, then the packet does not match the filter.

All traffic on ingress are processed by that EFP if a match occurs, and this can in turn change VLAN IDs, add or remove VLAN tags, and change ethertypes. After the frames are matched to a particular EFP, any appropriate feature (such as, any frame manipulations specified by the configuration as well as things such as QoS and ACLs) can be applied.

The benefits of EFP include:

- Identifying all frames that belong to a particular flow on a given interface
- Performing VLAN header rewrites

  (See, Configure VLAN Header Rewrite, on page 24)
- Adding features to the identified frames
- Optionally defining how to forward the identified frames in the data path

Limitations of EFP

Egress EFP filtering is not supported on Cisco IOS XR.

Identify Frames of an EFP

The EFP identifies frames belonging to a particular flow on a given port, independent of their Ethernet encapsulation. An EFP can flexibly map frames into a flow or EFP based on the fields in the frame header. The frames can be matched to an EFP using VLAN tags.

The frames can’t be matched to an EFP through this:

- Any information outside the outermost Ethernet frame header and its associated tags such as
  - IPv4, IPv6, or MPLS tag header data
  - C-DMAC, C-SMAC, or C-VLAN

VLAN Tag Identification

Below table describes the different encapsulation types and the EFP identifier corresponding to each.

Encapsulation Type	EFP Identifier
Single tagged frames	802.1Q customer-tagged Ethernet frames

Encapsulation Type	EFP Identifier
Double tagged frames Double tagged frames can be of the following types: • Single range • Range-in-Q • Q-in-Range	802.1Q (ethertype 0x9100) double tagged frames 802.1ad (ethertype 0x9200) double tagged frames • In single range, a range of VLAN IDs can be added for an EFP. • In Range-in-Q, a range of outer VLAN IDs can have a single inner VLAN ID. • In Q-in-Range, a single outer VLAN ID can have a range of inner VLAN IDs.

You can use wildcards while defining frames that map to a given EFP. EFPs can distinguish flows based on a single VLAN tag, a stack of VLAN tags or a combination of both (VLAN stack with wildcards). It provides the EFP model, a flexibility of being encapsulation agnostic, and allows it to be extensible as new tagging or tunneling schemes are added.

Single Tagged VLAN Range Support for Double Tagged Frames

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Single Tagged VLAN Range Support for Double Tagged Frames	Release 7.8.1	From this release, L2 subinterface configuration with single tagged VLAN range can be matched with the double tagged frames. Previously, the packet matching was done only with single VLAN ID and the double tagged packets were dropped. With single tagged VLAN range support for double tagged frames, the traffic can reach the VLAN destination safely.

Starting from Cisco IOS XR Release 7.8.1, L2 subinterfaces with single tagged VLAN range, can be configured to match the double tagged frames. Prior to this release, if you have a single VLAN ID configured using the command **encapsulation dot1q** *1* to define the matching criteria on a subinterface, then it only matches single tagged packets with VLAN ID as 1 and double tagged packets with outer VLAN ID as 1. VLAN range configuration is not supported, for example **encapsulation dot1q** *1-3*.

If VLAN range is configured, then the configuration matches only single tagged packets and drops the double tagged packets even if the outer VLAN is within the specified range.

Configuring Basic Dot1q Attachment Circuit with VLAN Range

The following configuration shows how to configure a basic dot1q Attachment Circuit, using this encapsulation mode with outer VLAN range between (1 and 3) and inner tag with matching single tagged frames. Irrespective of the packets, either it's single tagged or double tagged, the check is done only on the outer VLAN header.

```
Router#configure
Router(config)#interface TenGigE 0/0/0/0.1 l2transport
Router(config-if)#encapsulation dot1q 1-3
```

Use the Show interfaces command to display the operational information for Ethernet interfaces.

Apply Features

After the frames are matched to a particular EFP, any appropriate features can be applied. In this context, “features” means any frame manipulations specified by the configuration as well as things such as QoS and ACLs. The Ethernet infrastructure provides an appropriate interface to allow the feature owners to apply their features to an EFP. Hence, IM interface handles are used to represent EFPs, allowing feature owners to manage their features on EFPs in the same way the features are managed on regular interfaces or sub-interfaces.

The only L2 features that can be applied on an EFP that is part of the Ethernet infrastructure are the L2 header encapsulation modifications. The L2 features are described in this section.

Encapsulation Modifications

EFP supports these L2 header encapsulation modifications on both ingress and egress:

- Push 1 or 2 VLAN tags
- Pop 1 or 2 VLAN tags

Note This modification can only pop tags that are matched as part of the EFP.

- Rewrite 1 or 2 VLAN tags:
  - Rewrite outer tag
  - Rewrite outer 2 tags
  - Rewrite outer tag and push an additional tag

  For each of the VLAN ID manipulations, these can be specified:
- The VLAN tag type, that is, C-VLAN, S-VLAN, or I-TAG. The ethertype of the 802.1Q C-VLAN tag is defined by the dot1q tunneling type command.
- The VLAN ID. 0 can be specified for an outer VLAN tag to generate a priority-tagged frame.

Note For tag rewrites, the CoS bits from the previous tag should be preserved in the same way as the DEI bit for 802.1ad encapsulated frames.

Define Data-Forwarding Behavior

The EFP can be used to designate the frames belonging to a particular Ethernet flow forwarded in the data path. These forwarding cases are supported for EFPs in Cisco IOS XR software:

- L2 Switched Service (Bridging)—The EFP is mapped to a bridge domain, where frames are switched based on their destination MAC address. This includes multipoint services:
  - Ethernet to Ethernet Bridging
  - Multipoint Layer 2 Services
- L2 Stitched Service (AC to AC xconnect)—This covers point-to-point L2 associations that are statically established and do not require a MAC address lookup.
  - Ethernet to Ethernet Local Switching—The EFP is mapped to an S-VLAN either on the same port or on another port. The S-VLANs can be identical or different.
- Tunneled Service (xconnect)—The EFP is mapped to a Layer 3 tunnel. This covers point-to-point services, such as EoMPLS.

Ethernet Flow Points Visibility

EFP Visibility feature enables you to configure multiple VLANs only when IGMP snooping is enabled and multiple VLANs and sub-interfaces of same port is configured under the same bridge domain.

An Ethernet flow point (EFP) service instance is a logical interface that connects a bridge domain to a physical port or to an EtherChannel group. A VLAN tag is used to identify the EFP.

Earlier only one EFP was allowed per bridge-domain. With EFP visibility feature, you can configure a maximum of:

- 600 EFPs per bridge-domain.
- 100 EFPs per port.

Irrespective of number of ports available, you have flexibility to add more EFPs in one bridge group.

Configuring EFP Visibility

This example shows how to configure IGMP snooping on VLAN interfaces under a bridge domain with multiple EFPs.

```
/* Configure two IGMP Snooping profiles */
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# igmp snooping profile 1
RP/0/RP0/CPU0:router(config-igmp-snooping-profile)# exit
RP/0/RP0/CPU0:router(config)# igmp snooping profile 2
RP/0/RP0/CPU0:router(config-igmp-snooping-profile)#commit

 !

/* Configure VLAN interfaces for L2 transport */
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface gigabitEthernet 0/8/0/8
RP/0/RP0/CPU0:router(config-if)# bundle id 2 mode on
RP/0/RP0/CPU0:router(config-if)# no shut
RP/0/RP0/CPU0:router(config-if)# exit
RP/0/RP0/CPU0:router(config)# interface gigabitEthernet 0/8/0/9
RP/0/RP0/CPU0:router(config-if)# bundle id 3 mode on
RP/0/RP0/CPU0:router(config-if)# no shut
RP/0/RP0/CPU0:router(config-if)# exit

RP/0/RP0/CPU0:router(config)# interface Bundle-Ether2
RP/0/RP0/CPU0:router(config-if)# exit
```

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether3
RP/0/RP0/CPU0:router(config-if)# exit

RP/0/RP0/CPU0:router(config)# interface Bundle-Ether2.2 l2transport
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 2
RP/0/RP0/CPU0:router(config-subif)# rewrite ingress tag pop 1 symmetric
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether2.3 l2transport
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 3
RP/0/RP0/CPU0:router(config-subif)# rewrite ingress tag pop 1 symmetric
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether2.4 l2transport
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 4
RP/0/RP0/CPU0:router(config-subif)# rewrite ingress tag pop 1 symmetric
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether2.5 l2transport
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 5
RP/0/RP0/CPU0:router(config-subif)# rewrite ingress tag pop 1 symmetric
RP/0/RP0/CPU0:router(config-subif)# exit

RP/0/RP0/CPU0:router(config)# interface Bundle-Ether3.2 l2transport
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 2
RP/0/RP0/CPU0:router(config-subif)# rewrite ingress tag pop 1 symmetric
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether3.3 l2transport
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 3
RP/0/RP0/CPU0:router(config-subif)# rewrite ingress tag pop 1 symmetric
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# commit

/* Attach a profile and add interfaces to the bridge domain.
Attach a profile to one of the interfaces. The other interface
inherits IGMP snooping configuration attributes from the bridge domain profile */

RP/0/RP0/CPU0:router(config)#l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#bridge group VLAN2
RP/0/RP0/CPU0:router(config-l2vpn-bg)#bridge-domain VLAN2
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#efp-visibility
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#igmp snooping profile 1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#interface bundle-Ether2.2
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#interface bundle-Ether 2.3
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#interface bundle-Ether 2.4
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#interface bundle-Ether 2.5
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg)#bridge-domain vlan3
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#efp-visibility
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#igmp snooping profile 2
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#interface bundle-Ether3.2
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#interface bundle-Ether 3.3
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#routed interface bvi2
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-bvi)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#evi 2
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-evi)#exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#commit
```

Verification

Verify the configured bridge ports:

```
RP/0/RP0/CPU0:router# show igmp snooping port
                     Bridge Domain VLAN2:VLAN2

                                                      State
Port                                                  Oper  STP  Red   #Grps   #SGs
----                                                  ----  ---  ---   -----   ----
BVI2                                                  Up    -    -       0      0
Bundle-Ether2.2                                       Up    -    -     100      0
Bundle-Ether2.3                                       Up    -    -     100      0
Bundle-Ether2.4                                       Up    -    -     100      0
Bundle-Ether2.5                                       Up    -    -     100      0

 Bridge Domain VLAN3:VLAN3

                                                      State
Port                                                  Oper  STP  Red   #Grps   #SGs
----                                                  ----  ---  ---   -----   ----
BVI3                                                  Up    -    -       0      0
Bundle-Ether3.2                                       Up    -    -     100      0
Bundle-Ether3.3                                       Up    -    -     100      0
```

In the above output verify the status of BVI and EFPs are **Up**, and the **#Grps** and **#SG** show the correct number of IGMP join received.

Configure VLAN Header Rewrite

EFP supports the following VLAN header rewrites on both ingress and egress ports:

- Push 1 VLAN tag
- Pop 1 VLAN tag

Note This rewrite can only pop tags that are matched as part of the EFP.

- Translate 1 or 2 VLAN tags:
  - Translate 1-to-1 tag: Translates the outermost tag to another tag
  - Translate 1-to-2 tags: Translates the outermost tag to two tags
  - Translate 2-to-2 tags: Translates the outermost two tags to two other tags

Various combinations of ingress, egress VLAN rewrites with corresponding tag actions during ingress and egress VLAN translation, are listed in the following sections:

Limitations

The limitations for VLAN header rewrites are as follows:

- Push 1 is not supported for dot1ad configuration.
- Push 2 is supported only on:

  - Untagged EFP
  - Dot1q EFP with **exact** configuration statement

- Translate 1 to 1 is not supported for dot1ad configuration.
- Translate 1 to 2 is not supported with **dot1q tunneling ethertype** configuration statement.
- Translate 2 to 1 is not supported.
-
- When a single-tag range is used, double tagged traffic does not match.

  For example, in the following configuration, dot1q 2-6 is the outer tag.

```
Router#configure
Router(config)# interface GigabitEthernet0/0/0/0.0 l2transport
Router(config-if)# encapsulation dot1q 2-6
```

  - An incoming packet with an outer tag of 2 and ANY inner tag does not match. For example, the double tag packet of outer tag 2 and inner tag 1 is not be accepted on the interface 0/0/0/0.0.
  - But, an incoming packet with a single tag of 2 is accepted. For example, the single tag packet of outer tag between 2 to 6 is accepted on the interface 0/0/0/0.0.

Configuration Example

This topic covers VLAN header rewrites on various attachment circuits, such as:

- L2 single-tagged sub-interface
- L2 double-tagged sub-interface

Configuring VLAN header rewrite involves:

- Creating a TenGigabit Ethernet sub-interface
- Enabling L2 transport mode on the interface
- Defining the matching criteria (encapsulation mode) to be used in order to map single-tagged frames ingress on an interface to the appropriate service instance
- Specifying the encapsulation adjustment that is to be performed on the ingress frame

Configuration of VLAN Header Rewrite (single-tagged sub-interface)

```
Router# configure
Router(config)# interface TenGigE 0/0/0/10.1 l2transport
Router(config-if)# encapsulation dot1q 10 exact
Router(config-if)# rewrite ingress tag push dot1q 20 symmteric
```

Running Configuration

```
/* Configuration without rewrite */
```

```
configure
 interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10 exact
 !
!

/* Configuration with rewrite */

/* PUSH 1 */
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
 rewrite ingress tag push dot1q 20 symmteric
 !
!

/* POP 1 */
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
  rewrite ingress tag pop 1
 !
!

/* TRANSLATE 1-1 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
  rewrite ingress tag translate 1-to-1 dot1q 20
 !
!

/* TRANSLATE 1-2 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
  rewrite ingress tag translate 1-to-2 dot1q 20 second-dot1q 30
 !
!
```

Running Configuration (VLAN header rewrite on double-tagged sub-interface)

```
/* Configuration without rewrite */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
 !
!

/* Configuration with rewrite */

/* PUSH 1 */
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag push dot1q 20 symmteric
 !
!

/* TRANSLATE 1-1 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 1-to-1 dot1q 20
 !
```

```
!

/* TRANSLATE 1-2 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 1-to-2 dot1q 20 second-dot1q 30
 !
!

/* TRANSLATE 2-2 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 2-to-2 dot1q 20 second-dot1q 30
 !
!
```

Associated Commands

- encapsulation dot1ad dot1q
- encapsulation dot1q
- encapsulation dot1q second-dot1q
- l2transport (Ethernet)
- rewrite ingress tag

Valid Ingress Rewrite Actions

Table 3: Valid Ingress Rewrite Actions

Interface Configuration	Ingress Rewrite Action
dot1q	No rewrite
dot1q	Pop 1
dot1q	Push 1
dot1q	Push 2
dot1q	Translate 1 to 1
dot1q	Translate 1 to 2
QinQ	No rewrite
QinQ	Pop 1
QinQ	Push 1
QinQ	Translate 1 to 1
QinQ	Translate 1 to 2

Interface Configuration	Ingress Rewrite Action
QinQ	Translate 2 to 2
Untagged	No rewrite
Untagged	Push 1
Untagged	Push 2

The following notations are used for the rewrite actions mentioned in the table:

- Translate 1-to-1 tag: Translates the outermost tag to another tag.
- Translate 1-to-2 tags: Translates the outermost tag to two tags.
- Translate 2-to-2 tags: Translates the outermost two tags to two other tags.

Valid Ingress-Egress Rewrite Combinations

Table 4: Valid Ingress-Egress Rewrite Combinations

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
dot1q	No rewrite	dot1q	No rewrite
dot1q	No rewrite	dot1q	Pop 1
dot1q	No rewrite	dot1q	Push 1
dot1q	No rewrite	dot1q	Translate 1-to-1
dot1q	Pop 1	dot1q	No rewrite
dot1q	Pop 1	dot1q	Pop 1
dot1q	Push 1	dot1q	No rewrite
dot1q	Push 1	dot1q	Push 1
dot1q	Push 1	dot1q	Push 2
dot1q	Push 1	dot1q	Translate 1-to-1
dot1q	Push 1	dot1q	Translate 1-to-2
dot1q	Push 2 / Translate 1-to-2	dot1q	Push 1
dot1q	Push 2 / Translate 1-to-2	dot1q	Push 2
dot1q	Push 2 / Translate 1-to-2	dot1q	Translate 1-to-2
dot1q	Translate 1-to-1	dot1q	No rewrite

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
dot1q	Translate 1-to-1	dot1q	Push 1
dot1q	Translate 1-to-1	dot1q	Translate 1-to-1
dot1q	No rewrite	dot1q range	No rewrite
dot1q	No rewrite	dot1q range	Push 1
dot1q	Pop 1	dot1q range	No rewrite
dot1q	Push 1	dot1q range	No rewrite
dot1q	Push 1	dot1q range	Push 1
dot1q	Push 1	dot1q range	Push 2
dot1q	Translate 1-to-1	dot1q range	No rewrite
dot1q	Translate 1-to-1	dot1q range	Push 1
dot1q	Translate 1-to-2	dot1q range	Push 1
dot1q	Translate 1-to-2	dot1q range	Push 2
dot1q	No rewrite / Translate 1-to-1	QinQ	No rewrite
dot1q	No rewrite / Translate 1-to-1	QinQ	Pop 1
dot1q	No rewrite / Translate 1-to-1	QinQ	Push 1
dot1q	No rewrite / Translate 1-to-1	QinQ	Translate 1-to-1
dot1q	Pop 1	QinQ	No rewrite
dot1q	Pop 1	QinQ	Pop 1
dot1q	Push 1	QinQ	No rewrite
dot1q	Push 1	QinQ	Pop 1
dot1q	Push 1	QinQ	Push 1
dot1q	Push 1	QinQ	Translate 1-to-1
dot1q	Push 1	QinQ	Translate 1-to-2
dot1q	Push 1	QinQ	Translate 2-to-2
dot1q	Push 2 / Translate 1-to-2	QinQ	No rewrite
dot1q	Push 2 / Translate 1-to-2	QinQ	Push 1
dot1q	Push 2 / Translate 1-to-2	QinQ	Translate 1-to-1

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
dot1q	Push 2 / Translate 1-to-2	QinQ	Translate 1-to-2
dot1q	Push 2 / Translate 1-to-2	QinQ	Translate 2-to-2
dot1q	No rewrite / Translate 1-to-1	QinQ range	No rewrite
dot1q	No rewrite / Translate 1-to-1	QinQ range	Pop 1
dot1q	No rewrite / Translate 1-to-1	QinQ range	Push 1
dot1q	No rewrite / Translate 1-to-1	QinQ range	Translate 1-to-1
dot1q	Pop 1	QinQ range	No rewrite
dot1q	Pop 1	QinQ range	Pop 1
dot1q	Push 1	QinQ range	No rewrite
dot1q	Push 1	QinQ range	Pop 1
dot1q	Push 1	QinQ range	Push 1
dot1q	Push 1	QinQ range	Translate 1-to-1
dot1q	Push 1	QinQ range	Translate 1-to-2
dot1q	Push 2 / Translate 1-to-2	QinQ range	No rewrite
dot1q	Push 2 / Translate 1-to-2	QinQ range	Push 1
dot1q	Push 2 / Translate 1-to-2	QinQ range	Translate 1-to-1
dot1q	Push 2 / Translate 1-to-2	QinQ range	Translate 1-to-2
dot1q	No rewrite	QinQ range	No rewrite
dot1q	No rewrite	Untagged	Push 1
dot1q	Pop 1	Untagged	No rewrite
dot1q	Push 1	Untagged	Push 1
dot1q	Push 1	Untagged	Push 2
dot1q	Push 2	Untagged	Push 2
dot1q	Translate 1-to-1	Untagged	Push 1
dot1q	Translate 1-to-2	Untagged	Push 2
dot1q range	No rewrite	dot1q range	No rewrite
dot1q range	No rewrite	dot1q range	Push 1

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
dot1q range	Push 1	dot1q range	No rewrite
dot1q range	Push 1	dot1q range	Push 1
dot1q range	Push 1	dot1q range	Push 2
dot1q range	Push 2	dot1q range	Push 1
dot1q range	Push 2	dot1q range	Push 2
dot1q range	No rewrite	QinQ	No rewrite
dot1q range	No rewrite	QinQ	Pop 1
dot1q range	No rewrite	QinQ	Push 1
dot1q range	No rewrite	QinQ	Translate 1-to-1
dot1q range	Push 1	QinQ	No rewrite
dot1q range	Push 1	QinQ	Pop 1
dot1q range	Push 1	QinQ	Push 1
dot1q range	Push 1	QinQ	Translate 1-to-1
dot1q range	Push 1	QinQ	Translate 1-to-2
dot1q range	Push 1	QinQ	Translate 2-to-2
dot1q range	Push 2	QinQ	No rewrite
dot1q range	Push 2	QinQ	Push 1
dot1q range	Push 2	QinQ	Translate 1-to-1
dot1q range	Push 2	QinQ	Translate 1-to-2
dot1q range	Push 2	QinQ	Translate 2-to-2
dot1q range	No rewrite	QinQ range / QinAny	No rewrite
dot1q range	No rewrite	QinQ range	Pop 1
dot1q range	No rewrite	QinQ range / QinAny	Push 1
dot1q range	No rewrite	QinQ range / QinAny	Translate 1-to-1

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
dot1q range	Push 1	QinQ range / QinAny	No rewrite
dot1q range	Push 1	QinQ range / QinAny	Pop 1
dot1q range	Push 1	QinQ range	Push 1
dot1q range	Push 1	QinQ range / QinAny	Translate 1-to-1
dot1q range	Push 1	QinQ range / QinAny	Translate 1-to-2
dot1q range	Push 2	QinQ range / QinAny	No rewrite
dot1q range	Push 2	QinQ range / QinAny	Push 1
dot1q range	Push 2	QinQ range / QinAny	Translate 1-to-1
dot1q range	Push 2	QinQ range / QinAny	Translate 1-to-2
dot1q range	No rewrite	Untagged	No rewrite
dot1q range	No rewrite	Untagged	Push 1
dot1q range	Push 1	Untagged	Push 1
dot1q range	Push 1	Untagged	Push 2
dot1q range	Push 2	Untagged	Push 2
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ	No rewrite
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ	Pop 1
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ	Push 1
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ	Translate 1-to-1

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ	Translate 1-to-2
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ	Translate 2-to-2
QinQ	Pop 1	QinQ	No rewrite
QinQ	Pop 1	QinQ	Pop 1
QinQ	Pop 1	QinQ	Push 1
QinQ	Pop 1	QinQ	Translate 1-to-1
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ	No rewrite
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ	Push 1
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ	Translate 1-to-1
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ	Translate 1-to-2
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ	Translate 2-to-2
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	No rewrite
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	Pop 1
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	Push 1
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	Translate 1-to-1
QinQ	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	Translate 1-to-2
QinQ	Pop 1	QinQ range / QinAny	No rewrite
QinQ	Pop 1	QinQ range / QinAny	Pop 1
QinQ	Pop 1	QinQ range / QinAny	Push 1

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
QinQ	Pop 1	QinQ range / QinAny	Translate 1-to-1
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ range / QinAny	No rewrite
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ range / QinAny	Push 1
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ range / QinAny	Translate 1-to-1
QinQ	Translate 1-to-2 / Translate 2-to-2	QinQ range / QinAny	Translate 1-to-2
QinQ	No rewrite	Untagged	No rewrite
QinQ	No rewrite	Untagged	Push 1
QinQ	No rewrite	Untagged	Push 2
QinQ	Pop 1	Untagged	No rewrite
QinQ	Pop 1	Untagged	Push 1
QinQ	Push 1 / Translate 1-to-1	Untagged	Push 1
QinQ	Push 1 / Translate 1-to-1	Untagged	Push 2
QinQ	Translate 1-to-2 / Translate 2-to-2	Untagged	Push 2
QinQ range / QinAny	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	No rewrite
QinQ range / QinAny	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	Pop 1
QinQ range / QinAny	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	Push 1
QinQ range / QinAny	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	Translate 1-to-1
QinQ range / QinAny	No rewrite / push 1 / Translate 1-to-1	QinQ range / QinAny	Translate 1-to-2

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
QinQ range / QinAny	Pop 1	QinQ range / QinAny	No rewrite
QinQ range / QinAny	Pop 1	QinQ range / QinAny	Pop 1
QinQ range / QinAny	Pop 1	QinQ range / QinAny	Push 1
QinQ range	Pop 1	QinQ range	Translate 1-to-1
QinQ range / QinAny	Translate 1-to-2	QinQ range / QinAny	No rewrite
QinQ range / QinAny	Translate 1-to-2	QinQ range / QinAny	Push 1
QinQ range / QinAny	Translate 1-to-2	QinQ range / QinAny	Translate 1-to-1
QinQ range / QinAny	Translate 1-to-2	QinQ range / QinAny	Translate 1-to-2
QinQ range / QinAny	No rewrite	Untagged	No rewrite
QinQ range / QinAny	No rewrite	Untagged	Push 1
QinQ range / QinAny	No rewrite	Untagged	Push 2
QinQ range / QinAny	Pop 1	Untagged	No rewrite
QinQ range / QinAny	Pop 1	Untagged	Push 1
QinQ range / QinAny	Push 1 / Translate 1-to-1	Untagged	Push 1
QinQ range / QinAny	Push 1 / Translate 1-to-1	Untagged	Push 2

Ingress Interface Configuration	Ingress Interface Rewrite Action	Egress Interface Configuration	Egress Interface Rewrite Action
QinQ range / QinAny	Translate 1-to-2	Untagged	Push 2
Untagged	No rewrite	Untagged	No rewrite
Untagged	Push 1	Untagged	Push 1
Untagged	Push 2	Untagged	Push 2

The following notations are used for the rewrite actions mentioned in the table:

- Translate 1-to-1 tag: Translates the outermost tag to another tag
- Translate 1-to-2 tags: Translates the outermost tag to two tags
- Translate 2-to-2 tags: Translates the outermost two tags to two other tags

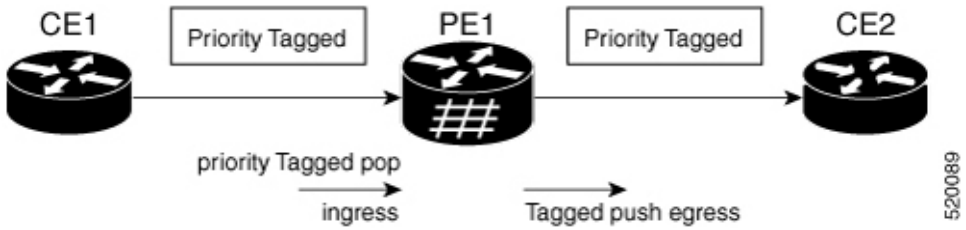
Rewrite of Priority Tag

The Rewrite of Priority Tag feature allows you to configure rewrite tag for a priority-tagged VLAN. This feature removes the priority-tagged VLAN in the ingress direction and adds the priority-tagged VLAN in the egress direction.

You can configure the **rewrite ingress tag symmetric** command for priority-tagged Ethernet Virtual Connections (EVC) on PE1.

This feature supports only rewrite tag pop1 for priority-tag.

Figure 1: Rewrite of Priority Tag



Configure Rewrite of Priority Tag

Perform this task to configure Rewrite of Priority Tag feature.

```
Router#configure
Router(config)#interface FortyGigE0/5/0/0.1 l2transport
Router(config-subif)#encapsulation dot1q priority-tagged
Router(config-subif)#rewrite ingress tag pop 1 symmetric
Router(config-subif)#commit
```

Running Configuration

This section shows Rewrite of Priority Tag running configuration.

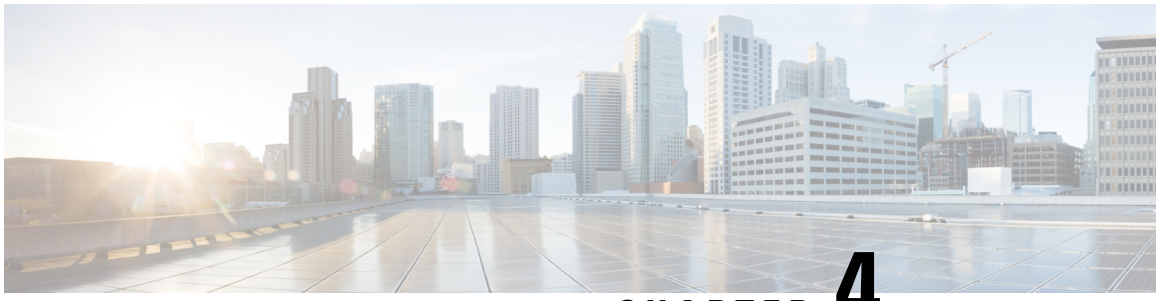
```
configure
 interface FortyGigE0/5/0/0.1 l2transport
  encapsulation dot1q priority-tagged
  rewrite ingress tag pop 1 symmetric
!
```

Related Topics

Rewrite of Priority Tag, on page 36

Associated Commands

- rewrite ingress tag pop 1 symmetric



CHAPTER 4

L2CP Tunneling MEF

This chapter introduces you to L2 Control Protocols (L2CP) tunneling to help initiate control packets from a local customer-edge (CE) device to a remote CE device.

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
L2CP Tunneling	Release 7.4.1	This feature is now supported on routers that have Cisco NC57 line cards installed and operates in native and compatibility modes. L2 Control Protocols (L2CP) tunneling helps initiate control packets from a local customer-edge (CE) device to a remote CE device.

- L2CP Tunneling, on page 39
- L2CP Protocol Support on Cisco NCS Series Router, on page 40

L2CP Tunneling

The router supports the following tunnel protocols:

- Link Layer Discovery Protocol (LLDP)
- Link Aggregation Control Protocol (LACP)
- Operation, Administration, Management (OAM)
- Ethernet Local Management Interface (ELMI)
- Cisco Discovery Protocol (CDP)

Some of the L2 transport interfaces are:

- VPWS L2 transport main
- VPWS L2 subinterface
- L2 transport main bridge port
- L2 subinterface bridge port

- VPWS L2 bundle main port
- VPWS L2 bundle subinterface
- L2 bundle main bridge port

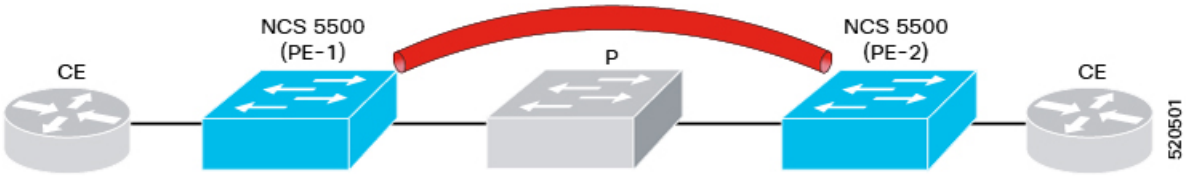
On a subinterface, when control packets such as LLDP and LACP are tunneled, the system tunnels the same control packets to the main interface.

The LACP packet for VPLS (also known as ELAN service) either gets peered or dropped.

The router tunnels Layer 2 packets between PEs.

The following figure depicts Layer 2 protocol tunneling.

Figure 2: L2CP Tunneling



L2CP packets are tunneled from NNI to NNI (depicted in red pipe). The Layer 2 traffic is sent through the Cisco NCS Routers, and these routers switch the traffic from end to end.

Restrictions

- VPLS service does not support LACP tunneling.
- VPWS and EVPN-VPWS services support LACP tunneling.

L2CP Protocol Support on Cisco NCS Series Router

The router supports Layer 2 peering functionalities on a per Ethernet Flow Point (EFP) basis. It supports maximum packet rate of 10 packets per second (per interface) for a protocol, and 100 packets per second for all protocols (on all interfaces).

You do not need to configure L2CP tunneling explicitly. L2CP packets are tunneled over Layer 2 tunnel by default.

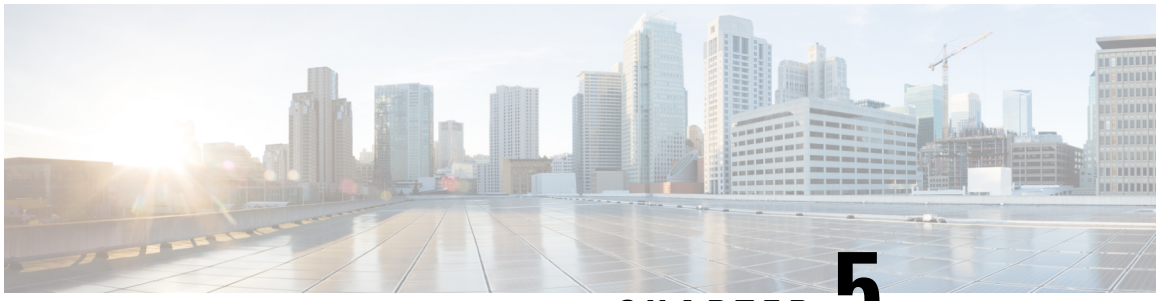
The following table lists the options that are supported on the router and displays the supported defaults and configuration options for the router.

Note LLDP will not detect neighboring devices if the main interface has a sub-interface configured with **encapsulation untagged** or **encapsulation default**.

Protocol	Packet Type	Action
CDP	Untagged	Peer
LACP	Untagged	Peer

Protocol	Packet Type	Action
LLDP	Untagged	Peer else Tunneled
STP	Untagged	Peer
VTP	Untagged	Peer
OAM	Untagged	Peer
BPDU	Untagged	Tunneled
UDLD	Untagged	Tunneled
CDP	Tagged	Tunneled
LACP	Tagged	Tunneled
LLDP	Tagged	Tunneled
STP	Tagged	Tunneled
VTP	Tagged	Tunneled
BPDU	Tagged	Tunneled
OAM	Tagged	Tunneled
ELMI	Tagged	Tunneled
UDLD	Tagged	Tunneled

Note L2CP protocols over BVI is not supported.



CHAPTER 5

Configure Link Bundles for Layer 2 VPNs

An ethernet link bundle is a group of one or more ports that are aggregated together and treated as a single link. Each bundle has a single MAC, a single IP address, and a single configuration set (such as ACLs or QoS).

The advantages of link bundling are:

- Redundancy - Because bundles have multiple links, the failure of a single link does not cause a loss of connectivity.
- Increased bandwidth - On bundled interfaces traffic is forwarded over all available members of the bundle aggregating individual port capacity.

There are two types of link bundling supported depending on the type of interface forming the bundle:

- Ethernet interfaces
- VLAN interfaces (bundle sub-interfaces)

This section describes the configuration of ethernet and VLAN link bundles for use in Layer 2 VPNs.

- Configure Gigabit Ethernet Link Bundle, on page 43
- Configure VLAN Bundle, on page 46
- References for Configuring Link Bundles, on page 47

Configure Gigabit Ethernet Link Bundle

Cisco IOS XR software supports the EtherChannel method of forming bundles of Ethernet interfaces. EtherChannel is a Cisco proprietary technology that allows the user to configure links to join a bundle, but has no mechanisms to check whether the links in a bundle are compatible.

IEEE 802.3ad encapsulation employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in an ethernet bundle are compatible. Links that are incompatible or have failed are automatically removed from the bundle.

Cisco NCS 560 Series Router supports 100G link bundles.

Restrictions

- All links within a single ethernet link bundle must be configured either to run 802.3ad (LACP) or Etherchannel (non-LACP). Mixed links within a single bundle are not supported.
- MAC accounting is not supported on Ethernet link bundles.

- The maximum number of supported links in each ethernet link bundle is .
- The maximum number of supported ethernet link bundles is .
- On an egress (NCS 5500 Series) router's NC57 line card, when BUM traffic is sent from the core network to the AC network over a bundle interface, the hash function is not performed on active bundle member interfaces. Rather, a specific bundle pin-down member interface is used for sending BUM traffic.

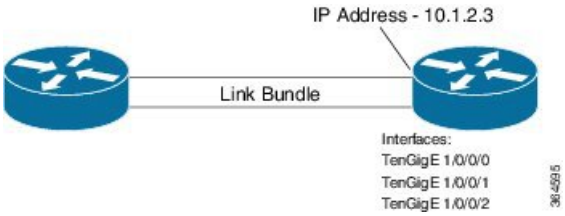
Configuration Example

To create a link bundle between two routers, you must complete the following configurations:

1. Create a bundle instance
2. Map physical interface (s) to the bundle.

Sample values are provided in the following figure.

Figure 3: Link Bundle Topology



For an Ethernet bundle to be active, you must perform the same configuration on both connection endpoints of the bundle.

Configuration

```
/* Enter the global configuration mode and create the ethernet link bundle */
Router# configure
Router(config)# interface Bundle-Ether 3
Router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
Router(config-if)# bundle maximum-active links 32 hot-standby
Router(config-if)# bundle minimum-active links 1
Router(config-if)# bundle minimum-active bandwidth 30000000
Router(config-if)# exit

/* Map physical interfaces to the bundle */
/* Note: Mixed link bundle mode is supported only when active-standby operation is configured
 */
Router(config)# interface TenGigE 1/0/0/0
Router(config-if)# bundle id 3 mode on
Router(config-if)# no shutdown
Router(config)# exit

Router(config)# interface TenGigE 1/0/0/1
Router(config-if)# bundle id 3 mode on
Router(config-if)# no shutdown
Router(config-if)# exit

Router(config)# interface TenGigE 1/0/0/2
Router(config-if)# bundle id 3 mode on
```

```
Router(config-if)# no shutdown
Router(config-if)# exit
```

Running Configuration

```
Router# show running-configuration
configure
interface Bundle-Ether 3
 ipv4 address 10.1.2.3 255.0.0.0
 bundle maximum-active links 32 hot-standby
 bundle minimum-active links 1
 bundle minimum-active bandwidth 30000000
!
interface TenGigE 1/0/0/0
 bundle-id 3 mode on
!

interface TenGigE 1/0/0/1
 bundle-id 3 mode on
!
interface TenGigE 1/0/0/2
 bundle-id 3 mode on
!
```

Verification

Verify that interfaces forming the bundle are active and the status of the bundle is Up.

```
Router# show bundle bundle-ether 3
Tue Feb  4 18:24:25.313 UTC

Bundle-Ether1
  Status:                                  Up
  Local links <active/standby/configured>:   3 / 0 / 3
  Local bandwidth <effective/available>:     30000000 (30000000) kbps
  MAC address (source):                      1234.1234.1234 (Configured)
  Inter-chassis link:                        No
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      32
  Wait while timer:                          2000 ms
  Load balancing:                            Default
  LACP:                                      Not operational
    Flap suppression timer:                  Off
    Cisco extensions:                        Disabled
    Non-revertive:                           Disabled
  mLACP:                                     Not configured
  IPv4 BFD:                                  Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Te1/0/0/0             Local            Active       0x8000, 0x0000    10000000
      Link is Active
  Te1/0/0/1             Local            Active       0x8000, 0x0000    10000000
      Link is Active
  Te1/0/0/2             Local            Active       0x8000, 0x0000    10000000
      Link is Active

------------------------------------------------------------------------------
```

Associated Commands

- bundle maximum-active links

- interface Bundle-Ether
- show bundle Bundle-Ether

Configure VLAN Bundle

The procedure for creating VLAN bundle is the same as the procedure for creating VLAN sub-interfaces on a physical ethernet interface.

Configuration Example

To configure VLAN bundles, complete the following configurations:

- Create a bundle instance.
- Create a VLAN interface (bundle sub-interface).
- Map the physical interface(s) to the bundle.

For a VLAN bundle to be active, you must perform the same configuration on both end points of the VLAN bundle.

Configuration

```
/* Enter global configuration mode and create VLAN bundle */
Router# configure
Router(config)# interface Bundle-Ether 2
Router(config-if)# ipv4 address 50.0.0.1/24
Router(config-if)# bundle maximum-active links 32 hot-standby
Router(config-if)# bundle minimum-active bandwidth 30000000
Router(config-if)# bundle minimum-active links 1
Router(config-if)# commit

/* Create VLAN sub-interface and add to the bundle */
Router(config)# interface Bundle-Ether 2.201
Router(config-subif)# ipv4 address 12.22.1.1 255.255.255.0
Router(config-subif)# encapsulation dot1q 201
Router(config-subif)# commit

/* Map the physical interface to the bundle */
Router(config)# interface TenGigE 0/0/0/14
Router(config-if)# bundle id 2 mode on
Router(config-if)# no shutdown
Router(config-if)# commit

/* Repeat the above steps for all the member interfaces:
   0/0/0/15, 0/0/0/16 and 0/0/0/17 in this example */
```

Running Configuration

```
configure
interface Bundle-Ether2
 ipv4 address 50.0.0.1 255.255.255.0
 mac-address 1212.1212.1212
 bundle maximum-active links 32 hot-standby
 bundle minimum-active links 1
```

```
 bundle minimum-active bandwidth 30000000
!
interface Bundle-Ether2.201
 ipv4 address 12.22.1.1 255.255.255.0
 encapsulation dot1q 201
!
interface TenGigE0/0/0/14
 bundle id 2 mode on
!
interface TenGigE0/0/0/15
 bundle id 2 mode on
!
interface TenGigE0/0/0/16
 bundle id 2 mode on
!
interface TenGigE0/0/0/17
 bundle id 2 mode on
!
```

Verification

Verify that the VLAN status is UP.

```
Router# show interfaces bundle-ether 2.201

Wed Feb  5 17:19:53.964 UTC
Bundle-Ether2.201 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 28c7.ce01.dc7b
  Internet address is 12.22.1.1/24
  MTU 1518 bytes, BW 20000000 Kbit (Max: 20000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation 802.1Q Virtual LAN, VLAN Id 201,  loopback not set,
  Last link flapped 07:45:25
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output never
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     2938 packets input, 311262 bytes, 0 total input drops
  - - -
  - - -
```

Associated Commands

- bundle maximum-active links
- interface Bundle-Ether
- show bundle Bundle-Ether

References for Configuring Link Bundles

This section provides references to configuring link bundles. For an overview of link bundles and configurations, see Configure Link Bundles for Layer 2 VPNs, on page 43.

Characteristics of Link Bundles

- Any type of Ethernet interfaces can be bundled, with or without the use of LACP (Link Aggregation Control Protocol).
- Physical layer and link layer configuration are performed on individual member links of a bundle.
- Configuration of network layer protocols and higher layer applications is performed on the bundle itself.
- A bundle can be administratively enabled or disabled.
- Each individual link within a bundle can be administratively enabled or disabled.
- Ethernet link bundles are created in the same way as Etherokinet channels, where the user enters the same configuration on both end systems.
- The MAC address that is set on the bundle becomes the MAC address of the links within that bundle.
- When LACP configured, each link within a bundle can be configured to allow different keepalive periods on different members.
- Load balancing is done by flow instead of by packet. Data is distributed to a link in proportion to the bandwidth of the link in relation to its bundle.
- QoS is supported and is applied proportionally on each bundle member.
- Link layer protocols, such as CDP, work independently on each link within a bundle.
- Upper layer protocols, such as routing updates and hello messages, are sent over any member link of an interface bundle.
- Bundled interfaces are point to point.
- A link must be in the UP state before it can be in distributing state in a bundle.
- Access Control List (ACL) configuration on link bundles is identical to ACL configuration on regular interfaces.
- Multicast traffic is load balanced over the members of a bundle. For a given flow, internal processes select the member link and all traffic for that flow is sent over that member.

Methods of Forming Bundles of Ethernet Interfaces

Cisco IOS-XR software supports the following methods of forming bundles of Ethernet interfaces:

- IEEE 802.3ad—Standard technology that employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in a bundle are compatible. Links that are incompatible or have failed are automatically removed from a bundle.

  For each link configured as bundle member, information is exchanged between the systems that host each end of the link bundle:

  - A globally unique local system identifier
  - An identifier (operational key) for the bundle of which the link is a member
  - An identifier (port ID) for the link

- The current aggregation status of the link

This information is used to form the link aggregation group identifier (LAG ID). Links that share a common LAG ID can be aggregated. Individual links have unique LAG IDs.

The system identifier distinguishes one router from another, and its uniqueness is guaranteed through the use of a MAC address from the system. The bundle and link identifiers have significance only to the router assigning them, which must guarantee that no two links have the same identifier, and that no two bundles have the same identifier.

The information from the peer system is combined with the information from the local system to determine the compatibility of the links configured to be members of a bundle.

Bundle MAC addresses in the routers come from a set of reserved MAC addresses in the backplane. This MAC address stays with the bundle as long as the bundle interface exists. The bundle uses this MAC address until the user configures a different MAC address. The bundle MAC address is used by all member links when passing bundle traffic. Any unicast or multicast addresses set on the bundle are also set on all the member links.

Note It is recommended that you avoid modifying the MAC address, because changes in the MAC address can affect packet forwarding.

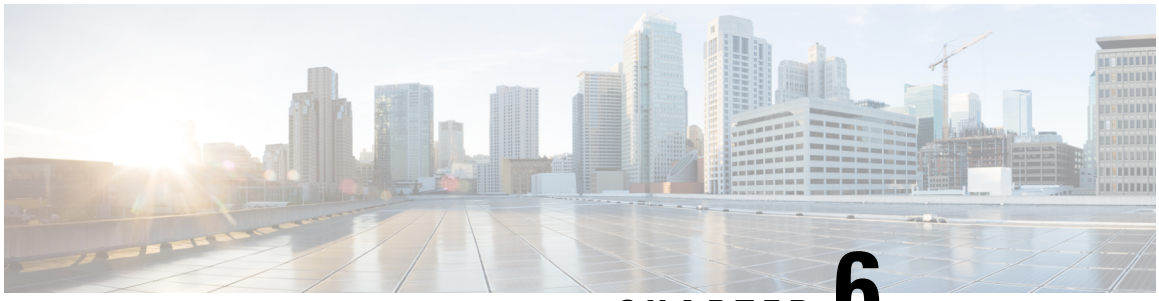
- EtherChannel—Cisco proprietary technology that allows the user to configure links to join a bundle, but has no mechanisms to check whether the links in a bundle are compatible.

Link Aggregation Through LACP

The optional Link Aggregation Control Protocol (LACP) is defined in the IEEE 802 standard. LACP communicates between two directly connected systems (or peers) to verify the compatibility of bundle members. For a router, the peer can be either another router or a switch. LACP monitors the operational state of link bundles to ensure these:

- All links terminate on the same two systems.
- Both systems consider the links to be part of the same bundle.
- All links have the appropriate settings on the peer.

LACP transmits frames containing the local port state and the local view of the partner system's state. These frames are analyzed to ensure both systems are in agreement.



CHAPTER 6

Configure Multipoint Layer 2 Services

This module provides the conceptual and configuration information for Multipoint Layer 2 Bridging Services, also called Virtual Private LAN Services (VPLS).

Note VPLS supports Layer 2 VPN technology and provides transparent multipoint Layer 2 connectivity for customers. This approach enables service providers to host a multitude of new services such as broadcast TV and Layer 2 VPNs.

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
Virtual Private LAN Services (VPLS)	Release 7.4.1	This feature is now supported on routers that have Cisco NC57 line cards installed and operates in native and compatibility modes. Virtual Private LAN Services (VPLS) is a class of VPN that supports the connection of multiple sites in a single bridged domain over a MPLS network.

- Prerequisites for Implementing Multipoint Layer 2 Services, on page 51
- Information About Implementing Multipoint Layer 2 Services, on page 52
- How to Implement Services, on page 72
- MAC Address Withdrawal, on page 92
- Configure MAC Address Withdrawal, on page 93
- MAC Loop Prevention, on page 95
- Configuration Examples for Multipoint Layer 2 Services, on page 99
- LDP-Based VPLS and VPWS FAT Pseudowire, on page 108
- PPPoE traffic-based load balancing, on page 113

Prerequisites for Implementing Multipoint Layer 2 Services

Before configuring Multipoint Layer 2 Services, ensure that these tasks and conditions are met:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

  If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- Configure IP routing in the core so that the provider edge (PE) routers can reach each other through IP.
- Configure a loopback interface to originate and terminate Layer 2 traffic. Make sure that the PE routers can access the other router's loopback interface.

Information About Implementing Multipoint Layer 2 Services

To implement Multipoint Layer 2 Services, you must understand these concepts:

Multipoint Layer 2 Services Overview

Multipoint Layer 2 Services enable geographically separated local-area network (LAN) segments to be interconnected as a single bridged domain over an MPLS network. The full functions of the traditional LAN such as MAC address learning, aging, and switching are emulated across all the remotely connected LAN segments that are part of a single bridged domain. A service provider can offer VPLS service to multiple customers over the MPLS network by defining different bridged domains for different customers. Packets from one bridged domain are never carried over or delivered to another bridged domain, thus ensuring the privacy of the LAN service.

Note VPLS PW is not supported over BGP multipath.

Some of the components present in a Multipoint Layer 2 Services network are described in these sections.

Note Multipoint Layer 2 services are also called as Virtual Private LAN Services.

Note Traffic to directly connected neighbor on EVPN or VPLS bridge won't work in the following scenarios:

- If neighbor doesn't advertise MPLS explicit null.
- If imposition node has a mix of implicit-null and labeled paths in ECMP or LFA deployment.

Restrictions for VPLS

These restrictions apply to multipoint layer 2 services:

- Pseudowire statistics are not supported on VPLS bridge domains. The show commands do not display PW statistics.

Bridge Domain

The native bridge domain refers to a Layer 2 broadcast domain consisting of a set of physical or virtual ports (including VFI). Data frames are switched within a bridge domain based on the destination MAC address. Multicast, broadcast, and unknown destination unicast frames are flooded within the bridge domain. In addition, the source MAC address learning is performed on all incoming frames on a bridge domain. A learned address is aged out. Incoming frames are mapped to a bridge domain, based on either the ingress port or a combination of both an ingress port and a MAC header field.

When the number of bridge domains exceeds 200, to enable clean up and reprogramming, it takes about 120 seconds for unconfiguring L2VPN and rollback.

The following table details the minimum interval required between unconfiguring L2VPN and rollback:

Number of BDs	Minimum interval in seconds
250	180
500	300
750 or greater	600

Bridge Domain and BVI Scale

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
Bridge Domain and MAC Address Scale	Release 7.4.1	This feature is now supported on Cisco NCS 5700 series fixed port routers and the Cisco NCS 5500 series routers that have the Cisco NC57 line cards installed and operating in the native mode. This feature is supported with the following scale values: • 8K Bridge Domain • 4K Pseudowires • 300K MAC addresses

The number of bridge domains (BDs) depends on the number of attachment circuits (ACs) configured per BD and also if Bridge-Group Virtual Interface (BVI) is configured or not. The number of logical interfaces (LIF) supported is less than 4000.

The following table provides an example of how the number of logical interfaces (LIF) required is calculated when two ACs are configured per BD.

Bridge Domain	Number of Bridges	AC	Total LIF required
BD with BVI	625	2	3750
BD without BVI	125	2	250
Total BD	750	-	-

Here is how the number of LIF required is calculated:

a*3+b, where a is the number of ACs with BVI and b is the number of ACs without BVI, must not exceed 4000.

Pseudowires

A pseudowire is a point-to-point connection between pairs of PE routers. Its primary function is to emulate services like Ethernet over an underlying core MPLS network through encapsulation into a common MPLS format. By encapsulating services into a common MPLS format, a pseudowire allows carriers to converge their services to an MPLS network.

Access pseudowires

The Access Pseudowire (PW) feature streamlines network architecture by:

- reducing the number of PWs required between provider edge (PE) devices and
- minimizes signaling overhead and packet duplication by connecting customer edge (CE) devices through an access PW.

Key characteristics of access PW:

- **Termination**: Access PWs terminate directly on PE devices rather than on a physical or logical port, which distinguishes them from traditional VPLS PW.
- **VPLS configuration**: In a Virtual Private LAN Service (VPLS) setup, establish an access PW between CE devices and PE devices for each VPLS service. This setup enables efficient connectivity and traffic management.
- **Scalability**:
  - VPLS typically requires a full mesh of PWs between Layer 2 VPN PEs participating in the service. As the number of PEs increases, the number of required PWs also grows, which can lead to scalability issues.
  - Implementing a hierarchical PE design can help reduce the number of PW by using access PWs, thereby mitigating scalability challenges.

Configure Access Pseudowire

Perform these steps to configure the access pseudowire.

Procedure

Step 1 Perform these steps on user provider edge devices such as U-PE1, see Access Pseudowire.

a) Configure a Layer 2 transport sub-interface with VLAN encapsulation and symmetric VLAN tag rewriting to facilitate efficient traffic handling and seamless integration into the network.

Example:

```
Router# configure
Router(config)# interface TenGigE0/1/0/5.2 l2transport
```

```
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# exit
```

b) Configure a point-to-point pseudowire connection with an xconnect group and neighbor specification to establish a seamless Layer 2 VPN link between local and remote provider edge devices.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/1/0/5.2
Router(config-l2vpn-xc-p2p)# neighbor 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Step 2 Perform these steps on network provider edge devices such as N-PE1, see Access Pseudowire.

a) Configure a pseudowire class with MPLS encapsulation and Ethernet transport mode to enable consistent settings and efficient data transport over the network.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# router-id 172.16.0.1
Router(config-l2vpn)# pw-class class1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet
Router(config-l2vpn-pwc-mpls)# exit
```

b) Configure a bridge group with a VFI and neighbor specifications to establish seamless Layer 2 VPN connectivity using pseudowires and split-horizon rules for efficient and loop-free data transport.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# neighbor 10.2.2.2 pw-id 3
Router(config-l2vpn-bg-bd-pw)# split-horizon group
Router(config-l2vpn-bg-bd-pw)# exit
Router(config-l2vpn-bg-bd)# vfi vfi1
Router(config-l2vpn-bg-bd-vfi)# neighbor 10.0.0.1 pw-id 2
Router(config-l2vpn-bg-bd-vfi-pw)# pw-class class1
Router(config-l2vpn-bg-bd-vfi-pw)# commit
```

Step 3 Run the **show running-config l2vpn** command on the network provider edge devices such as N-PE1 to verify the access PW configuration.

Example:

```
Router# show running-config l2vpn
l2vpn
 bridge group bg
  bridge-domain bd
   neighbor 10.2.2.2 pw-id 3
    split-horizon group
   !
   vfi vfi1
    neighbor 10.0.0.1 pw-id 2
     pw-class class1
```

```
    !
   !
  !
 !
!
```

Step 4 Run the **show l2vpn xconnect group XCON1** command on the network provider edge devices such as N-PE1 to verify the xconnect group is active.

Example:

```
Router# show l2vpn xconnect group XCON1
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                       Segment 2
Group      Name       ST    Description            ST       Description      ST
-----------------    -----  --------------    ------------------------------   ------
XCON_1     xc1        UP    Te0/1/0/5.2            UP       172.16.0.1   1   UP
-------------------------------------------------------------------------------
```

Step 5 Run the **show l2vpn bridge-domain bd1** command on the network provider edge devices such as N-PE1 to verify the access PW on the interface.

Example:

```
Router:N-PE1#show l2vpn bridge-domain bd1
PW: neighbor 10.0.0.1, PW ID 2, state is up ( established )
      PW class mpls, XC ID 0xc0000008
      Encapsulation MPLS, protocol LDP
      Source address 172.16.0.1
      PW type Ethernet, control word disabled, interworking none
      PW backup disable delay 0 sec
      Sequencing not set
      LSP : Up

      PW Status TLV in use
        MPLS         Local                          Remote
        ------------ ------------------------------ --------------------------
        Label        24752                          24752
        Group ID     0x2                            0x2
        Interface    Access PW                      Access PW
        MTU          1500                           1500
        Control word disabled                       disabled
        PW type      Ethernet                       Ethernet
        VCCV CV type 0x2                            0x2
                     (LSP ping verification)        (LSP ping verification)
        VCCV CC type 0x6                            0x6
                     (router alert label)           (router alert label)
                     (TTL expiry)                   (TTL expiry)
        ------------ ------------------------------ --------------------------
```

Virtual Forwarding Instance

VPLS is based on the characteristic of virtual forwarding instance (VFI). A VFI is a virtual bridge port that is capable of performing native bridging functions, such as forwarding, based on the destination MAC address, source MAC address learning and aging, and so forth.

A VFI is created on the PE router for each VPLS instance. The PE routers make packet-forwarding decisions by looking up the VFI of a particular VPLS instance. The VFI acts like a virtual bridge for a given VPLS instance. More than one attachment circuit belonging to a given VPLS are connected to the VFI. The PE router establishes emulated VCs to all the other PE routers in that VPLS instance and attaches these emulated VCs to the VFI. Packet forwarding decisions are based on the data structures maintained in the VFI.

Information About Implementing Multipoint Layer 2 Services

To implement Multipoint Layer 2 Services, you must understand these concepts:

VPLS for an MPLS-based Provider Core

VPLS is a multipoint Layer 2 VPN technology that connects two or more customer devices using bridging techniques. A bridge domain, which is the building block for multipoint bridging, is present on each of the PE routers. The access connections to the bridge domain on a PE router are called attachment circuits. The attachment circuits can be a set of physical ports, virtual ports, or both that are connected to the bridge at each PE device in the network.

After provisioning attachment circuits, neighbor relationships across the MPLS network for this specific instance are established through a set of manual commands identifying the end PEs. When the neighbor association is complete, a full mesh of pseudowires is established among the network-facing provider edge devices, which is a gateway between the MPLS core and the customer domain.

The MPLS/IP provider core simulates a virtual bridge that connects the multiple attachment circuits on each of the PE devices together to form a single broadcast domain. This also requires all of the PE routers that are participating in a VPLS instance to form emulated virtual circuits (VCs) among them.

Now, the service provider network starts switching the packets within the bridged domain specific to the customer by looking at destination MAC addresses. All traffic with unknown, broadcast, and multicast destination MAC addresses is flooded to all the connected customer edge devices, which connect to the service provider network. The network-facing provider edge devices learn the source MAC addresses as the packets are flooded. The traffic is unicasted to the customer edge device for all the learned MAC addresses.

VPLS for Layer 2 Switching

VPLS technology includes the capability of configuring the router to perform Layer 2 bridging. In this mode, the router can be configured to operate like other Cisco switches.

Note

- Storm control counters are not supported.

The storm control that is applied to multiple subinterfaces of the same physical port pertains to that physical port only. All subinterfaces with storm control configured are policed as aggregate under a single policer rate shared by all EFPs. None of the subinterfaces are configured with a dedicated policer rate. When a storm occurs on several subinterfaces simultaneously, and because subinterfaces share the policer, you can slightly increase the policer rate to accommodate additional policing.

These features are supported:

- Bridging IOS XR Trunk Interfaces
- Bridging on EFPs

Storm Control on Bridge Domain

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Storm Control Configuration for Subinterfaces	Release 7.8.1	Storm control helps prevent LAN ports from being disrupted by a broadcast, multicast, or unicast traffic storm. You can now configure different storm control rates for each subinterface on a physical port. This will give you control at a granular level and prevent flooding of excess traffic at the subinterface level. In earlier releases, storm control could be configured only at the physical port level or only on one subinterface under a main interface. This feature modifies the **hw-module storm-control-combine-policer-bw enable** command to enable per subinterface configuration support for storm control.

Storm Control provides Layer 2 port security under a Virtual Private LAN Services (VPLS) bridge by preventing excess traffic from disrupting the bridge.

A traffic storm occurs when packets flood a VPLS bridge, creating excessive traffic and degrading network performance. Storm control prevents VPLS bridge disruption by suppressing traffic when the number of packets reaches configured threshold levels. You can configure separate threshold levels for different types of traffic on an access circuit (AC) under a VPLS bridge.

Storm control monitors incoming traffic levels on a port or a subinterface, and drops traffic when the number of packets reaches the configured threshold level during any 1-second interval. The 1-second interval is set in the hardware and is not configurable. The number of packets allowed to pass during this interval is configurable, per subinterface, per port, per traffic type. During this interval, the traffic level is compared with the configured storm control level. When the incoming traffic reaches the storm control level configured on the bridge port, storm control drops traffic until the end of storm control interval. At the beginning of a new interval, traffic of the specified type is allowed to pass on the port. The thresholds are configured using a packets per second (pps) and kilobit per second (kbps) rate.

Storm control has little impact on router performance. Packets passing through ports are counted regardless of whether the feature is enabled. Additional counting occurs only for the drop counters, which monitor dropped packets. Storm control counts the number of packets dropped per port. The drop counters are cumulative for all traffic types.

Supported Traffic Types for Storm Control

On each VPLS bridge port, you can configure up to three storm control thresholds—one for each of the supported traffic types. If you do not configure a threshold for a traffic type, then storm control is not enabled on that port or interface for that traffic type.

The supported traffic types are:

- Broadcast traffic—Packets with a packet destination MAC address equal to FFFF.FFFF.FFFF.
- Multicast traffic—Packets with a packet destination MAC address not equal to the broadcast address, but with the multicast bit set to 1. The multicast bit is bit 0 of the most significant byte of the MAC address.
- Unknown unicast traffic—Packets with a packet destination MAC address not yet learned.

Storm control configurations

You can configure storm control at individual Layer 2 attachment points or bridge ports.

Comparison of storm control configurations

Storm control configuration characteristics	Storm control default configuration	Storm control with subinterface configuration
Support	From Release 7.4.2	From Release 7.8.1
Enforcement level	Main port level affects all sub interfaces collectively.	Per sub-interface level, policing on AC or Logical Interface (LIF).
Packets Per Second (PPS) rate application	Configured rates apply to aggregated traffic across all sub interfaces and main interface for the port.	Configured rates apply to sub interfaces.
Hierarchical QoS for ingress traffic	Supported	Not supported
AC scale	16k	By default, 16k ACs is supported, once you enable the hw-module storm-control- combine-policer-bw enable command the ACs support reduces to 8K.

Combined Policer Mode

Combined policer mode is introduced in Cisco NCS routers to conserve the policer resources used which helps in increasing the storm control policer scale.

In this mode when more than one policer is defined under a single attachment circuit [AC], then the result of policer is sum of all the policers applied under that AC. Layer 2 storm control feature is applicable only for the BUM traffic type in that particular AC.

Use the **hw-module storm-control-combine-policer-bw enable** command to enable the policer mode.

Note You must manually reload the router to activate the **hw-module storm-control-combine-policer-bw** command.

With the combined mode policer profile, any policer type configured is applicable for any of the *broadcast*, *unknown-unicast*, or *multicast* traffic under that AC and all are rate limited up to the policer configured value.

For example, *broadcast* policer is configured at 1000 kbps will rate limit any of the *broadcast*, *unknown-unicast*, *multicast* to a maximum of 1000 kbps combined. *Broadcast* policer configured at 1000 kbps in addition to the multicast policer configured at 2000 kbps, will rate limit any of the *broadcast*, *unknown-unicast*, *multicast* to the maximum of 3000 kbps combined.

Starting from Release 7.8.1, you can use the **hw-module storm-control-combine-policer-bw enable** command to enable storm control configuration per subinterface. When you configure storm control on the subinterfaces, for each subinterface configuration, a policer ID is derived from the logical interface. The multicast destination will derive the policer ID for further processing of packets.

Restrictions for Storm Control

- On NPU systems, the storm-control policer applies to each NPU core. If the members span across multiple NPU cores, the policing occurs on a per-core basis.
- Only BUM traffic ingress from ACs will be subjected to storm-control policers. BUM traffic ingressing on VPLS PW and EVPN core will not be subjected to storm-control policers.
- Bridge-domains and Bridge-ports do not support storm-control configuration.
- Cisco IOS XR does not support storm-control counters.
- Storm-control on Bundle ACor BD spanning across NPUs is policed at specified rate per NPU.
- EVPN core or PWE traffic does not experience storm control policer.

Configure Storm Control on Bridge Domain

You can configure storm control on a physical port or on a subinterface. The storm control rates that are configured on a subinterface is applied to all the subinterfaces in the main port.

The thresholds are configured using packets per second (pps) or kilobit per second (kbps) rate.

Configuration Example

1. Create a bridge group with bridge domain.
2. Assign an interface or subinterface to the bridge domain.
3. Configure storm control for the interface or subinterface.

The following example shows storm control configured for broadcast traffic type on an interface:

```
Router# configure
```

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg0
Router(config-l2vpn-bg)# bridge-domain bd0
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/1
Router(config-l2vpn-bg-bd-ac)# storm-control broadcast pps 4500
Router(config-l2vpn-bg-bd-ac)# commit
Router(config-l2vpn-bg-bd-ac)# exit
```

Running Configuration

```
configure
 l2vpn
  bridge group bg0
   bridge-domain bd0
    interface HundredGigE0/0/0/1
     storm-control broadcast pps 4500
    !
```

Verification

The following example shows a truncated output.

```
Router# show l2vpn bridge-domain bd-name bd0 detail
Legend: pp = Partially Programmed.
Bridge group: bg0, bridge-domain: bd0, id: 0, state: up, ShgId: 0, MSTi: 0

  ...................

  No status change since creation
  ACs: 1 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    AC: HundredGigE0/0/0/1, state is unresolved
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 131072, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      E-Tree: Root
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control:
        Broadcast: enabled(4500 pps)
        Multicast: disabled
        Unknown unicast: disabled
      Static MAC addresses:
      PD System Data: Learn key: 0
```

Configure Storm Control per Subinterface

Starting from Release 7.8.1, you can enable per subinterface configuration support for storm control by using the **hw-module storm-control-combine-policer-bw enable** command. For more information, see Combined Policer Mode, on page 59.

By default, when storm control is configured on a subinterface, the same configuration is applied to all the subinterfaces in that main port.

Configuration Example

1. Configure combined policer to enable the storm control configuration on subinterfaces.
2. Create a bridge group with bridge domain.
3. Assign a subinterface to the bridge domain.
4. Configure storm control for the subinterface.
5. Assign another subinterface to the bridge domain and configure storm control with different parameters for the subinterface.

You can repeat the configuration of storm control on different subinterfaces of the same main port.

```
/* Configure combined policer */


Router# configure
Router(config)# hw-module storm-control-combine-policer-bw enable
```

Note You must manually reload the router to activate the **hw-module storm-control-combine-policer-bw enable** command.

```
/* Create a bridge group */


Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1

/* Assign a subinterface and configure storm control */


Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/1.10
Router(config-l2vpn-bg-bd-ac)# storm-control unknown-unicast pps 500
Router(config-l2vpn-bg-bd-ac)# storm-control multicast pps 2000
Router(config-l2vpn-bg-bd-ac)# storm-control broadcast pps 1000
Router(config-l2vpn-bg-bd-ac)# commit
Router(config-l2vpn-bg-bd-ac)# exit

/* Assign another subinterface and configure storm control with different parameters */


Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/1.20
Router(config-l2vpn-bg-bd-ac)# storm-control unknown-unicast pps 200
Router(config-l2vpn-bg-bd-ac)# storm-control multicast pps 1000
Router(config-l2vpn-bg-bd-ac)# storm-control broadcast pps 2000
Router(config-l2vpn-bg-bd-ac)# commit
Router(config-l2vpn-bg-bd-ac)# exit
```

Running Configuration

```
configure
 hw-module storm-control-combine-policer-bw enable
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface HundredGigE0/0/0/1.10
     storm-control unknown-unicast pps 500
     storm-control multicast pps 2000
     storm-control broadcast pps 1000
    !
    interface HundredGigE0/0/0/1.20
     storm-control unknown-unicast pps 200
     storm-control multicast pps 1000
     storm-control broadcast pps 2000
    !
```

Verification

The following example shows a truncated output.

```
Router# show l2vpn bridge-domain bd-name bd1 detail
Legend: pp = Partially Programmed.
Bridge group: bg1, bridge-domain: bd1, id: 1, state: up, ShgId: 0, MSTi: 0

 ..............

  ACs: 2 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    AC: HundredGigE0/0/0/1.10, state is unresolved
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 131072, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      E-Tree: Root
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control:
        Broadcast: enabled(1000 pps)
        Multicast: enabled(2000 pps)
        Unknown unicast: enabled(500 pps)
      Static MAC addresses:
      PD System Data: Learn key: 0

    AC: HundredGigE0/0/0/1.20, state is unresolved
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
```

```
MAC limit: 131072, Action: none, Notification: syslog
MAC limit reached: no, threshold: 75%
MAC port down flush: enabled
MAC Secure: disabled, Logging: disabled
Split Horizon Group: none
E-Tree: Root
Dynamic ARP Inspection: disabled, Logging: disabled
IP Source Guard: disabled, Logging: disabled
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control:
  Broadcast: enabled(2000 pps)
  Multicast: enabled(1000 pps)
  Unknown unicast: enabled(200 pps)
Static MAC addresses:
PD System Data: Learn key: 0
```

Interoperability Between Cisco IOS XR and Cisco IOS on VPLS LDP Signaling

The Cisco IOS Software encodes the NLRI length in the fist byte in bits format in the BGP Update message. However, the Cisco IOS XR Software interprets the NLRI length in 2 bytes. Therefore, when the BGP neighbor with VPLS-VPWS address family is configured between the IOS and the IOS XR, NLRI mismatch can happen, leading to flapping between neighbors. To avoid this conflict, IOS supports **prefix-length-size 2** command that needs to be enabled for IOS to work with IOS XR. When the **prefix-length-size 2** command is configured in IOS, the NLRI length is encoded in bytes. This configuration is mandatory for IOS to work with IOS XR.

This is a sample IOS configuration with the **prefix-length-size 2** command:

```
router bgp 1
 address-family l2vpn vpls
  neighbor 5.5.5.2 activate
  neighbor 5.5.5.2 prefix-length-size 2 --------> NLRI length = 2 bytes
 exit-address-family
```

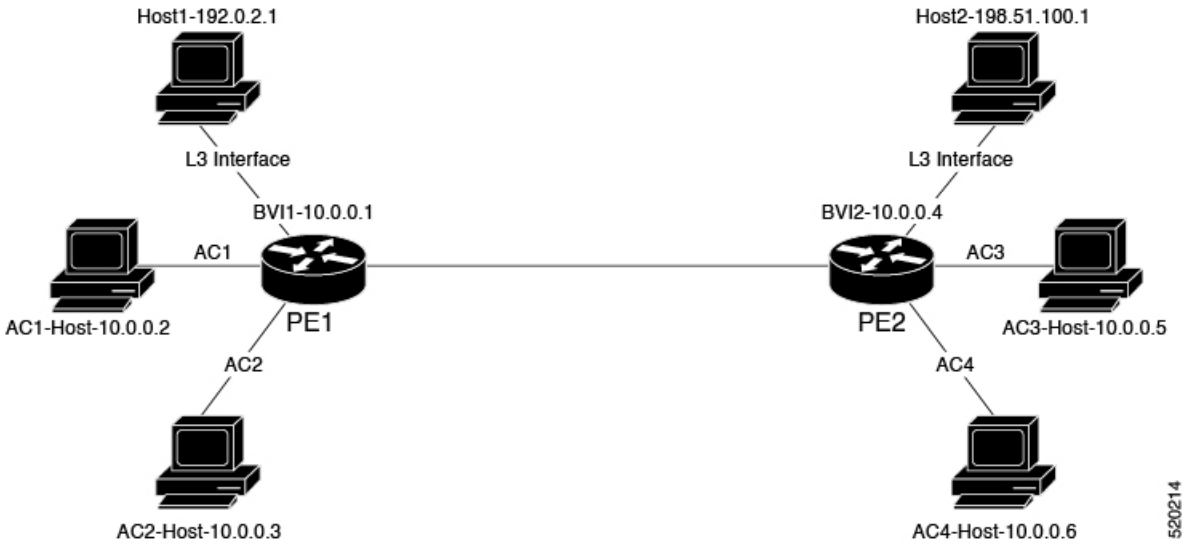
VPLS VFI with BVI as Routed Interface

The VPLS VFI with BVI as Routed Interface feature allows you to route the VPLS PW traffic over the BVI interface.

Integrated routing and bridging (IRB) enables you to route the packets in and out of a bridge domain using a Bridge-Group Virtual Interface (BVI). The BVI is a virtual interface configured on the router. It acts as a gateway routed interface towards the core network.

Configure a BVI on a single bridge domain that represents the link between the bridging and the routing domains on the router. To receive the packets from a bridged interface that are destined to a routed interface, configure BVI with the appropriate IP address, which is in the same subnet as the hosts in the bridge domain.

Figure 4: VPLS VFI with BVI as Routed Interface



This topology explains two types of traffic flow:

- Routed local traffic: Consider a traffic flow from AC1 Host to Host1. AC1 Host sends the traffic to BVI1. Attach AC1 Host and BVI1 to the same bridge domain of PE1. PE1 routs the traffic through BVI1 and sends it to Host1. L3 interface connects Host1 and PE1.
- Routed remote traffic: Consider a traffic flow from AC2 Host to Host2. AC2 Host sends the traffic to the bridge domain of PE1. PE1 sends the traffic to BVI2. AC2 Host is part of BVI2 subnet. PW sends the traffic to the bridge domain of PE2. PE2 routes the traffic through BVI2 and sends it to Host2. L3 interface connects Host2 and PE2.

Restrictions

- The following protocols are not supported when the bridge domain is attached with both PW and BVI: DHCP, ERPS, CDP, HSRP, IGMP Snooping, VRRP, CFM, LACP, and BFDoBVI.

Configure VPLS VFI with BVI as Routed Interface

Perform this task to route the VPLS PW traffic dynamically over the BVI interface.

Configuration Example

```
/* PE1 Configuration */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/0.1 -> AC1-L2 Sub-Interface (AC)
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#vfi core
Router(config-l2vpn-bg-bd-vfi)#neighbor 209.165.200.225 pw-id 1 -> VPLS Core-PW
Router(config-l2vpn-bg-bd-vfi-pw)#exit
Router(config-l2vpn-bg-bd-vfi)#exit
Router(config-l2vpn-bg-bd)#routed interface BVI1 -> BVI-1 Interface
```

```
Router(config-l2vpn-bg-bd-bvi)#root
Router(config)#interface BVI1
Router(config-if)#ipv4 address 10.0.0.1 255.0.0.0
Router(config-if)#commit


/* PE2 Configuration */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/1.1 -> AC3 L2 subinterface(AC)
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#vfi core
Router(config-l2vpn-bg-bd-vfi)#neighbor 209.165.200.226 pw-id 1 -> VPLS Core-PW
Router(config-l2vpn-bg-bd-vfi-pw)#exit
Router(config-l2vpn-bg-bd-vfi)#exit
Router(config-l2vpn-bg-bd)#routed interface BVI2 -> BVI-2 Interface
Router(config-l2vpn-bg-bd-bvi)#root
Router(config)#interface BVI2
Router(config-if)#ipv4 address 10.0.0.4 255.0.0.0
Router(config-if)#commit
```

Running Configuration

This section shows VPLS VFI with BVI as Routed Interface configuration.

```
/* PE1 Configuration */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface TenGigE0/0/0/0.1 -> AC1-L2 Sub-Interface (AC)
    !
    vfi core
     neighbor 209.165.200.225 pw-id 1 -> VPLS Core-PW
     !
    !
    routed interface BVI1 -> BVI-1 Interface
    !
  !
interface BVI1
 ipv4 address 10.0.0.1 255.0.0.0


/* PE2 Configuration */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd2
    interface TenGigE0/0/0/1.1 -> AC3 L2 Sub-Interface (AC)
    !
    vfi core
     neighbor 209.165.200.226 pw-id 1 -> VPLS Core-PW
     !
    !
    routed interface BVI2 -> BVI2 Interface
    !
  !
interface BVI2
 ipv4 address 10.0.0.4 255.0.0.0
```

Verification

Verify that you have configured the VPLS VFI with BVI as Routed Interface feature successfully.

```
Router-PE1#show l2vpn bridge-domain neighbor 209.165.200.225 detail
Legend: pp = Partially Programmed.
Bridge group: 1, bridge-domain: 1, id: 0, state: up, ShgId: 0, MSTi: 0
  VINE state: BVI Resolved
  MAC learning: enabled
  MAC withdraw: enabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  Create time: 10/01/2020 04:18:29 (00:14:06 ago)
  ACs: 2 (2 up), VFIs: 1, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of Access PWs:
  List of VFIs:
    VFI 1 (up)
      PW: neighbor 209.165.200.225, PW ID 1, state is up ( established )
        PW class mpls, XC ID 0xc0000002
        Encapsulation MPLS, protocol LDP
        Source address 209.165.200.226
        PW type Ethernet, control word disabled, interworking none
        Sequencing not set
        LSP : Up

        PW Status TLV in use
          MPLS         Local                          Remote
          ------------ ------------------------------ ------------------------
          Label        24006                          24002
          Group ID     0x0                            0x0
          Interface    1                              1
          MTU          1500                           1500
          Control word disabled                       disabled

          ------------ ------------------------------ ------------------------
```

Related Topics

- VPLS VFI with BVI as Routed Interface, on page 64

Associated Commands

- show l2vpn bridge-domain detail

MAC Address-related Parameters

The MAC address table contains a list of the known MAC addresses and their forwarding information. In the current VPLS design, the MAC address table and its management are maintained on the route processor (RP) card.

These topics provide information about the MAC address-related parameters:

MAC Address Flooding

Ethernet services require that frames that are sent to broadcast addresses and to unknown destination addresses be flooded to all ports. To obtain flooding within VPLS broadcast models, all unknown unicast, broadcast,

and multicast frames are flooded over the corresponding pseudowires and to all attachment circuits. Therefore, a PE must replicate packets across both attachment circuits and pseudowires.

MAC Address-based Forwarding

To forward a frame, a PE must associate a destination MAC address with a pseudowire or attachment circuit. This type of association is provided through a static configuration on each PE or through dynamic learning, which is flooded to all bridge ports.

MAC Address Source-based Learning

When a frame arrives on a bridge port (for example, pseudowire or attachment circuit) and the source MAC address is unknown to the receiving PE router, the source MAC address is associated with the pseudowire or attachment circuit. Outbound frames to the MAC address are forwarded to the appropriate pseudowire or attachment circuit.

MAC address source-based learning uses the MAC address information that is learned in the hardware forwarding path. The updated MAC tables are propagated and programs the hardware for the router.

Note Static MAC move is not supported from one port, interface, or AC to another port, interface, or AC. For example, if a static MAC is configured on AC1 (port 1) and then, if you send a packet with the same MAC as source MAC on AC2 (port 2), then you can't attach this MAC to AC2 as a dynamic MAC. Therefore, do not send any packet with a MAC as any of the static MAC addresses configured.

The number of learned MAC addresses is limited through configurable per-port and per-bridge domain MAC address limits.

MAC Address Aging

A MAC address in the MAC table is considered valid only for the duration of the MAC address aging time. When the time expires, the relevant MAC entries are repopulated. When the MAC aging time is configured only under a bridge domain, all the pseudowires and attachment circuits in the bridge domain use that configured MAC aging time.

A bridge forwards, floods, or drops packets based on the bridge table. The bridge table maintains both static entries and dynamic entries. Static entries are entered by the network manager or by the bridge itself. Dynamic entries are entered by the bridge learning process. A dynamic entry is automatically removed after a specified length of time, known as *aging time*, from the time the entry was created or last updated.

If hosts on a bridged network are likely to move, decrease the aging-time to enable the bridge to adapt to the change quickly. If hosts do not transmit continuously, increase the aging time to record the dynamic entries for a longer time, thus reducing the possibility of flooding when the hosts transmit again.

The range of MAC address aging time is from 300 seconds to 30,000 seconds. The maximum MAC address aging time among all bridges is considered for calculating the age. You cannot configure the MAC address aging time on each AC or PW interface. Configure MAC address aging time in the bridge domain configuration mode. There is no show command to display the highest MAC address aging time.

MAC Address Limit

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
Configure MAC Address Limit for Bridge Domains to Learn Static Addresses	Release 7.8.1	You can now configure the MAC address limit for bridge domains to learn only static MAC addresses and to drop traffic from unknown sources. Malicious attackers can spoof a Layer 2 MAC address to change dynamic entries in the MAC table. However, with this functionality enabling you to configure the MAC address limit for bridge domains to learn only static MAC addresses, the dynamic MAC addresses are blocked. In addition, a static entry always overrules dynamic entries. This functionality thus prevents the interception of your data by unauthorized users and improves your network security.

The MAC address limit is used to limit the number of learned MAC addresses. The default value for the MAC address limit is 32000.

When you configure MAC address limit for a bridge-domain, the following MAC addresses are considered by the network:

- Dynamic MAC addresses that are learned automatically via an ARP request.
- Static MAC addresses that are manually configured.
- EVPN Sync MAC addresses that are MAC entries synchronized across EVPN PEs.
- EVPN Remote MAC addresses that are assigned to remote EVPN PEs.
- You can configure up to a maximum of six different mac-limit values under a bridge group in the router.

Configure MAC Address Limit

Configure the MAC address limit using the **maximum** command. The MAC address learning is restricted to the configured limit.

When the number of learned MAC addresses reaches the configured limit, you can configure the bridge behavior by using the **action** command. You can configure the action to perform one of the following:

- **flood**: All the unknown unicast packets, with unknown destinations addresses, are flooded over the bridge.
- **no-flood**: All the unknown unicast packets, with unknown destination addresses, are dropped.

- **shutdown** : All the packets are dropped.

When the MAC limit is exceeded, use the **notification {both | none | trap}** command to send notifications in one of the following forms:

- **trap**: Sends Simple Network Management Protocol (SNMP) trap notification.
- **both**: Sends both syslog and trap notifications.
- **none**: No notifications are sent.

By default, syslog message is sent.

MAC address limit action applies only when the number of local MAC addresses exceeds the configured limit. The software unlearns the MAC addresses until it reaches the configured MAC limit threshold value. Later, the router restarts learning new MAC addresses. In the event when the MAC limit threshold is not configured, the default threshold is 75% of the configured MAC address limit.

Configuration Example

In this example, MAC address limit is configured as 5000 and MAC limit action is set to flood the packets. As notification is not configured, syslog entries are sent when the MAC limit is exceeded.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg-0
Router(config-l2vpn-bg)# bridge-domain bd-0
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# limit
Router(config-l2vpn-bg-bd-mac-limit)# maximum 5000
Router(config-l2vpn-bg-bd-mac-limit)# action flood
```

Verification

Use the **show l2vpn bridge-domain** command to view the MAC address limit configuration.

```
Router# show l2vpn bridge-domain bd-name bd-0 detail
Legend: pp = Partially Programmed.
Bridge group: bg-0, bridge-domain: bd-0, id: 25, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: EVPN Native
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 5000, Action: flood, Notification: syslog
  MAC limit reached: no, threshold: 80%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
```

Configure MAC Address Limit for Static MAC Addresses

The dynamic MAC addresses are not learned when the MAC address limit is configured to be less than the number of static MAC address entries. When you configure the MAC address limit as zero using the **maximum** command, the dynamic MAC addresses are blocked and only static MAC addresses are learned.

To discard the traffic arriving from an unknown source, set the MAC limit action as **no-flood**, so that all the unknown unicast, broadcast, and multicast packets are dropped.

Configuration Example

In this example, MAC address limit is configured as zero and MAC limit action is set to **no-flood**.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg-0
Router(config-l2vpn-bg)# bridge-domain bd-0
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# limit
Router(config-l2vpn-bg-bd-mac-limit)# maximum 0
Router(config-l2vpn-bg-bd-mac-limit)# action no-flood
```

Verification

Use the **show l2vpn bridge-domain** command to view the MAC address limit configuration.

```
Router# show l2vpn bridge-domain bd-name bd-0 detail
Legend: pp = Partially Programmed.
Bridge group: bg-0, bridge-domain: bd-0, id: 25, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: EVPN Native
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 0, Action: no-flood, Notification: syslog
  MAC limit reached: no, threshold: 80%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
```

MAC Address Withdrawal

For faster VPLS convergence, you can remove or unlearn the MAC addresses that are learned dynamically. The Label Distribution Protocol (LDP) Address Withdrawal message is sent with the list of MAC addresses, which need to be withdrawn to all other PEs that are participating in the corresponding VPLS service.

For the Cisco IOS XR VPLS implementation, a portion of the dynamically learned MAC addresses are cleared by using the MAC addresses aging mechanism by default. The MAC address withdrawal feature is added through the LDP Address Withdrawal message. To enable the MAC address withdrawal feature, use the **withdrawal** command in l2vpn bridge group bridge domain MAC configuration mode. To verify that the MAC address withdrawal is enabled, use the **show l2vpn bridge-domain** command with the **detail** keyword.

Note By default, the LDP MAC Withdrawal feature is enabled on Cisco IOS XR.

The LDP MAC Withdrawal feature is generated due to these events:

- Attachment circuit goes down. You can remove or add the attachment circuit through the CLI.

- MAC withdrawal messages are received over a VFI pseudowire. RFC 4762 specifies that both wildcards (by means of an empty Type, Length and Value [TLV]) and a specific MAC address withdrawal. Cisco IOS XR software supports only a wildcard MAC address withdrawal.

How to Implement Services

This section describes the tasks that are required to implement Multipoint Layer 2 Services:

Configuring a Bridge Domain

These topics describe how to configure a bridge domain:

Creating a Bridge Domain

Perform this task to create a bridge domain .

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group that can contain bridge domains, and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Associating Members with a Bridge Domain

After a bridge domain is created, perform this task to assign interfaces to the bridge domain. These types of bridge ports are associated with a bridge domain:

- Ethernet and VLAN
- VFI

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/4/0/0
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

Step 6 *(Optional)* **static-mac-address** { *MAC-address* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# static-mac-address 1.1.1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Bridge Domain Parameters

To configure bridge domain parameters, associate these parameters with a bridge domain:

- Maximum transmission unit (MTU)—Specifies that all members of a bridge domain have the same MTU. The bridge domain member with a different MTU size is not used by the bridge domain even though it is still associated with a bridge domain.
- Flooding—Flooding is enabled always.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the l2vpn configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5 **flooding disable**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# flooding disable
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Disables flooding.

Step 6 **mtu** *bytes*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mtu 1000
```

Adjusts the maximum packet size or maximum transmission unit (MTU) size for the bridge domain.

- Use the *bytes* argument to specify the MTU size, in bytes. The range is from 64 to 65535.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Disabling a Bridge Domain

Perform this task to disable a bridge domain. When a bridge domain is disabled, all VFIs that are associated with the bridge domain are disabled. You are still able to attach or detach members to the bridge domain and the VFIs that are associated with the bridge domain.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5 **shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# shutdown
```

Shuts down a bridge domain to bring the bridge and all attachment circuits and pseudowires under it to admin down state.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Flooding Disable

The Flooding Disable feature prevents forwarding of Broadcast, Unknown-unicast and Multicast (BUM) traffic on the bridge domain. You can disable flooding of BUM traffic at the bridge level or at the interface level. By disabling flooding at the bridge level, you can prevent forwarding of BUM traffic on attachment circuit (AC), pseudowire (PW), and EVPN LIFs.

You can also disable only unknown unicast traffic at the bridge level or at the interface level. By disabling flooding of unknown unicast traffic at the bridge level, you can prevent forwarding of unknown unicast traffic on attachment circuit (AC), pseudowire (PW), and EVPN LIFs.

By disabling flooding of unknown unicast traffic at the interface level, you can prevent forwarding of unknown unicast traffic on AC alone.

Configure Flooding Disable

Perform this task to configure Flooding Disable feature.

You can disable flooding of:

- BUM traffic at the bridge level
- Unknown-unicast traffic at the bridge level
- Unknown-unicast traffic at the interface level

However, the flooding disable of unknown-unicast traffic at the bridge level takes effect only when the **flooding disable** command is not configured for BUM traffic at the bridge level.

The flooding disable of unknown-unicast traffic at the interface level takes effect only when **flooding disable** and **flooding unknown-unicast disable** commands are not configured at the bridge level.

Configuration Example

```
/* Configuration to disable flooding of BUM traffic at the bridge level */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#flooding disable\
Router(config-l2vpn-bg-bd)#commit

/* Configuration to disable flooding of unknown-unicast traffic at the bridge level */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#flooding unknown-unicast disable
Router(config-l2vpn-bg-bd)#commit

/* Configuration to disable flooding of unknown-unicast traffic at the interface level */
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/0.2
Router(config-l2vpn-bg-bd-ac)#flooding unknown-unicast disable
Router(config-l2vpn-bg-bd-ac)#commit
```

Running Configuration

This section shows flooding disable running configuration.

```
/* Configuration to disable flooding of BUM traffic at the bridge level */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding disable
    flooding unknown-unicast disable
    interface TenGigE0/0/0/0.2
      flooding unknown-unicast disable
   !

/* Configuration to disable flooding of unknown-unicast traffic at the bridge level */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding unknown-unicast disable
    !
  !


/* Configuration to disable flooding of unknown-unicast traffic at the interface level */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface TenGigE0/0/0/0.2
      flooding unknown-unicast disable
     !
   !
 !
```

Associated Commands

- flooding disable
- flooding unknown-unicast disable

Configuring a Layer 2 Virtual Forwarding Instance

These topics describe how to configure a Layer 2 virtual forwarding instance (VFI):

Creating the Virtual Forwarding Instance

Perform this task to create a Layer 2 Virtual Forwarding Instance (VFI) on all provider edge devices under the bridge domain.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **vfi** {*vfi-name*}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Associating Pseudowires with the Virtual Forwarding Instance

After a VFI is created, perform this task to associate one or more pseudowires with the VFI.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **vfi** { *vfi name* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Associating a Virtual Forwarding Instance to a Bridge Domain

Perform this task to associate a VFI to be a member of a bridge domain.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **vfi** { *vfi name* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7 **static-mac-address** { *MAC-address* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# static-mac-address 1.1.1
```

Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Attaching Pseudowire Classes to Pseudowires

Perform this task to attach a pseudowire class to a pseudowire.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **vfi** { *vfi-name* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7 **pw-class** { *class-name* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# pw-class canada
```

Configures the pseudowire class template name to use for the pseudowire.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Pseudowires Using Static Labels

Perform this task to configure the Any Transport over Multiprotocol (AToM) pseudowires by using the static labels. A pseudowire becomes a static AToM pseudowire by setting the MPLS static labels to local and remote.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **vfi** { *vfi-name* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7 **mpls static label** { **local** *value* } { **remote** *value* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# mpls static label local 800 remote 500
```

Configures the MPLS static labels and the static labels for the pseudowire configuration. You can set the local and remote pseudowire labels.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Disabling a Virtual Forwarding Instance

Perform this task to disable a VFI. When a VFI is disabled, all the previously established pseudowires that are associated with the VFI are disconnected. LDP advertisements are sent to withdraw the MAC addresses that are associated with the VFI. However, you can still attach or detach attachment circuits with a VFI after a shutdown.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **vfi** { *vfi-name* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# shutdown
```

Disables the virtual forwarding interface (VFI).

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Step 8 **show l2vpn bridge-domain** [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show l2vpn bridge-domain detail
```

Displays the state of the VFI. For example, if you shut down the VFI, the VFI is shown as shut down under the bridge domain.

Configuring the MAC Address-related Parameters

These topics describe how to configure the MAC address-related parameters:

The MAC table attributes are set for the bridge domains.

Note The **show l2vpn forwarding bridge-domain BRIDGE_GROUP:BRIDGE_DOMAIN mac-address location R/S/I** command does not automatically dump MAC address hardware information. The show output information might not be current. Perform any of the following actions before executing the **show l2vpn forwarding bridge-domain BRIDGE_GROUP:BRIDGE_DOMAIN mac-address location R/S/I** command:

- Resynchronize the MAC address entries by executing **l2vpn resynchronize forwarding mac-address location R/S/I** command.
- Dump the MAC address table by running **show l2vpn forwarding bridge-domain mac-address location R/S/I** command.

Configuring the MAC Address Source-based Learning

Perform this task to configure the MAC address source-based learning.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domainname*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
```

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **mac**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6 **learning disable**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# learning disable
```

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Step 8 **show l2vpn bridge-domain** [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show l2vpn bridge-domain detail
```

Displays the details that the MAC address source-based learning is disabled on the bridge.

Configuring the MAC Address Aging

Perform this task to configure the parameters for MAC address aging.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **mac**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6 **aging**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# aging
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#
```

Enters the MAC aging configuration submode to set the aging parameters such as time and type.

Step 7 **time** { *seconds* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# time 300
```

Configures the maximum aging time.

- Use the *seconds* argument to specify the maximum age of the MAC address table entry. Aging time is counted from the last time that the switch saw the MAC address. The range of MAC address aging time is from 300 seconds to 30,000 seconds. The default value is 300 seconds.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Step 9 **show l2vpn bridge-domain** [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show l2vpn bridge-domain detail
```

Displays the details about the aging fields.

Disabling MAC Flush at the Bridge Port Level

Perform this task to disable the MAC flush at the bridge domain level.

You can disable the MAC flush at the bridge domain or bridge port level. By default, the MACs learned on a specific port are immediately flushed, when that port becomes nonfunctional.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5 **mac**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters l2vpn bridge group bridge domain MAC configuration mode.

Step 6 **port-down flush disable**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
port-down flush disable
```

Disables MAC flush when the bridge port becomes nonfunctional.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

MAC Address Withdrawal

The MAC Address Withdrawal feature provides faster convergence by removing MAC addresses that are dynamically learned. This feature uses Label Distribution Protocol (LDP)-based MAC address withdrawal message. A MAC list Type Length Value (TLV) is part of the MAC address withdrawal message.

This feature also supports optimization of MAC address withdrawal. The optimization allows PEs to retain the MAC addresses that are learned from the CE devices over the access side. Only MAC addresses that are learned from peer PEs are flushed out. This avoids unnecessary MAC flushing toward attachment circuit (AC) side and ensures better utilization of bandwidth and resources.

The MAC address withdrawal feature is enabled by default. Use **mac withdraw disable** command to disable the MAC address withdrawal feature.

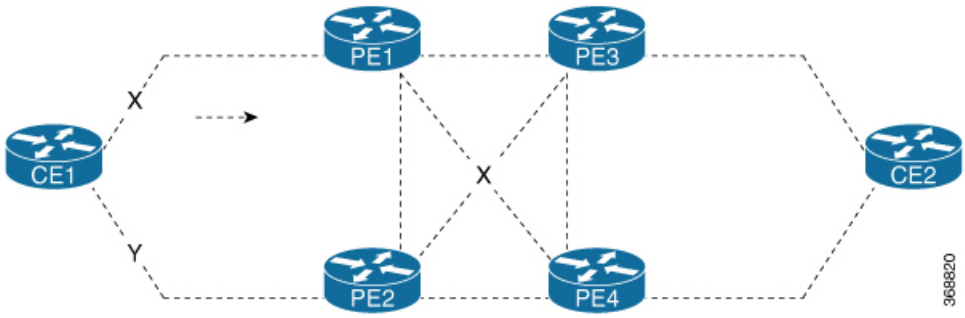
Topology

Consider the following topology in which CE1 is dual-homed to PE1 and PE2. The link X actively participates in VPLS while Y is a redundant link. Initially PE1, PE2, PE3, and PE4 learn their MAC forwarding tables that are based on the traffic profile and traffic becomes a known unicast. When the MAC address withdrawal

feature is enabled on all PEs, PEs delete MAC entries when they receive MAC address withdrawal message. The following are the MAC address withdrawal messages that are based on the status of link:

- Scenario 1: When link X, which is the AC of PE1 goes down, PE1 sends an LDP MAC withdrawal TLV message “FLUSH ALL MAC FROM ME” to neighbor PEs. Peer PEs delete MAC addresses that are learned only from PE1. PE2, PE3, and PE4 flush only MAC addresses that are learned from PE1. The PE1 initiates MAC flush when its access side AC goes down.
- Scenario 2: When link Y, which is the AC of PE2 comes up, PE2 sends an LDP MAC withdrawal TLV message “FLUSH ALL MAC BUT ME” to neighbor PEs. Peer PEs flush all MAC addresses except those from the PE which receives the request.

Figure 5: MAC Address Withdrawal



Restrictions

To configure MAC address withdrawal, the following restrictions are applicable:

- This feature is not supported over H-VPLS network.
- This feature is not supported over BGP signaling and discovery.
- MAC withdraw relaying is not supported.

Configure MAC Address Withdrawal

Configuration Example

Perform this task to configure MAC address withdrawal.

```
/* Configure MAC address withdrawal on PE1. This configuration is required for scenario 1
*/
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# withdraw state-down
Router(config-l2vpn-bg-bd-mac)# exit
Router(config-l2vpn-bg-bd)# interface tenGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# vfi vf1
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.1 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)# commit
```

```
/* Configure optimization of MAC address withdrawal on PE1. This configuration is required
 for scenario 1  */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# withdraw optimize
Router(config-l2vpn-bg-bd-mac)# exit
Router(config-l2vpn-bg-bd)# neighbor 192.0.2.1 pw-id 1234
Router(config-l2vpn-bg-bd-pw)# exit
Router(config-l2vpn-bg-bd)# vfi vf1
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.2 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)# exit
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.3 pw-id 2
Router(config-l2vpn-bg-bd-vfi-pw)# commit

/* MAC address withdrawal is enabled by default when AC comes up. Use the following
configuration if you want to disable MAC address withdrawal. This configuration is required
 for scenario 2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# withdraw disable
Router(config-l2vpn-bg-bd-mac)# commit
```

Running Configuration

This section shows the running configuration of MAC address withdrawal.

```
/* Configure MAC address withdrawal on PE1 */
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw state-down
   !
   interface tengige 0/0/0/0
   !
   vfi vf1
    neighbor 192.0.2.1 pw-id 1
    !

/* Configure optimization of MAC address withdrawal on PE1 */
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw optimize
   !
   neighbor neighbor 192.0.2.1 pw-id 1234
   !
   vfi vf1
    neighbor neighbor 192.0.2.2 pw-id 1
    !
    neighbor neighbor 192.0.2.3 pw-id 2


/* Disable MAC address widhrawal on PE2 */
```

```
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw disable
   !
```

Verification

Verify MAC address withdrawal configuration.

```
/* Verify if MAC address widhrawal is configured on PE1 */
Router:PE1# show l2vpn bridge-domain detail
MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port down

/* Verify if optimization of MAC address withdrawal is configured on PE1 */
Router:PE1# show l2vpn bridge-domain detail
MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port down (optimization)
```

Related Topics

- MAC Address Withdrawal, on page 92

Associated Commands

- mac withdraw
- show l2vpn bridge-domain detail

MAC Loop Prevention

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
MAC Loop Prevention	Release 7.5.2	This feature helps reduce network congestion and avoid traffic loss by shutting down a port after it reaches the configured number of MAC moves within the specified move interval. You can configure this feature at the bridge-domain level using the **mac secure** command. This feature is now supported on routers that operate in native and compatibility modes.

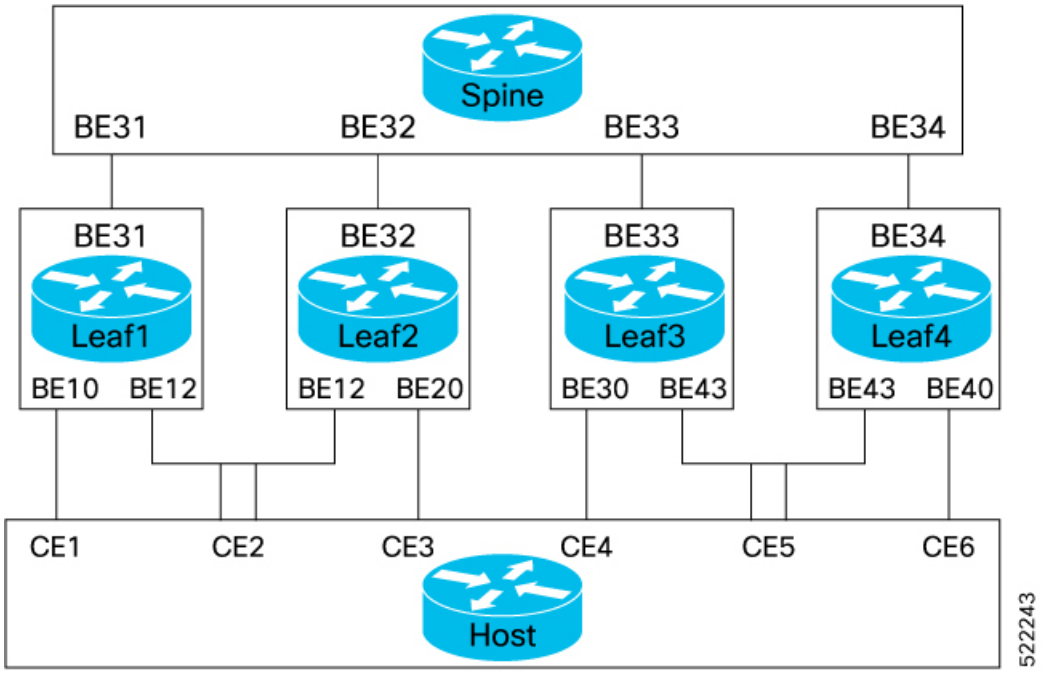
In case of network instability like an interface flap, a MAC address might be learned from a new interface. This is normal network convergence, and the mac-address-table is updated dynamically. A MAC move occurs when the same MAC address is learned on multiple interfaces. However, constant MAC moves often indicate network instability during an L2 loop. This feature lets you report MAC moves and take corrective actions such as shutting down an offending port.

The MAC Loop Prevention feature allows you to shut down the port after it exceeds the configured number of MAC moves within the specified move interval. You can configure this feature at the bridge-domain level using the **mac secure** command. The default number of MAC moves is five times for a move interval of 180 seconds. If the number of MAC moves exceeds the configured value, the MAC entry is marked as duplicate and the port is shut down. This feature helps you to reduce network congestion and avoid traffic loss. This feature is supported on physical and bundle AC, PW, and EVPN.

You can recover the shutdown port after a particular time by using the **shutdown-recovery-timeout** command after which the port automatically becomes active. If the recovery time is not configured, the shutdown port is recovered after three times of move interval. For example, if the move interval is 30 seconds, the shutdown port becomes active after 90 seconds.

If the recovery time is configured, the shutdown port is recovered after a maximum of either three times the move interval or the configured recovery timeout value. For example, if the move interval is 180 seconds and shutdown-recovery-timeout is 60 seconds, the shutdown port becomes active after 540 seconds.

Let's see how this feature works in the following scenarios:



- MAC learning within the node - When Leaf1 learns the same MAC address on both the interfaces, BE10 and BE12, the MAC is marked as duplicate and the port is shutdown after it exceeds the configured number of MAC moves within specified interval. For example, consider the MAC move count is configured as 5 for 180 seconds. If the traffic flows starts from BE10, and, the configured MAC move count ends at interface BE12, the port at the interface BE12 is shutdown.
- MAC learning between the nodes - When the same MAC address is learnt on BE10 and 20, the MAC is marked as duplicate and the port is shutdown after it exceeds the configured number of MAC moves within specified interval. For example, consider the MAC move count is configured as 5 for 180 seconds. If the traffic starts from BE10, the configured MAC move count ends at interface BE20, the port at the interface BE20 is shutdown.

Note When a Leaf that is enabled with this feature receives two MACs from the Leafs that are not enabled with this feature, this feature won't take effect.

Configuration Example

Perfrom this task on a Leaf to configure MAC loop prevention.

```
/* MAC Loop Prevention for VPLS */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge-group BG1
Router(config-l2vpn-bg)#bridge-domain BD1
Router(config-l2vpn-bg-bd)#mac secure
Router(config-l2vpn-bg-bd-mac-sec)#action shutdown
Router(config-l2vpn-bg-bd-mac-sec)#threshold
Router(config-l2vpn-bg-bd-mac-sec)#shutdown-recovery-timeout 300
Router(config-l2vpn-bg-bd-mac-sec)#exit
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0.1
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0.2
Router(config-l2vpn-bg-bd-ac)#commit

/* MAC Loop Prevention for PW */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge-group BG1
Router(config-l2vpn-bg)#bridge-domain BD1
Router(config-l2vpn-bg-bd)#mac secure
Router(config-l2vpn-bg-bd-mac-sec)#action shutdown
Router(config-l2vpn-bg-bd-mac-sec)#threshold
Router(config-l2vpn-bg-bd-mac-sec)#shutdown-recovery-timeout 300
Router(config-l2vpn-bg-bd-mac-sec)#exit
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0.3
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#vfi VFI1
Router(config-l2vpn-bg-bd-vfi)#neighbor 192.168.0.4 pw-id 3
Router(config-l2vpn-bg-bd-vfi-pw)#commit

/* MAC Loop Prevention for EVPN */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge-group BG1
Router(config-l2vpn-bg)#bridge-domain BD1
Router(config-l2vpn-bg-bd)#mac secure
Router(config-l2vpn-bg-bd-mac-sec)#action shutdown
Router(config-l2vpn-bg-bd-mac-sec)#threshold
Router(config-l2vpn-bg-bd-mac-sec)#shutdown-recovery-timeout 300
Router(config-l2vpn-bg-bd-mac-sec)#exit
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0.3
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#evi 100
Router(config-l2vpn-bg-bd-evi)#commit

/* Configure move count and move-interval */
Router#configure
Router(config)#evpn
Router(config-evpn)#mac secure
```

```
Router(config-evpn-mac-secure)#move-count 7
Router(config-evpn-mac-secure)#move-interval 30
Router(config-evpn-mac-secure)#commit
```

Running Configuration

This section shows the MAC loop prevention running configuration.

```
/* MAC Loop Prevention for VPLS */
l2vpn
 bridge group BG1
  bridge-domain BD1
   mac
    secure
     action shutdown
     threshold
     shutdown-recovery-timeout 300
    !
   interface GigabitEthernet0/2/0/0.1
   interface GigabitEthernet0/2/0/0.2
  !

/* MAC Loop Prevention for PW */
l2vpn
 bridge group BG1
  bridge-domain BD1
   mac
    secure
     action shutdown
     threshold
     shutdown-recovery-timeout 300
    !
   interface GigabitEthernet0/2/0/0.3
  !
   vfi VFI1
    neighbor 192.168.0.4 pw-id 3

/* MAC Loop Prevention for EVPN */
l2vpn
 bridge group BG1
  bridge-domain BD1
   mac
    secure
     action shutdown
     threshold
     shutdown-recovery-timeout 300
    !
   interface GigabitEthernet0/2/0/0.3
  !
   evi 100
!
/* Configure move-count and move-interval */
evpn
 mac
  secure
   move-count 7
   move-interval 30
 !
```

Verification

Verify that you have successfully configured the MAC Loop Prevention feature. The following show output displays the MAC security information:

```
Router# show l2vpn bridge-doamin detail
Bridge group: bg1, bridge-domain: bd1, id: 0, state: up, ShgId: 0, MSTi: 0
  MAC Secure: enabled, Logging: disabled, Action: shutdown, Threshold: enabled
MAC Secure Shutdown recovery timer : 300
List of ACs:
    AC: interface GigabitEthernet0/2/0/0.1, state is up
    AC: interface GigabitEthernet0/2/0/0.2, state is up
      MAC Secure: enabled, Logging: disabled, Action: shutdown, Threshold: enabled
      MAC Secure Shutdown recovery timer: 300
```

Configuration Examples for Multipoint Layer 2 Services

This section includes these configuration examples:

Multipoint Layer 2 Services Configuration for Provider Edge-to-Provider Edge: Example

These configuration examples show how to create a Layer 2 VFI with a full-mesh of participating Multipoint Layer 2 Services provider edge (PE) nodes.

This configuration example shows how to configure PE 1:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE1-VPLS-A
    interface TenGigE0/0/0/0
    vfi 1
     neighbor 172.16.0.1 pw-id 1
     neighbor 192.168.0.1 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 10.0.0.1 255.0.0.0
```

This configuration example shows how to configure PE 2:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE2-VPLS-A
    interface TenGigE0/0/0/1

    vfi 1
     neighbor 10.0.0.1 pw-id 1
     neighbor 192.168.0.1 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 172.16.0.1 255.240.0.0
```

This configuration example shows how to configure PE 3:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE3-VPLS-A
    interface TenGigE0/0/0/2
    vfi 1
     neighbor 10.0.0.1 pw-id 1
     neighbor 172.16.0.1 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 192.168.0.1 255.255.0.0
```

Multipoint Layer 2 Services Configuration for Provider Edge-to-Customer Edge: Example

This configuration shows how to configure Multipoint Layer 2 Services for a PE-to-CE nodes:

```
configure
 interface TenGigE0/0/0/0
  l2transport---AC interface

  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
```

Displaying MAC Address Withdrawal Fields: Example

This sample output shows the MAC address withdrawal fields:

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain detail

Legend: pp = Partially Programmed.
Bridge group: 222, bridge-domain: 222, id: 0, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 4000, Action: none, Notification: syslog
  MAC limit reached: no
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
  Split Horizon Group: none
  Dynamic ARP Inspection: disabled, Logging: disabled
  IP Source Guard: disabled, Logging: disabled
  DHCPv4 snooping: disabled
  IGMP Snooping: enabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  Storm Control: disabled
  Bridge MTU: 1500
  MIB cvplsConfigIndex: 1
```

```
  Filter MAC addresses:
  P2MP PW: disabled
  Create time: 01/03/2017 11:01:11 (00:21:33 ago)
  No status change since creation
  ACs: 1 (1 up), VFIs: 1, PWs: 1 (1 up), PBBs: 0 (0 up)
  List of ACs:
    AC: TenGigE0/2/0/1.7, state is up
      Type VLAN; Num Ranges: 1
      Outer Tag: 21
      VLAN ranges: [22, 22]
      MTU 1508; XC ID 0x208000b; interworking none
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 4000, Action: none, Notification: syslog
      MAC limit reached: no
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 snooping: disabled
      IGMP Snooping: enabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control: bridge-domain policer
      Static MAC addresses:
      Statistics:
        packets: received 714472608 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 97708776
        bytes: received 88594603392 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 12115888224
        MAC move: 0
      Storm control drop counters:
        packets: broadcast 0, multicast 0, unknown unicast 0
        bytes: broadcast 0, multicast 0, unknown unicast 0
      Dynamic ARP inspection drop counters:
        packets: 0, bytes: 0
      IP source guard drop counters:
        packets: 0, bytes: 0
  List of VFIs:
    VFI 222 (up)
      PW: neighbor 10.0.0.1, PW ID 222, state is up ( established )
        PW class not set, XC ID 0xc000000a
        Encapsulation MPLS, protocol LDP
        Source address 21.21.21.21
        PW type Ethernet, control word disabled, interworking none
        Sequencing not set

        PW Status TLV in use
          MPLS         Local                          Remote
          ------------ ------------------------------ ------------------------
          Label        24017                          24010
          Group ID     0x0                            0x0
          Interface    222                            222
          MTU          1500                           1500
          Control word disabled                       disabled
          PW type      Ethernet                       Ethernet
          VCCV CV type 0x2                            0x2
                       (LSP ping verification)        (LSP ping verification)
          VCCV CC type 0x6                            0x6
                       (router alert label)           (router alert label)
```

```
                (TTL expiry)                    (TTL expiry)
    ------------ ------------------------------ -------------------------
  Incoming Status (PW Status TLV):
    Status code: 0x0 (Up) in Notification message
  MIB cpwVcIndex: 3221225482
  Create time: 01/03/2017 11:01:11 (00:21:33 ago)
  Last time status changed: 01/03/2017 11:21:01 (00:01:43 ago)
  Last time PW went down: 01/03/2017 11:15:21 (00:07:23 ago)
  MAC withdraw messages: sent 0, received 0
  Forward-class: 0
  Static MAC addresses:
  Statistics:
    packets: received 95320440 (unicast 0), sent 425092569
    bytes: received 11819734560 (unicast 0), sent 52711478556
    MAC move: 0
  Storm control drop counters:
    packets: broadcast 0, multicast 0, unknown unicast 0
    bytes: broadcast 0, multicast 0, unknown unicast 0
DHCPv4 snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
VFI Statistics:
  drops: illegal VLAN 0, illegal length 0
```

Bridging on IOS XR Trunk Interfaces: Example

This example shows how to configure a as a simple L2 switch.

Important notes:

Create a bridge domain that has four attachment circuits (AC). Each AC is an IOS XR trunk interface (i.e. not a subinterface/EFP).

- This example assumes that the running config is empty, and that all the components are created.
- This example provides all the necessary steps to configure the to perform switching between the interfaces. However, the commands to prepare the interfaces such as no shut, negotiation auto, etc., have been excluded.
- The bridge domain is in a no shut state, immediately after being created.
- Only trunk (i.e. main) interfaces are used in this example.
- The trunk interfaces are capable of handling tagged (i.e. IEEE 802.1Q) or untagged (i.e. no VLAN header) frames.
- The bridge domain learns, floods, and forwards based on MAC address. This functionality works for frames regardless of tag configuration.
- The bridge domain entity spans the entire system. It is not necessary to place all the bridge domain ACs on a single LC. This applies to any bridge domain configuration.
- The show bundle and the show l2vpn bridge-domain commands are used to verify that the router was configured as expected, and that the commands show the status of the new configurations.
- The ACs in this example use interfaces that are in the admin down state.

Configuration Example

```
RP/0/RSP0/CPU0:router#config
RP/0/RSP0/CPU0:router(config)#interface Bundle-ether10
```

```
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface GigabitEthernet0/2/0/5
RP/0/RSP0/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP0/CPU0:router(config-if)#interface GigabitEthernet0/2/0/6
RP/0/RSP0/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP0/CPU0:router(config-if)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain test-switch
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#commit
RP/0/RSP0/CPU0:Jul 26 10:48:21.320 EDT: config[65751]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'lab'. Use 'show configuration commit changes 1000000973'
to view the changes.
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#end
RP/0/RSP0/CPU0:Jul 26 10:48:21.342 EDT: config[65751]: %MGBL-SYS-5-CONFIG_I : Configured
from console by lab
RP/0/RSP0/CPU0:router#show bundle Bundle-ether10

Bundle-Ether10
  Status:                                  Down
  Local links <active/standby/configured>:   0 / 0 / 2
  Local bandwidth <effective/available>:     0 (0) kbps
  MAC address (source):                    0024.f71e.22eb (Chassis pool)
  Minimum active links / bandwidth:        1 / 1 kbps
  Maximum active links:                    64
  Wait while timer:                        2000 ms
  LACP:                                    Operational
    Flap suppression timer:                Off
  mLACP:                                   Not configured
  IPv4 BFD:                                Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Gi0/2/0/5             Local            Configured   0x8000, 0x0001     1000000
      Link is down
  Gi0/2/0/6             Local            Configured   0x8000, 0x0002     1000000
      Link is down

RP/0/RSP0/CPU0:router#
RP/0/RSP0/CPU0:router#show l2vpn bridge-domain group examples
Bridge group: examples, bridge-domain: test-switch, id: 2000, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 4 (1 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
  List of ACs:
    BE10, state: down, Static MAC addresses: 0
    Gi0/2/0/0, state: up, Static MAC addresses: 0
    Gi0/2/0/1, state: down, Static MAC addresses: 0
    Te0/5/0/1, state: down, Static MAC addresses: 0
  List of VFIs:
RP/0/RSP0/CPU0:router#
```

This table lists the configuration steps (actions) and the corresponding purpose for this example:

Procedure

Step 1 **configure**

Enters global configuration mode.

Step 2 **interface Bundle-ether10**

Creates a new bundle trunk interface.

Step 3 **l2transport**

Changes Bundle-ether10 from an L3 interface to an L2 interface.

Step 4 **interface GigabitEthernet0/2/0/5**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/5.

Step 5 **bundle id 10 mode active**

Establishes GigabitEthernet0/2/0/5 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

Step 6 **interface GigabitEthernet0/2/0/6**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/6.

Step 7 **bundle id 10 mode active**

Establishes GigabitEthernet0/2/0/6 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

Step 8 **interface GigabitEthernet0/2/0/0**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/0.

Step 9 **l2transport**

Change GigabitEthernet0/2/0/0 from an L3 interface to an L2 interface.

Step 10 **interface GigabitEthernet0/2/0/1**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/1.

Step 11 **l2transport**

Change GigabitEthernet0/2/0/1 from an L3 interface to an L2 interface.

Step 12 **interface TenGigE0/1/0/2**

Enters interface configuration mode. Changes configuration mode to act on TenGigE0/1/0/2.

Step 13 **l2transport**

Changes TenGigE0/1/0/2 from an L3 interface to an L2 interface.

Step 14 **l2vpn**

Enters L2VPN configuration mode.

Step 15 **bridge group examples**

Creates the bridge group **examples**.

Step 16 **bridge-domain test-switch**

Creates the bridge domain **test-switch**, that is a member of bridge group **examples**.

Step 17 **interface Bundle-ether10**

Establishes Bundle-ether10 as an AC of bridge domain test-switch.

Step 18 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 19 **interface GigabitEthernet0/2/0/0**

Establishes GigabitEthernet0/2/0/0 as an AC of bridge domain **test-switch**.

Step 20 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 21 **interface GigabitEthernet0/2/0/1**

Establishes GigabitEthernet0/2/0/1 as an AC of bridge domain **test-switch**.

Step 22 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 23 **interface TenGigE0/1/0/2**

Establishes interface TenGigE0/1/0/2 as an AC of bridge domain **test-switch**.

Step 24 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Bridging on Ethernet Flow Points: Example

This example shows how to configure a to perform Layer 2 switching on traffic that passes through Ethernet Flow Points (EFPs). EFP traffic typically has one or more VLAN headers. Although both IOS XR trunks and IOS XR EFPs can be combined as attachment circuits in bridge domains, this example uses EFPs exclusively.

Important notes:

- An EFP is a Layer 2 subinterface. It is always created under a trunk interface. The trunk interface must exist before the EFP is created.
- In an empty configuration, the bundle interface trunk does not exist, but the physical trunk interfaces are automatically configured. Therefore, only the bundle trunk is created.
- In this example the subinterface number and the VLAN IDs are identical, but this is out of convenience, and is not a necessity. They do not need to be the same values.
- The bridge domain test-efp has three attachment circuits (ACs). All the ACs are EFPs.
- Only frames with a VLAN ID of 999 enter the EFPs. This ensures that all the traffic in this bridge domain has the same VLAN encapsulation.
- The ACs in this example use interfaces that are in the admin down state (**unresolved** state). Bridge domains that use nonexistent interfaces as ACs are legal, and the commit for such configurations does not fail. In this case, the status of the bridge domain shows **unresolved** until you configure the missing interface.

Configuration Example

```
RP/0/RSP1/CPU0:router#configure
RP/0/RSP1/CPU0:router(config)#interface Bundle-ether10
RP/0/RSP1/CPU0:router(config-if)#interface Bundle-ether10.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface GigabitEthernet0/6/0/5
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/6
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/7.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface TenGigE0/1/0/2.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#l2vpn
RP/0/RSP1/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP1/CPU0:router(config-l2vpn-bg)#bridge-domain test-efp
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-ether10.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/6/0/7.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface TenGigE0/1/0/2.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#commit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#end
RP/0/RSP1/CPU0:router#
RP/0/RSP1/CPU0:router#show l2vpn bridge group examples
Fri Jul 23 21:56:34.473 UTC Bridge group: examples, bridge-domain: test-efp, id: 0, state:
 up, ShgId: 0, MSTi: 0
Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 3 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
  List of ACs:
    BE10.999, state: down, Static MAC addresses: 0
    Gi0/6/0/7.999, state: unresolved, Static MAC addresses: 0
    Te0/1/0/2.999, state: down, Static MAC addresses: 0
  List of VFIs:
RP/0/RSP1/CPU0:router#
```

This table lists the configuration steps (actions) and the corresponding purpose for this example:

Procedure

Step 1 **configure**

Enters global configuration mode.

Step 2 **interface Bundle-ether10**

Creates a new bundle trunk interface.

Step 3 **interface Bundle-ether10.999 l2transport**

Creates an EFP under the new bundle trunk.

Step 4 **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

Step 5 **interface GigabitEthernet0/6/0/5**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/6/0/5.

Step 6 **bundle id 10 mode active**

Establishes GigabitEthernet0/6/0/5 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

Step 7 **interface GigabitEthernet0/6/0/6**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/6/0/6.

Step 8 **bundle id 10 mode active**

Establishes GigabitEthernet0/6/0/6 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

Step 9 **interface GigabitEthernet0/6/0/7.999 l2transport**

Creates an EFP under GigabitEthernet0/6/0/7.

Step 10 **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

Step 11 **interface TenGigE0/1/0/2.999 l2transport**

Creates an EFP under TenGigE0/1/0/2.

Step 12 **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

Step 13 **l2vpn**

Enters L2VPN configuration mode.

Step 14 **bridge group examples**

Creates the bridge group named **examples**.

Step 15 **bridge-domain test-efp**

Creates the bridge domain named **test-efp**, that is a member of bridge group **examples**.

Step 16 **interface Bundle-ether10.999**

Establishes Bundle-ether10.999 as an AC of the bridge domain named **test-efp**.

Step 17 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 18 **interface GigabitEthernet0/6/0/7.999**

Establishes GigabitEthernet0/6/0/7.999 as an AC of the bridge domain named **test-efp**.

Step 19 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 20 **interface TenGigE0/1/0/2.999**

Establishes interface TenGigE0/1/0/2.999 as an AC of bridge domain named **test-efp**.

Step 21 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

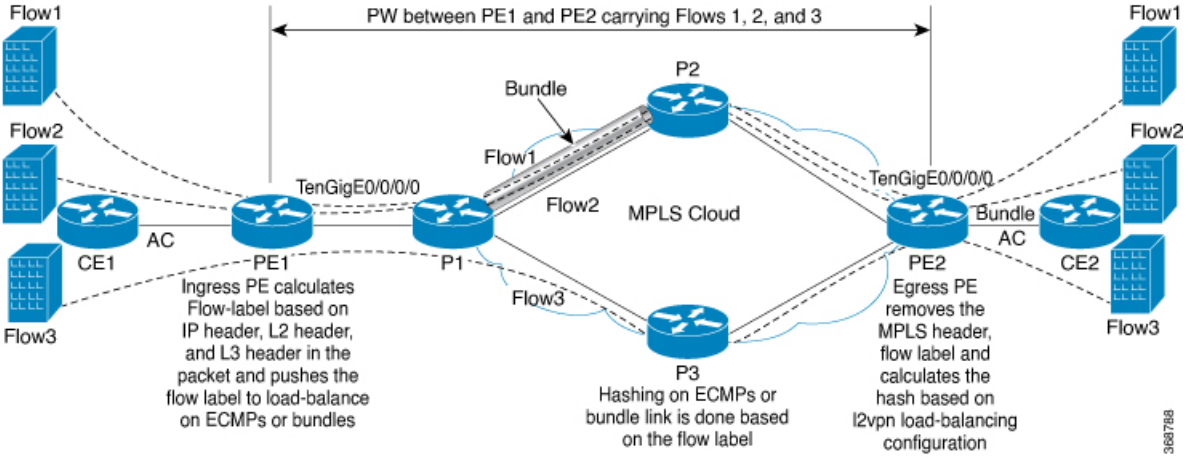
LDP-Based VPLS and VPWS FAT Pseudowire

The LDP-based VPLS and VPWS FAT Pseudowire feature enables provider (P) routers to use the flow-based load balancing to forward traffic between the provider edge (PE) devices. This feature uses Flow-Aware Transport (FAT) of pseudowires (PW) over an MPLS packet switched network for load-balancing traffic across LDP-based signaled pseudowires for Virtual Private LAN Services (VPLS) and Virtual Private Wire Service (VPWS).

FAT PWs provide the capability to identify individual flows within a PW and provide routers the ability to use these flows to load-balance the traffic. FAT PWs are used to load balance the traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on indivisible packet flows entering an imposition PE. This flow label is inserted as the lower most label in the packet. P routers use the flow label for load balancing to provide better traffic distribution across ECMP paths or link-bundled paths in the core. A flow is identified either by the source and destination IP address of the traffic, or the source and destination MAC address of the traffic.

The following figure shows a FAT PW with two flows distributing over ECMPs and bundle links.

Figure 6: FAT PW with Two Flows Distributing over ECMPs and Bundle



An extra label is added to the stack, called the flow label, which is generated for each unique incoming flow on the PE. A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from source and destination MAC addresses, and source and destination IP addresses. The flow label contains the end of label stack (EOS) bit set. The flow label is inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The FAT PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

Core routers perform load balancing using the flow-label in the FAT PW with other information like MAC address and IP address. The flow-label adds greater entropy to improve traffic load balancing. Therefore, it's possible to distribute flows over ECMPs and link bundles.

In this topology, the imposition router, PE1, adds a flow label in the traffic. The disposition router, PE2, allows mixed types of traffic of which some have flow label, others do not. The P router uses flow label to load balance the traffic between the PEs. PE2 ignores the flow label in traffic, and uses one label for all unicast traffic.

Configure LDP-Based VPLS and VPWS FAT Pseudowire

This feature is not supported for traffic across BGP-signaled pseudowires for VPLS and VPWS services.

Configuration Example

Perform this task to configure VPLS and VPWS FAT Pseudowire on both PE1 and PE2.

```
/* Configure LDP-based VPLS FAT Pseudowire */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class vpls
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# load-balancing
Router(config-l2vpn-pwc-mpls-load-bal)# flow-label both
Router(config-l2vpn-pwc-mpls-load-bal)# exit
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg0
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)#
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# vfi 2001
```

```
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.1 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)# pw-class vpls
Router(config-l2vpn-bg-bd-vfi-pw)# commit

/* Configure LDP-based VPWS FAT Pseudowire */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class vpws
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# load-balancing
Router(config-l2vpn-pwc-mpls-load-bal)# flow-label both
Router(config-l2vpn-pwc-mpls-load-bal)# exit
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group vpws
Router(config-l2vpn-xc)# p2p 1001
Router(config-l2vpn-xc-p2p)#
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.1 pw-id 1001
Router(config-l2vpn-xc-p2p-pw)# pw-class vpws
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the running configuration of VPLS and VPWS FAT Pseudowire.

```
/* Configure LDP-based VPLS FAT Pseudowire */
l2vpn
pw-class vpls
  encapsulation mpls
   load-balancing
    flow-label both
   !
  !
bridge group bg0
  bridge-domain bd1

   !
   vfi 2001
    neighbor 192.0.2.1 pw-id 1
     pw-class vpls
    !
    !

/* Configure LDP-based VPWS FAT Pseudowire */
l2vpn
pw-class vpws
  encapsulation mpls
   load-balancing
    flow-label both
   !
  !
!
l2vpn
 xconnect group vpws
  p2p 1001

   neighbor ipv4 192.0.2.1 pw-id 1001
    pw-class vpws
   !
  !
```

Verification

Verify that you have successfully configure the LDP-based VPLS and VPWS FAT Pseudowire feature.

```
/* Verify the LDP-based VPLS FAT Pseudowire configuration */
Router# show l2vpn bridge-domain group bg0 bd-name bd1 detail
Fri May 17 06:00:45.745 UTC
  List of VFIs:
    VFI 1 (up)
      PW: neighbor 192.0.2.1, PW ID 1, state is up ( established )
        PW class vpws, XC ID 0xc0000001
        Encapsulation MPLS, protocol LDP
        Source address 192.0.2.5
        PW type Ethernet, control word disabled, interworking none
        Sequencing not set
        LSP : Up
        Flow Label flags configured (Tx=1,Rx=1), negotiated (Tx=1,Rx=1)

        PW Status TLV in use
          MPLS         Local                          Remote
          ------------ ------------------------------ -------------------------
          Label        24000                          24000
          Group ID     0x0                            0x0
          Interface    1                              1
          MTU          1500                           1500
          Control word disabled                       disabled
          PW type      Ethernet                       Ethernet
          VCCV CV type 0x2                            0x2
                       (LSP ping verification)        (LSP ping verification)
          VCCV CC type 0x6                            0x6
                       (router alert label)           (router alert label)
                       (TTL expiry)                   (TTL expiry)
          ------------ ------------------------------ -------------------------
        Incoming Status (PW Status TLV):
          Status code: 0x0 (Up) in Notification message
        MIB cpwVcIndex: 3221225473
        Create time: 12/05/2019 11:17:59 (4d18h ago)
        Last time status changed: 12/05/2019 11:24:03 (4d18h ago)
        MAC withdraw messages: sent 7, received 9
        Forward-class: 0
        Static MAC addresses:
        Statistics:
          packets: received 0 (unicast 0), sent 0
          bytes: received 0 (unicast 0), sent 0
          MAC move: 0
        Storm control drop counters:
          packets: broadcast 0, multicast 0, unknown unicast 0
          bytes: broadcast 0, multicast 0, unknown unicast 0
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 900 s, Type: inactivity
      MAC limit: 32000, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      E-Tree: Root
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
```

```
      MLD Snooping profile: none
      Storm Control: bridge-domain policer
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none

/* Verify the LDP-based VPWS FAT Pseudowire configuration */
Router# show l2vpn xconnect group vpws detail
Group vpws, XC 1001, state is up; Interworking none
  AC: , state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1001, 1001]
    MTU 1504; XC ID 0x47f; interworking none
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor 192.0.2.1, PW ID 1001, state is up ( established )
    PW class vpws, XC ID 0xc0000548
    Encapsulation MPLS, protocol LDP
    Source address 192.0.2.2
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    LSP : Up
    Flow Label flags configured (Tx=1,Rx=1), negotiated (Tx=1,Rx=1)

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        25011                          25010
      Group ID     0xf000190                      0x228
      Interface
      MTU          1504                           1504
      Control word disabled                       disabled
      PW type      Ethernet                       Ethernet
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x6                            0x6
                   (router alert label)           (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------
    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    MIB cpwVcIndex: 3221226824
    Create time: 17/05/2019 05:52:59 (00:05:22 ago)
    Last time status changed: 17/05/2019 05:53:11 (00:05:10 ago)
    Statistics:
      packets: received 0, sent 0
     bytes: received 0, sent 0
```

Related Topics

- LDP-Based VPLS and VPWS FAT Pseudowire, on page 108

Associated Commands

- show l2vpn xconnect detail

PPPoE traffic-based load balancing

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
PPPoE traffic-based load balance using flow-aware transport labels	Release 7.4.1	This feature allows you to load balance the incoming PPPoE traffic received based on the inner PPPoE payload, source and destination IPv4 or IPv6 header. When you enable this feature, the router generates a unique Flow-Aware Transport (FAT) label for the incoming traffic based on inner IPv4 or IPv6 headers and uses the FAT labels for load balancing the PPPoE traffic. This feature introduces the **hw-module profile load-balance algorithm PPPoE** command.

Point-To-Point Protocol over Ethernet (PPPoE) is a network protocol that encapsulates Point-to-Point Protocol (PPP) frames inside Ethernet frames to allow data communication between two network entities or points.

For most of L2 switched packets, the source and destination MAC address in outer layer header remains the same, it becomes challenging to load balance the incoming PPPoE traffic based on outer headers. So, inner payloads or headers has been taken for hashing.

To enable PPPoE load-balancing feature, use the **hw-module profile load-balance algorithm PPPoE** command.

When you enable the feature, a unique Flow-Aware Transport (FAT) label is generated based on the inner PPPoE payload IPv4 or IPv6 header. The traffic is load balanced using the FAT label.

A flow is defined as a sequence of related packets having the same source and destination pair which is sent from a source Provider Edge (PE) router to a destination PE router. FAT labels provide the capability to identify individual flows within a PW and provide routers the ability to use these flows to load balance the traffic.

For more information on Flow-Aware Transport, see LDP-Based VPLS and VPWS FAT Pseudowire, on page 108.

How PPPoE traffic-based load balancing process works

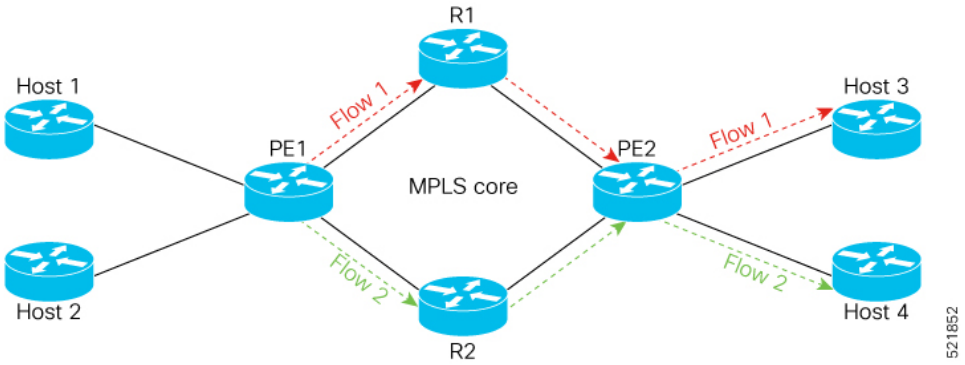
Summary

The key components that are involved in PPPoE traffic-based load balancing process are:

- Source: Host 1 and Host 2
- Destination: Host 3 and Host 4
- PE routers: PE1 and PE2
- Routers: R1 and R2

Workflow

Figure 7: Fat label-based load balancing node workflow



These stages describe how PPPoE traffic-based load balancing works.

1. PE1 receives two packets flows,
   - Flow 1 as traffic from Host 1 to Host 3, and
   - Flow 2 as traffic from Host 2 to Host 4.
2. PE1 performs the following traffic load balancing:

   Before Release 24.4.1,

When...	Then...
PE1 receives traffic from Host 1	PE1 adds a unique FAT label for flow 1 and forwards the traffic to PE2 through R1.
PE2 removes the FAT label	PE2 routes the outgoing traffic in the same direction to Host3.
PE1 receives traffic from Host 2	PE1 checks for the destination address and adds a different FAT label for Flow 2 and forwards the traffic to PE2 through R2.
PE2 removes the FAT label	PE2 routes the outgoing traffic in the same direction to Host4.

Limitations and restrictions for PPPoE traffic-based load balancing

These are limitations and restrictions for PPPoE traffic-based load balancing:

- Supports only transit PPPoE header over Ethernet transport.

- The PPPoE load-balancing feature can't coexist with features such as SRv6 and GUE.

Configure Load balancing for PPPoE Traffic

To enable load balancing for PPPoE traffic:

Prerequisites

You must configure the FAT label for L2VPN, before you enable the PPPoE load-balancing feature. For configuring the FAT label, see LDP-Based VPLS and VPWS FAT Pseudowire, on page 108.

Configuration Example

Note After you enable or disable the PPPoE load-balancing feature, you should reload the line card for this configuration to take effect.

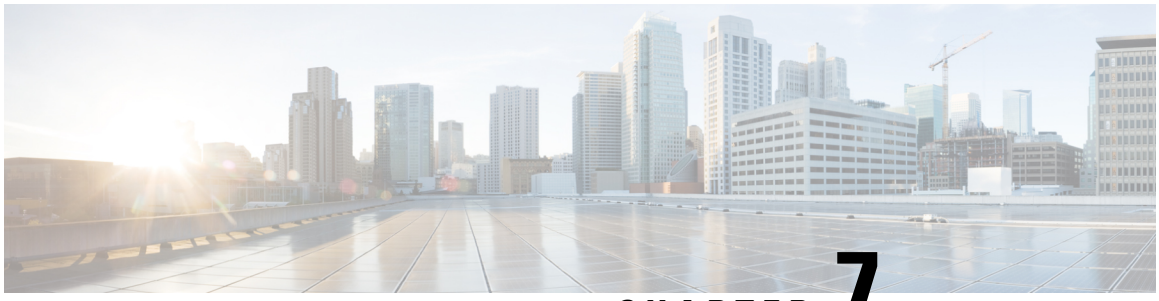
```
Router# configure terminal
Router(config)# hw-module profile load-balance algorithm pppoe
Router(config)# commit
Router(config)# exit
Router# reload
```

To disable load balancing for PPPoE traffic:

```
Router# configure terminal
Router(config)# no hw-module profile load-balance algorithm pppoe
Router(config)# commit
Router(config)# exit
Router# reload
```

Verification

```
Router# Show interface accounting
Bundle-Ether1.1
Protocol Pkts In Chars In Pkts Out Chars Out
IPV4_UNICAST 4494 251664 4494 251664
IPV6_UNICAST 3 228 0 0
MPLS 0 0 252573 35360220
IPV6_ND 12 1032 4 320
CLNS 152 91885 26 2028

Bundle-Ether2.1
Protocol Pkts In Chars In Pkts Out Chars Out
IPV4_UNICAST 4494 251664 4494 251664
IPV6_UNICAST 3 228 0 0
MPLS 0 0 252573 35360220
IPV6_ND 12 1032 4 320
CLNS 155 95324 26 2028
```



CHAPTER 7

Configure Point-to-Point Layer 2 Services

This section introduces you to point-to-point Layer 2 services, and also describes the configuration procedures to implement it.

The following point-to-point services are supported:

- Local Switching—A point-to-point internal circuit on a router, also known as local connect.
- Attachment circuit—A connection between a PE-CE router pair.
- Pseudowires—A virtual point-to-point circuit from one PE router to another. Pseudowires are implemented over the MPLS network.

Note Point-to-point Layer 2 services are also called as MPLS Layer 2 VPNs.

- Ethernet over MPLS , on page 118
- Configure Local Switching Between Attachment Circuits, on page 121
- Configure Static Point-to-Point Connections Using Cross-Connect Circuits, on page 125
- Flexible Cross-Connect Service, on page 127
- Flexible Cross-Connect Service Supported Modes, on page 129
- Configure Preferred Tunnel Path, on page 143
- Multisegment Pseudowire, on page 144
- Configure Multisegment Pseudowire, on page 146
- Split Horizon Groups, on page 150
- G.8032 Ethernet Ring Protection, on page 153
- Configuring G.8032 Ethernet Ring Protection: Example, on page 160
- Pseudowire Redundancy , on page 163
- Configure Pseudowire Redundancy, on page 166
- Access Pseudowire Redundancy, on page 167
- Virtual Circuit Connection Verification on L2VPN, on page 169
- GTP Load Balancing, on page 169
- VPLS over SR-TE and RSVP-TE, on page 171

Ethernet over MPLS

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
Ethernet over MPLS	Release 7.11.2	Introduced in this release on: NCS 5700 fixed port routers. This feature is now supported on Cisco NCS 5700 series fixed port routers and the Cisco NCS 5500 series routers that have the Cisco NC57 line cards installed and operating in the native and compatible modes.
Ethernet over MPLS	Release 6.3.3	Introduced in this release on: NCS 5500 fixed port routers; NCS 5500 modular routers (NCS 5500 line cards) Ethernet-over-MPLS (EoMPLS) provides a tunneling mechanism for Ethernet traffic through an MPLS-enabled Layer 3 core, and encapsulates Ethernet protocol data units (PDUs) inside MPLS packets (using label stacking) to forward them across the MPLS network.

The following table summarizes the load balancing behavior for VPLS and VPWS Ethernet bundle attachment circuits from Release 6.3.3 onwards. In the default configuration mode for load balancing, the parameters used for load balancing through LAG Hashing is provided for disposition traffic flowing from MPLS network, for example, pseudowires to Ethernet attachment circuits.

Note

- VLAN tags (Service and Customer) are not considered for load balancing.
- To enable hashing based on the inner IP header information while doing layer 2 forwarding with inner payload as MPLS, use the **hw-module profile load-balance algorithm layer2** command.

Table 13: Load Balancing Parameters for Ethernet Frames

Ethernet Frame Type	Parameters for Load Balancing Through LAG Hashing
Ethernet Frame with non-IP payload	• Router ID • Input Port • Source Ethernet MAC • Destination Ethernet MAC

Ethernet Frame Type	Parameters for Load Balancing Through LAG Hashing
Ethernet Frame with IP payload	• Router ID • Input Port • Source Ethernet MAC • Destination Ethernet MAC • Source IP Address • Destination IP Address • IP Protocol

Note To enable hashing based on the inner ethernet fields of the Destination MAC and Source MAC addresses for ECMP and bundle member selection, use the **hw-module profile load-balance algorithm inner-L2-field** command.

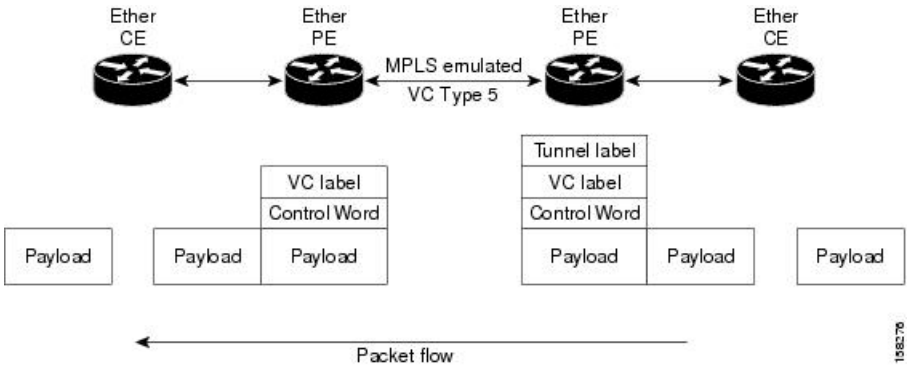
The following sections describe the different modes of implementing EoMPLS.

Ethernet Port Mode

In Ethernet port mode, both ends of a pseudowire are connected to Ethernet ports. In this mode, the port is tunneled over the pseudowire or, using local switching (also known as an *attachment circuit-to-attachment circuit cross-connect*) switches packets or frames from one attachment circuit (AC) to another AC attached to the same PE node.

This figure shows a sample ethernet port mode packet flow:

Figure 8: Ethernet Port Mode Packet Flow

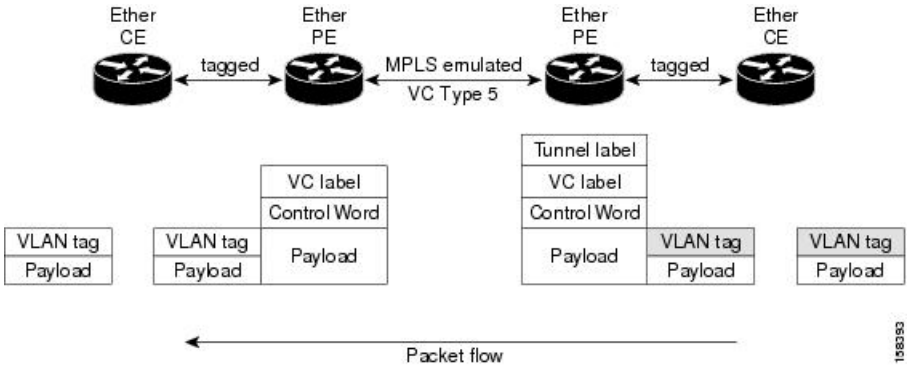


VLAN Mode

In VLAN mode, each VLAN on a customer-end to provider-end link can be configured as a separate L2VPN connection using virtual connection (VC) type 4 or VC type 5. VC type 5 is the default mode.

As illustrated in the following figure, the Ethernet PE associates an internal VLAN-tag to the Ethernet port for switching the traffic internally from the ingress port to the pseudowire; however, before moving traffic into the pseudowire, it removes the internal VLAN tag.

Figure 9: VLAN Mode Packet Flow



At the egress VLAN PE, the PE associates a VLAN tag to the frames coming off of the pseudowire and after switching the traffic internally, it sends out the traffic on an Ethernet trunk port.

Note Because the port is in trunk mode, the VLAN PE doesn't remove the VLAN tag and forwards the frames through the port with the added tag.

QinQ Mode

QinQ is an extension of 802.1Q for specifying multiple 802.1Q tags (IEEE 802.1QinQ VLAN Tag stacking). Layer 3 VPN service termination and L2VPN service transport are enabled over QinQ sub-interfaces.

Cisco NCS 500x Series RouterCisco NCS Routers implement the Layer 2 tunneling or Layer 3 forwarding depending on the sub-interface configuration at provider edge routers. This function only supports up to two QinQ tags on the router:

- Layer 2 QinQ VLANs in L2VPN attachment circuit: QinQ L2VPN attachment circuits are configured under the Layer 2 transport sub-interfaces for point-to-point EoMPLS based cross-connects using both virtual circuit type 4 and type 5 pseudowires and point-to-point local-switching-based cross-connects including full inter-working support of QinQ with 802.1q VLANs and port mode.
- Layer 3 QinQ VLANs: Used as a Layer 3 termination point, both VLANs are removed at the ingress provider edge and added back at the remote provider edge as the frame is forwarded.

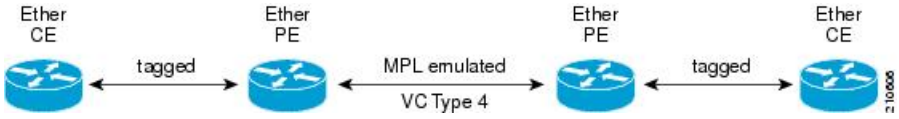
Layer 3 services over QinQ include:

- IPv4 unicast and multicast
- IPv6 unicast and multicast
- MPLS
- Connectionless Network Service (CLNS) for use by Intermediate System-to-Intermediate System (IS-IS) Protocol

In QinQ mode, each CE VLAN is carried into an SP VLAN. QinQ mode should use VC type 5, but VC type 4 is also supported. On each Ethernet PE, you must configure both the inner (CE VLAN) and outer (SP VLAN).

The following figure illustrates QinQ using VC type 4.

Figure 10: EoMPLS over QinQ Mode



Note EoMPLS does not support pseudowire stitching or multi segments.

Configure Local Switching Between Attachment Circuits

Local switching involves the exchange of L2 data from one attachment circuit (AC) to the other, and between two interfaces of the same type on the same router. The two ports configured in a local switching connection form an attachment circuit (AC). A local switching connection works like a bridge domain that has only two bridge ports, where traffic enters from one port of the local connection and leaves through the other.

These are some of the characteristics of Layer 2 local switching:

- Layer 2 local switching uses Layer 2 MAC addresses instead of the Layer 3 IP addresses.
- Because there is no bridging involved in a local connection, there is neither MAC learning nor flooding.
- Unlike in a bridge domain, the ACs in a local connection are not in the UP state if the interface state is DOWN.
- Local switching ACs utilize a full variety of Layer 2 interfaces, including Layer 2 trunk (main) interfaces, bundle interfaces, and EFPs.
- Same-port local switching allows you to switch Layer 2 data between two circuits on the same interface.

Restrictions

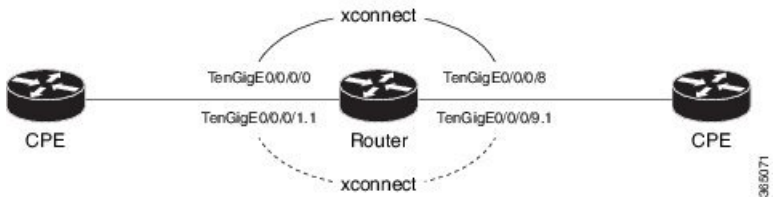
- All sub-interfaces under the given physical port support only two Tag Protocol Identifiers (TPIDs), such as:
  - 0x88a8, 0x8100
  - 0x9100, 0x8100
  - 0x9200, 0x8100
- VLAN and TPID-based ingress packet filtering is not supported.
- Egress TPID rewrite is not supported.
-

Topology

An Attachment Circuit (AC) binds a Customer Edge (CE) router to a Provider Edge (PE) router. The PE router uses a pseudowire over the MPLS network to exchange routes with a remote PE router. To establish a point-to-point connection in a Layer 2 VPN from one Customer Edge (CE) router to another (remote router), a mechanism is required to bind the attachment circuit to the pseudowire. A Cross-Connect Circuit (CCC) is used to bind attachment circuits to pseudowires to emulate a point-to-point connection in a Layer 2 VPN.

The following topology is used for configuration.

Figure 11: Local Switching Between Attachment Circuits



Configuration

To configure an AC-AC local switching, complete the following configuration:

- Enable Layer 2 transport on main interfaces.
- Create sub-interfaces with Layer 2 transport enabled, and specify the respective encapsulation for each.
- Enable local switching between the main interfaces, and between the sub-interfaces.
  - Create a cross-connect group.
  - Create a point-to-point cross connect circuit (CCC).
  - Assign interface(s) to the point-to-point cross connect group.

```
/* Enter the interface configuration mode and configure
  L2 transport on the TenGigE interfaces */
Router# configure
Router(config)# interface TenGigE 0/0/0/1 l2transport
Router(config-if-l2)# no shutdown
Router(config-if)# exit
Router(config)# interface TenGigE 0/0/0/9 l2transport
Router(config-if-l2)# no shutdown
Router(config-if-l2)# commit

/* Configure L2 transport and encapsulation on the VLAN sub-interfaces */
Router# configure
Router(config)# interface TenGigE 0/0/0/0.1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# exit
Router(config)# interface TenGigE 0/0/0/8.1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# commit


/* Configure ethernet link bundles */
Router# configure
Router(config)# interface Bundle-Ether 3
```

```
Router(config-if)# ipv4 address 10.1.3.3 255.0.0.0
Router(config-if)# bundle maximum-active links 32 hot-standby
Router(config-if)# bundle minimum-active links 1
Router(config-if)# bundle minimum-active bandwidth 30000000
Router(config-if)# exit

Router(config)# interface Bundle-Ether 2
Router(config-if)# ipv4 address 10.1.2.2 255.0.0.0
Router(config-if)# bundle maximum-active links 32 hot-standby
Router(config-if)# bundle minimum-active links 1
Router(config-if)# bundle minimum-active bandwidth 30000000
Router(config-if)# exit

/* Add physical interfaces to the ethernet link bundles */
Router(config)# interface TenGigE 0/0/0/1
Router(config-if)# bundle id 3 mode on
Router(config-if)# no shutdown
Router(config)# exit
Router(config)# interface TenGigE 0/0/0/2
Router(config-if)# bundle id 3 mode on
Router(config-if)# no shutdown
Router(config)# exit
Router(config)# interface TenGigE 0/0/0/9
Router(config-if)# bundle id 2 mode on
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface TenGigE 0/0/0/8
Router(config-if)# bundle id 2 mode on
Router(config-if)# no shutdown
Router(config-if)# exit

/* Configure Layer 2 transport on the ethernet link bundles */
Router(config)# interface Bundle-Ether 3 l2transport
Router(config-if-l2)# no shutdown
Router(config-if)# exit
Router(config)# interface Bundle-Ether 2 l2transport
Router(config-if-l2)# no shutdown
Router(config-if-l2)# commit

/* Configure local switching on the TenGigE Interfaces */
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p XCON1_P2P3
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/9
Router(config-l2vpn-xc-p2p)# commit
Router(config-l2vpn-xc-p2p)# exit

/* Configure local switching on the VLAN sub-interfaces */
Router(config-l2vpn-xc)# p2p XCON1_P2P1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/0.1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/8.1
Router(config-l2vpn-xc-p2p)# commit
Router(config-l2vpn-xc-p2p)# exit

/* Configure local switching on ethernet link bundles */
Router(config-l2vpn-xc)# p2p XCON1_P2P4
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 3
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 2
Router(config-l2vpn-xc-p2p)# commit
```

Running Configuration

```
configure
 interface tenGigE 0/0/0/1 l2transport
 !
 interface tenGigE 0/0/0/9 l2transport
 !
!

interface tenGigE 0/0/0/0.1 l2transport
 encapsulation dot1q 5
 rewrite ingress tag push dot1q 20 symmetric
 !
interface tenGigE 0/0/0/8.1 l2transport
  encapsulation dot1q 5
 !
interface Bundle-Ether 3 l2transport
 !
interface Bundle-Ether 2 l2transport
!

l2vpn
 xconnect group XCON1
   p2p XCON1_P2P3
    interface TenGigE0/0/0/1
    interface TenGigE0/0/0/9
    !
   !
 !
l2vpn
 xconnect group XCON1
   p2p XCON1_P2P1
    interface TenGigE0/0/0/0.1
    interface TenGigE0/0/0/8.1
    !
   !
 !
l2vpn
 xconnect group XCON1
   p2p XCON1_P2P4
    interface Bundle-Ether 3
    interface Bundle-Ether 2
    !
   !
 !
```

Verification

- Verify if the configured cross-connect is UP

```
router# show l2vpn xconnect brief

Locally Switching

  Like-to-Like                          UP        DOWN         UNR

    EFP                                  1           0           0

    Total                                1           0           0
```

```
  Total                                1          0          0

Total: 1 UP, 0 DOWN, 0 UNRESOLVED


router# show l2vpn xconnect

Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                   Segment 1                         Segment 2
Group      Name       ST   Description            ST        Description       ST
------------------------   ------------------------------------------------------------
XCON1      XCON_P2P1 UP    Te0/0/0/1      UP  Te0/0/0/9      UP
XCON1      XCON_P2P3 UP    Te0/0/0/0.1    UP  Te0/0/0/8.1    UP
------------------------------------------------------------------------------------------
```

Associated Commands

- interface (p2p)
- l2vpn
- p2p
- xconnect group

Configure Static Point-to-Point Connections Using Cross-Connect Circuits

This section describes how you can configure static point-to-point cross connects in a Layer 2 VPN.

Requirements and Limitations

Before you can configure a cross-connect circuit in a Layer 2 VPN, ensure that the following requirements are met:

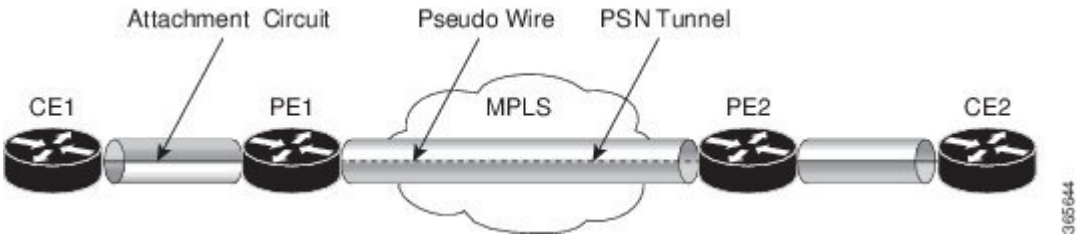
- The CE and PE routers are configured to operate in the MPLS network.
- The name of a cross-connect circuit is configured to identify a pair of PE routers and must be unique within the cross-connect group.
- A segment (an attachment circuit or pseudowire) is unique and can belong only to a single cross-connect circuit.
- A static virtual circuit local label is globally unique and can be used in only one pseudowire.
- A maximum of 16,000 cross-connects can be configured per PE router.

Note Static pseudowire connections do not use LDP for signaling.

Topology

The following topology is used to configure static cross-connect circuits in a Layer 2 VPN.

Figure 12: Static Cross-Connect Circuits in a Layer 2 VPN



Configuration

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface gigabitethernet0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor 10.165.100.151 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 50 remote 40
Router(config-l2vpn-xc-p2p-pw)# commit
```

```
/*Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface gigabitethernet0/2/0/0.4
Router(config-l2vpn-xc-p2p)# neighbor 10.165.200.254 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 40 remote 50
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* On PE1 */
!
l2vpn
 xconnect group XCON1
  p2p xc1
   interface GigabitEthernet0/1/0/0.1
   neighbor ipv4 10.165.100.151 pw-id 100
    mpls static label local 50 remote 40
!
```

```
/* On PE2 */
!
l2vpn
 xconnect group XCON2
  p2p xc1
   interface GigabitEthernet0/2/0/0.4
   neighbor ipv4 10.165.200.254 pw-id 100
    mpls static label local 40 remote 50
  !
```

Verification

```
/* Verify the static cross connect on PE1 */
Router# show l2vpn xconnect
Tue Apr 12 20:18:02.971 IST
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                       Segment 2
Group      Name       ST    Description            ST       Description            ST
-------------------------   ------------------------------  ------------------------------
XCON1      xc1        UP    Gi0/1/0/0.1            UP       10.165.100.151  100    UP
-------------------------------------------------------------------------------------------

/* Verify the static cross connect on PE2 */

Router# show l2vpn xconnect
Tue Apr 12 20:18:02.971 IST
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                       Segment 2
Group      Name       ST    Description            ST       Description            ST
-------------------------   ------------------------------  ------------------------------
XCON2      xc1        UP    Gi0/2/0/0.4            UP       10.165.200.254  100    UP
-------------------------------------------------------------------------------------------
```

Flexible Cross-Connect Service

Table 14: Feature History Table

Feature Name	Release Information	Feature Description
Flexible Cross-Connect Service	Release 7.4.1	This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.

The flexible cross-connect service feature enables aggregation of attachment circuits (ACs) across multiple endpoints in a single Ethernet VPN Virtual Private Wire Service (EVPN-VPWS) service instance, on the same Provider Edge (PE). ACs are represented either by a single VLAN tag or double VLAN tags. The associated AC with the same VLAN tag(s) on the remote PE is cross-connected. The VLAN tags define the matching criteria to be used in order to map the frames on an interface to the appropriate service instance. As a result, the VLAN rewrite value must be unique within the flexible cross-connect (FXC) instance to create the lookup table. The VLAN tags can be made unique using the rewrite configuration. The lookup table helps determine the path to be taken to forward the traffic to the corresponding destination AC. This feature reduces the number of tunnels by muxing VLANs across many interfaces. It also reduces the number of MPLS labels used by a router. This feature supports both single-homing and multi-homing.

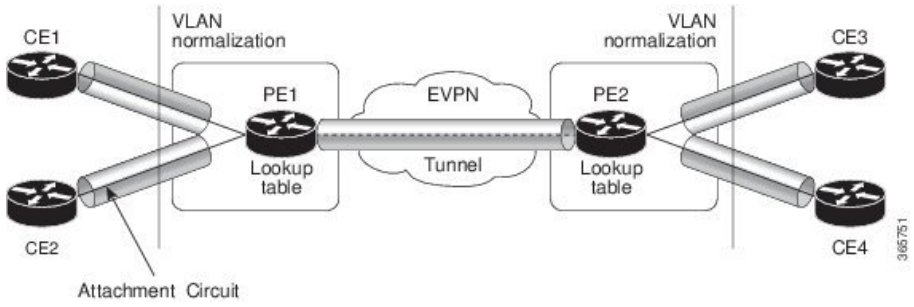
Restrictions for NCS 5700 routers and line cards for FXC services:

1. The router does not support a configuration with the inner VLAN tag set to **any**.
2. All ACs under a given FXC should have the same number of VLAN tags after a rewrite operation. For example, the router does not support a configuration that contains one AC with two VLAN tags, and another AC with a single VLAN tag.

Flexible Cross-Connect Service - Single-Homed

Consider the following topology in which the traffic flows from CE1 and CE2 to PE1 through ACs. ACs are aggregated across multiple endpoints on the same PE. The VLAN (rewrite) creates the lookup table based on the rewrite configured at AC interfaces on PE1. PE1 uses BGP to exchange routes with PE2 and creates a tunnel over EVPN MPLS network. The VLANs (rewrite) on PE2 must match the rewrite configured on PE1. Based on the rewrite tag, the PE2 forwards the traffic to the corresponding ACs. For example, if the ACs for CE1 and CE3 are configured with the same rewrite tag, the end-to-end traffic is sent from CE1 to CE3.

Figure 13: Flexible Cross-Connect Service

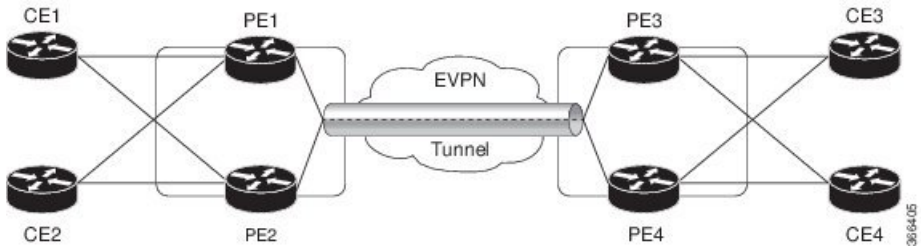


Flexible Cross-Connect Service - Multi-Homed

The Flexible Cross-Connect Service multihoming capability enables you to connect a customer edge (CE) device to two or more provider edge (PE) devices to provide load balancing and redundant connectivity. Flow-based load balancing is used to send the traffic between PEs and CEs. Flow-based load balancing is used to connect source and remote PEs as well. The customer edge device is connected to PE through Ethernet bundle interface.

When a CE device is multi-homed to two or more PEs and when all PEs can forward traffic to and from the multi-homed device for the VLAN, then such multihoming is referred to as all-active multihoming.

Figure 14: Flexible Cross-Connect Service Multi-Homed



Consider the topology in which CE1 and CE2 are multi-homed to PE1 and PE2; CE3 and CE4 are multi-homed to PE3 and PE4. PE1 and PE2 advertise Ethernet A-D Ethernet Segment (ES-EAD) route to remote PEs that is PE3 and PE4. Similarly, PE3 and PE4 advertise ES-EAD route to remote PEs that is PE1 and PE2. The ES-EAD route is advertised per main interface.

Consider a traffic flow from CE1 to CE3. Traffic is sent to either PE1 or PE2. The selection of path is dependent on the CE implementation for forwarding over a LAG. Traffic is encapsulated at each PE and forwarded to the remote PEs (PE 3 and PE4) through the MPLS tunnel. Selection of the destination PE is established by flow-based load balancing. PE3 and PE4 send the traffic to CE3. The selection of path from PE3 or PE4 to CE3 is established by flow-based load balancing.

Flexible Cross-Connect Service Supported Modes

The Flexible Cross-Connect Service feature supports the following modes:

- VLAN Unaware
- VLAN Aware
- Local Switching

VLAN Unaware

In this mode of operation, a group of normalized ACs on a single ES that are destined to a single endpoint or interface are multiplexed into a single EVPN VPWS tunnel represented by a single VPWS service ID. The VLAN-Unaware FXC reduces the number of BGP states. VLAN failure is not signaled over BGP. One EVI/EAD route is advertised per VLAN-Unaware FXC rather than per AC. In multihoming scenario, there will be ES-EAD route as well. EVI can be shared with other VLAN-Unaware FXC or EVPN VPWS. If AC goes down on PE1, the remote PE is not be informed of the failure, and PE3 or PE4 continues to send the traffic to PE1 and PE2 resulting in packet drop.

Multihoming is supported on VLAN Unaware FXC only if all ACs belong to the same main interface.

If you have multiple ESIs, regardless of whether it is a zero-ESI or non-zero ESI, only ESI 0 is signalled. Only single-home mode is supported in this scenario.

Configure Single-Homed Flexible Cross-Connect Service using VLAN Unaware

This section describes how you can configure single-homed flexible cross-connect service using VLAN unaware

```
/* Configure PE1 */
Router# configure
Router(config)# interface GigabitEthernet 0/2/0/3.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100
 symetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface GigabitEthernet 0/2/0/0.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200
 symetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxs1
Router(config-l2vpn-fxs-vu)# interface GigabitEthernet 0/2/0/3.1
Router(config-l2vpn-fxs-vu)# interface GigabitEthernet 0/2/0/0.1
Router(config-l2vpn-fxs-vu)# neighbor evpn evi 1 target 1
Router(config-l2vpn-fxs-vu)# commit

/* Configure PE2 */
Router# configure
Router(config)# interface GigabitEthernet 0/0/0/3.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100
 symetric
```

```
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface GigabitEthernet 0/0/0/0.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200
 symetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxs1
Router(config-l2vpn-fxs-vu)# interface GigabitEthernet 0/0/0/3.1
Router(config-l2vpn-fxs-vu)# interface GigabitEthernet 0/0/0/0.1
Router(config-l2vpn-fxs-vu)# neighbor evpn evi 1 target 1
Router(config-l2vpn-fxs-vu)# commit
```

Running Configuration

```
/* On PE1 */
!
Configure
interface GigabitEthernet 0/2/0/3.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 symetric
!

Configure
interface GigabitEthernet 0/2/0/0.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symetric
!

l2vpn
  flexible-xconnect-service vlan-unaware fxs1
   interface GigabitEthernet 0/2/0/3.1
   interface GigabitEthernet0/2/0/0.1
   neighbor evpn evi 1 target 1

!

/* On PE2 */
!
Configure
interface GigabitEthernet 0/0/0/3.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 symetric
!

Configure
interface GigabitEthernet 0/0/0/0.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symetric
!

l2vpn
  flexible-xconnect-service vlan-unaware fxs1
   interface GigabitEthernet 0/0/0/3.1
   interface GigabitEthernet0/0/0/0.1
   neighbor evpn evi 1 target 1

!
```

Configure Multi-Homed Flexible Cross-Connect Service using VLAN Unaware

This section describes how you can configure multi-homed flexible cross-connect service using VLAN unaware.

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc1_16
Router(config-l2vpn-fxs)# interface Bundle-Ether10.11
Router(config-l2vpn-fxs)# interface Bundle-Ether10.12
Router(config-l2vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether10.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether10.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# evpn
Router (config-evpn)# interface Bundle-Ether10
Router (config-evpn-ac)# ethernet-segment
Router (config-evpn-ac-es)# identifier type  0 00.01.00.ac.ce.55.00.0a.00
Router (config-evpn-ac-es)# commit

/* Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc1_16
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether10.11
Router(config-l2vpn-fxs)# interface Bundle-Ether10.12
Router(config-l2vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether10.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether10.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# evpn
Router (config-evpn)# interface Bundle-Ether10
Router (config-evpn-ac)# ethernet-segment
Router (config-evpn-ac-es)# identifier type  0 00.01.00.ac.ce.55.00.0a.00
Router (config-evpn-ac-es)# commit

/* Configure PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc1_16
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether20.11
Router(config-l2vpn-fxs)# interface Bundle-Ether20.12
```

```
Router(config-l2vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether20.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-subif)# exit
Router(config)# interface Bundle-Ether20.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
Router(config-subif)# exit
Router(config)# evpn
Router (config-evpn)# interface Bundle-Ether20
Router (config-evpn-ac)# ethernet-segment
Router (config-evpn-ac-es)# identifier type  0 00.01.00.ac.ce.55.00.14.00
Router (config-evpn-ac-es)# commit

/* Configure PE4 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc1_16
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether20.11
Router(config-l2vpn-fxs)# interface Bundle-Ether20.12
Router(config-l2vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether20.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-subif)# exit
Router(config)# interface Bundle-Ether20.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
Router(config-subif)# exit
Router(config)# evpn
Router (config-evpn)# interface Bundle-Ether20
Router (config-evpn-ac)# ethernet-segment
Router (config-evpn-ac-es)# identifier type  0 00.01.00.ac.ce.55.00.14.00
Router (config-evpn-ac-es)# commit
```

Running Configuration

```
/* On PE1 */

configure
l2vpn
 flexible-xconnect-service vlan-unaware fxc1_16
  interface Bundle-Ether10.11
  interface Bundle-Ether10.12
  neighbor evpn evi 1 target 16

!

configure
interface Bundle-Ether10.11 l2transport
  encapsulation dot1q 1
```

```
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!


configure
interface Bundle-Ether10.12 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

!

evpn
  interface Bundle-Ether10
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.0a.00

!

/* On PE2 */

configure
l2vpn
 flexible-xconnect-service vlan-unaware fxc1_16
  interface Bundle-Ether10.11
  interface Bundle-Ether10.12
  neighbor evpn evi 1 target 16

!

configure
interface Bundle-Ether10.11 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!


configure
interface Bundle-Ether10.12 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

!

evpn
  interface Bundle-Ether10
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.0a.00

!

/* On PE3 */

configure
l2vpn
 flexible-xconnect-service vlan-unaware fxc1_16
  interface Bundle-Ether20.11
  interface Bundle-Ether20.12
  neighbor evpn evi 1 target 16

!

configure
interface Bundle-Ether20.11 l2transport
  encapsulation dot1q 1
```

```
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether20.12 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

!

evpn
  interface Bundle-Ether20
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.14.00

!

/* On PE4 */

configure
l2vpn
 flexible-xconnect-service vlan-unaware fxc1_16
  interface Bundle-Ether20.11
  interface Bundle-Ether20.12
  neighbor evpn evi 1 target 16

!

configure
interface Bundle-Ether20.11 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether20.12 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

!

evpn
  interface Bundle-Ether20
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.14.00

!
```

VLAN Aware

In this mode of operation, normalized ACs across different Ethernet segments and interfaces are multiplexed into a single EVPN VPWS service tunnel. This single tunnel is represented by many VPWS service IDs (one per normalized VLAN ID (VID)) and these normalized VIDs are signaled using EVPN BGP. The VLAN-Aware FXC reduces the number of PWs; but it does not reduce the BGP states. VLAN failure is signaled over BGP. The VLAN-Aware FXC advertises one EAD route per AC rather than per FXC. For VLAN-Aware FXC, the EVI must be unique to the FXC itself. It cannot be shared with any other service such as FXC, EVPN, EVPN-VPWS, PBB-EVPN. If a single AC goes down on PE1, it withdraws only the EAD routes associated with that AC. The ES-EAD route will also be withdrawn on failure of the main interface. The equal-cost multipath (ECMP) on PE3 or PE4 stops sending traffic for this AC to PE1, and only sends it to PE2.

For the same VLAN-Aware FXC, you can either configure all non-zero ESIs or all zero-ESIs. You cannot configure both zero-ESI and non-zero ESI for the same VLAN-Aware FXC. This applies only to single-home mode.

Configure Single-Homed Flexible Cross-Connect using VLAN Aware

This section describes how you can configure single-homed flexible cross-connect service using VLAN aware.

```
/* Configure PE1 */
Router# configure
Router(config)#  interface GigabitEthernet 0/2/0/7.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100
 symetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)#  interface GigabitEthernet 0/2/0/7.2 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200
 symetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 4
Router(config-l2vpn-fxs-va)# interface GigabitEthernet 0/2/0/7.1
Router(config-l2vpn-fxs-va)# interface GigabitEthernet 0/2/0/7.2
Router(config-l2vpn-fxs-va)# commit

/* Configure PE2 */
Router# configure
Router(config)#  interface GigabitEthernet 0/0/0/7.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100
 symetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)#  interface GigabitEthernet 0/0/0/7.2 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200
 symetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 4
Router(config-l2vpn-fxs-va)# interface GigabitEthernet 0/0/0/7.1
Router(config-l2vpn-fxs-va)# interface GigabitEthernet 0/0/0/7.2
Router(config-l2vpn-fxs-va )# commit
```

Running Configuration

```
/* On PE1 */
!
Configure
interface GigabitEthernet 0/2/0/7.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 symetric
!

Configure
interface GigabitEthernet 0/2/0/7.2 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symetric
```

```
!

l2vpn
  flexible-xconnect-service vlan-aware evi 4
   interface GigabitEthernet 0/2/0/7.1
   interface GigabitEthernet 0/2/0/7.2


!

/* On PE2 */
!
Configure
interface GigabitEthernet 0/0/0/7.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 symetric
!

Configure
interface GigabitEthernet 0/0/0/7.2 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symetric
!

l2vpn
  flexible-xconnect-service vlan-aware evi 4
   interface GigabitEthernet 0/0/0/7.1
   interface GigabitEthernet 0/0/0/7.2

!
```

Configure Multi-Homed Flexible Cross-Connect Service using VLAN Aware

This section describes how you can configure multi-homed flexible cross-connect service using VLAN aware.

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-l2vpn-fxs-va)# interface Bundle-Ether2.1
Router(config-l2vpn-fxs-va)# interface Bundle-Ether3.1
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs-va)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether2.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 22.33.44.55.66.77.88.99.aa
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether3
Router(config-evpn-ac)# ethernet-segment
```

```
Router(config-evpn-ac-es)# identifier type 0 33.44.55.66.77.88.99.aa.bb
Router(config-evpn-ac-es)# commit

/* Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-l2vpn-fxs-va)# interface Bundle-Ether2.1
Router(config-l2vpn-fxs-va)# interface Bundle-Ether3.1
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs-va)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether2.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 22.33.44.55.66.77.88.99.aa
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether3
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 33.44.55.66.77.88.99.aa.bb
Router(config-evpn-ac-es)# commit

/* Configure PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-l2vpn-fxs-va)# interface Bundle-Ether4.1
Router(config-l2vpn-fxs-va)# interface Bundle-Ether5.1
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs-va)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether4.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether5.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether4
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether5
Router(config-evpn-ac)# ethernet-segment
```

```
Router(config-evpn-ac-es)# identifier type identifier type 0 00.01.00.ac.ce.55.00.15.00
Router(config-evpn-ac-es)# commit

/* Configure PE4 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-l2vpn-fxs-va)# interface Bundle-Ether4.1
Router(config-l2vpn-fxs-va)# interface Bundle-Ether5.1
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs-va)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether4.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether5.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether4
Router(config-evpn-ac)# ethernet-segment
Router config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether5
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type identifier type 0 00.01.00.ac.ce.55.00.15.00
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
/* On PE1 */
!
configure
l2vpn
 flexible-xconnect-service vlan-aware evi 6
  interface Bundle-Ether2.1
  interface Bundle-Ether3.1

!

configure
interface Bundle-Ether2.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether3.1 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
!

evpn
  interface Bundle-Ether2
   ethernet-segment identifier type 0 22.33.44.55.66.77.88.99.aa
  interface Bundle-Ether3
```

```
   ethernet-segment identifier type 0 33.44.55.66.77.88.99.aa.bb

!

/* On PE2 */
!
configure
l2vpn
 flexible-xconnect-service vlan-aware evi 6
  interface Bundle-Ether2.1
  interface Bundle-Ether3.1

!

configure
interface Bundle-Ether2.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether3.1 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
!

evpn
  interface Bundle-Ether2
   ethernet-segment identifier type 0 22.33.44.55.66.77.88.99.aa
  interface Bundle-Ether3
   ethernet-segment identifier type 0 33.44.55.66.77.88.99.aa.bb

!

/* On PE3 */
!
configure
l2vpn
 flexible-xconnect-service vlan-aware evi 6
  interface Bundle-Ether4.1
  interface Bundle-Ether5.1

!

configure
interface Bundle-Ether4.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether5.1 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
!

evpn
  interface Bundle-Ether4
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.14.00
  interface Bundle-Ether5
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.15.00
```

```
!

/* On PE4 */
!
configure
l2vpn
 flexible-xconnect-service vlan-aware evi 6
  interface Bundle-Ether4.1
  interface Bundle-Ether5.1

!

configure
interface Bundle-Ether4.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether5.1 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
!

evpn
  interface Bundle-Ether4
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.14.00
  interface Bundle-Ether5
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.15.00

!
```

Local Switching

Traffic between the two ACs is locally switched within the PE when two ACs belonging to different Ethernet Segment have the same normalization VLANs. Local switching is supported only on FXC VLAN-aware.

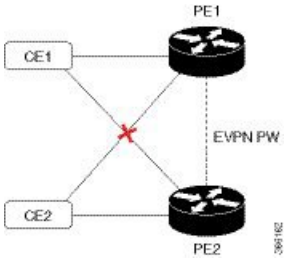
Consider a topology in which CE1 and CE2 have different Ethernet Segment. However, they both have the same normalized VLANs. Hence, when a traffic is sent from CE1 to CE2, PE1 routes the traffic to CE2 using local switching.

If there is a failure and when the link from CE1 to PE1 goes down, PE1 sends the traffic to PE2 through EVPN pseudowire. Then the PE2 sends the traffic to CE2.

CE1 and CE2 must be on different non-zero ESI.

Note EVPN FXC local switching is not supported. VPWS local switching is supported on routers that have Cisco NC57 line cards installed and operate in compatibility mode.

Figure 15: Local Switching



Configure Multi-Homed Flexible Cross-Connect Service using Local Switching

This section describes how you can configure multi-homed flexible cross-connect service using local switching.

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-l2vpn-fxs-va)# interface Bundle-Ether2.1
Router(config-l2vpn-fxs-va)# interface Bundle-Ether3.1
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether2.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3
symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3
symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 22.33.44.55.66.77.88.99.aa
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether3
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 33.44.55.66.77.88.99.aa.bb
Router(config-evpn-ac-es)# commit

/* Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-l2vpn-fxs-va)# interface Bundle-Ether2.1
Router(config-l2vpn-fxs-va)# interface Bundle-Ether3.1
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether2.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3
```

```
symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3
symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 22.33.44.55.66.77.88.99.aa
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether3
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 33.44.55.66.77.88.99.aa.bb
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
/* On PE1 */

configure
l2vpn
 flexible-xconnect-service vlan-aware evi 6
  interface Bundle-Ether2.1
  interface Bundle-Ether3.1

!

configure
interface Bundle-Ether2.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric

!

configure
interface Bundle-Ether3.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric
!

evpn
  interface Bundle-Ether2
   ethernet-segment identifier type 0 22.33.44.55.66.77.88.99.aa
  interface Bundle-Ether3
   ethernet-segment identifier type 0 33.44.55.66.77.88.99.aa.bb

!

/* On PE2 */

configure
l2vpn
 flexible-xconnect-service vlan-aware evi 6
  interface Bundle-Ether2.1
  interface Bundle-Ether3.1

!
```

```
configure
interface Bundle-Ether2.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric

!

configure
interface Bundle-Ether3.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric
!

evpn
  interface Bundle-Ether2
   ethernet-segment identifier type 0 22.33.44.55.66.77.88.99.aa
  interface Bundle-Ether3
   ethernet-segment identifier type 0 33.44.55.66.77.88.99.aa.bb


!
```

Configure Preferred Tunnel Path

Preferred tunnel path functionality lets you map pseudowires to specific traffic-engineering tunnels. Attachment circuits are cross-connected to specific MPLS traffic engineering tunnel interfaces instead of remote PE router IP addresses (reachable using IGP or LDP).

When using a preferred tunnel path, it is assumed that the traffic engineering tunnel that transports the Layer 2 traffic runs between the two PE routers (that is, its head starts at the imposition PE router and its tail terminates on the disposition PE router).

Configuration

```
/* Enter global configuration mode */
Router# configure
Router(config)# l2vpn

/* Configure pseudowire class name */
Router(config-l2vpn)# pw-class path1

/* Configure MPLS encapsulation for the pseudowire */
Router(config-l2vpn-pwc)# encapsulation mpls


/* Configure preferred path tunnel settings.
If fallback disable configuration is used, and when
 the TE/ tunnel is configured,
 if the preferred path goes down,
 the corresponding pseudowire can also go down. */

Router(config-l2vpn-pwc-encap-mpls)# preferred-path
                  interface tunnel-te 11 fallback disable

/* Commit your configuration */
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# commit
```

Running Configuration

```
Router# show running-configuration
!
l2vpn
 pw-class path1
  encapsulation mpls
   preferred-path interface tunnel-te 11 fallback disable
  !
 !
!
```

Multisegment Pseudowire

The Multisegment Pseudowire feature allows you to extend L2VPN pseudowires across an inter-AS boundary or across two separate MPLS networks. A multisegment pseudowire connects two or more contiguous pseudowire segments to form an end-to-end multi-hop pseudowire as a single point-to-point pseudowire. These segments act as a single pseudowire, allowing you to:

- Manage the end-to-end service by separating administrative or provisioning domains.
- Keep IP addresses of provider edge (PE) nodes private across interautonomous system (inter-AS) boundaries. Use IP address of autonomous system boundary routers (ASBRs) and treat them as pseudowire aggregation routers. The ASBRs join the pseudowires of the two domains.

A multisegment pseudowire can span either an inter-AS boundary or two multiprotocol label switching (MPLS) networks.

A pseudowire is a tunnel between two PE nodes. There are two types of PE nodes:

- A Switching PE (S-PE) node
  - Terminates PSN tunnels of the preceding and succeeding pseudowire segments in a multisegment pseudowire.
  - Switches control and data planes of the preceding and succeeding pseudowire segments of the multisegment pseudowire.
- A Terminating PE (T-PE) node
  - Located at both the first and last segments of a multisegment pseudowire.
  - Where customer-facing attachment circuits (ACs) are bound to a pseudowire forwarder.

Note Every end of a multisegment pseudowire must terminate at a T-PE.

A multisegment pseudowire is used in two general cases when:

- It is not possible to establish a PW control channel between the source and destination PE nodes.

  For the PW control channel to be established, the remote PE node must be accessible. Sometimes, the local PE node may not be able to access the remote node due to topology, operational, or security constraints.

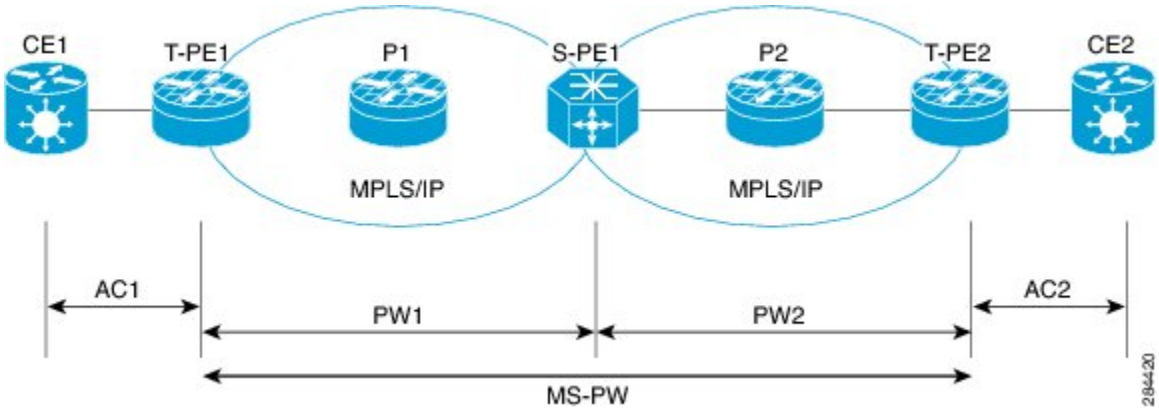
A multisegment pseudowire dynamically builds two discrete pseudowire segments and performs a pseudowire switching to establish a PW control channel between the source and destination PE nodes.

- Pseudowire Edge To Edge Emulation (PWE3) signaling and encapsulation protocols are different.

  The PE nodes are connected to networks employing different PW signaling and encapsulation protocols. Sometimes, it is not possible to use a single segment PW.

  A multisegment pseudowire, with the appropriate interworking performed at the PW switching points, enables PW connectivity between the PE nodes in the network.

Figure 16: Multisegment Pseudowire



The topology shows MS-PW stitching between PW1 and PW2. You can configure a set of two or more contiguous PW segments that behave and function as a single point-to-point PW. You can configure static or dynamic multisegment PW (MS-PW). The maximum number of contiguous PW segments is 254. Each end of an MS-PW terminates on a T-PE. A switching PE (S-PE) terminates the PSN tunnels of the preceding and succeeding PW segments in an MS-PW. The S-PE switches the control and data planes of the preceding and succeeding PW segments of the MS-PW. An MS-PW is up when all the SS-PWs are up.

Restrictions

You must consider the following restrictions while configuring the Multisegment Pseudowire feature:

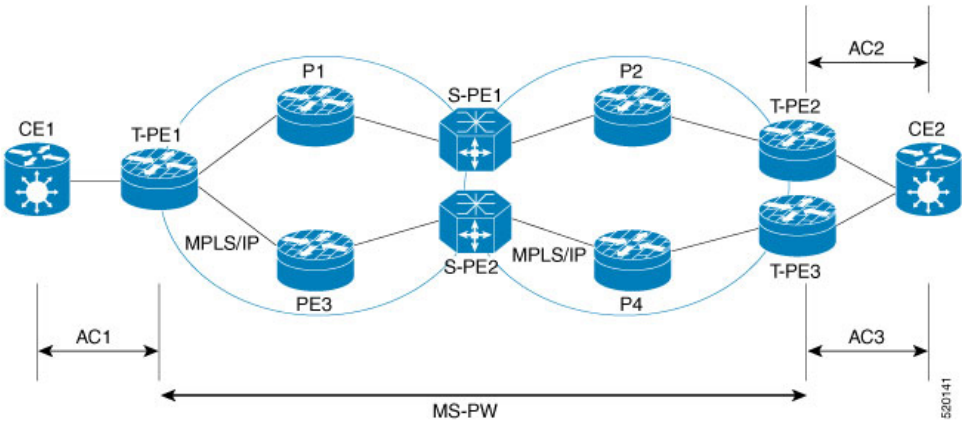
- Connect both segments of an MS-PW to different peers.
- Supports only LDP and does not support L2TPv3. Each PW segment in the MS-PW xconnect can be either static or dynamic.
- The neighbor pw-id pair of each PW segment of an MS-PW is unique on the node.
- The end-to-end pw-type has to be the same. Hence, both segments of an MS-PW must have the same transport mode.
- You cannot configure PW redundancy on an MS-PW xconnect at the S-PE. You can configure PW redundancy at the T-PEs.
- Both segments of an MS-PW xconnect can not have the same preferred path.
- Supports MS-PW over LDP, MPLS-TE, SR, and SR-TE as transport protocols.
- Does not support MS-PW over BGP-LU and LDPoTE.

- When you enable MSPW on an S-PE, configure the *ip-ttl-propagation disable* command for the MSPW ping and traceroute to work. Alternatively, use *segment-count 255 option* for MSPW ping to work from T-PE1. MSPW does not support the partial ping.

Multisegment Pseudowire Redundancy

Pseudowire redundancy enables you to create backup MS-PWs between the T-PEs. Pseudowire redundancy allows you to configure your network to detect a failure in the network. And reroute the Layer 2 service to another endpoint that can continue to provide service.

Figure 17: Multisegment Pseudowire Redundancy



Consider a topology where you create two MS-PWs and multihome CE2 to T-PE2 and T-PE3. Create a primary MS-PW between T-PE1 and T-PE2 connected through P1, S-PE1, and P2. Create a standby MS-PW between T-PE1 and T-PE3 connected through P3, S-PE2, and P4.

When a segment of the primary PW fails, the S-PE1 receives label withdraw message or LDP transport goes down. S-PE1 sends label withdraw message on the other PW segment and this triggers the switch-over to the backup at the T-PE. For example:

- T-PE1 detects LDP transport down, sends label withdraw message to S-PE1 and switches over to the backup MS-PW.
- S-PE1 receives the label withdraw message and sends a label withdraw message to T-PE2.
- T-PE2 performs “Tx Disable” of AC2 after it receives the label withdraw message.
- CE2 starts sending and receiving traffic on AC3.

Configure Multisegment Pseudowire

Peform this task to configure Multisegment Pseudowire.

```
/* Configure on T-PE1 */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#pw-class dynamic_mpls
Router(config-l2vpn-pwc)#encapsulation mpls
```

```
Router(config-l2vpn-pwc-encap-mpls)#protocol ldp

Router(config-l2vpn-pwc-encap-mpls)#exit
Router(config-l2vpn-pwc)#exit
Router(config-l2vpn)#xconnect group XCON1
Router(config-l2vpn-xc)#p2p xc1
Router(config-l2vpn-xc-p2p)#description T-PE1 MS-PW to 172.16.0.1 through 192.168.0.1
Router(config-l2vpn-xc-p2p)#interface gig0/1/0/0.1
Router(config-l2vpn-xc-p2p)#neighbor 192.168.0.1 pw-id 100
Router(config-l2vpn-xc-p2p-pw)#pw-class dynamic_mpls
Router(config-l2vpn-xc-p2p-pw)#commit

/* Configure on S-PE1 */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#xconnect group MS-PW1
Router(config-l2vpn-xc)#p2p ms-pw1
Router(config-l2vpn-xc-p2p)#description S-PE1 MS-PW between 10.0.0.1 and 172.16.0.1
Router(config-l2vpn-xc-p2p)#neighbor 10.0.0.1 pw-id 100
Router(config-l2vpn-xc-p2p-pw)#pw-class dynamic_mpls
Router(config-l2vpn-xc-p2p-pw)#exit
Router(config-l2vpn-xc-p2p)#neighbor 172.16.0.1 pw-id 300
Router(config-l2vpn-xc-p2p-pw)#pw-class dynamic_mpls
Router(config-l2vpn-xc-p2p-pw)#exit
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#pw-class dynamic_mpls
Router(config-l2vpn-pwc)#encapsulation mpls
Router(config-l2vpn-pwc-encap-mpls)#protocol ldp

Router(config-l2vpn-pwc-encap-mpls)#commit

/* Configure on T-PE2 */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#pw-class dynamic_mpls
Router(config-l2vpn-pwc)#encapsulation mpls
Router(config-l2vpn-pwc-encap-mpls)#protocol ldp

Router(config-l2vpn-pwc-encap-mpls)#exit
Router(config-l2vpn-pwc)#exit
Router(config-l2vpn)#xconnect group XCON1
Router(config-l2vpn-xc)#p2p xc1
Router(config-l2vpn-xc-p2p)#description T-PE2 MS-PW to 10.0.0.1 through 192.168.0.1
Router(config-l2vpn-xc-p2p)#interface gig0/2/0/0.4
Router(config-l2vpn-xc-p2p)#neighbor 192.168.0.1 pw-id 300
Router(config-l2vpn-xc-p2p-pw)#pw-class dynamic_mpls
Router(config-l2vpn-xc-p2p-pw)#commit
```

Running Configuration

This section shows multisegment pseudowire running configuration.

```
/* T-PE1 Configuration */
Configure
 l2vpn
  pw-class dynamic_mpls
   encapsulation mpls
   protocol ldp

 !
  xconnect group XCON1
```

```
   p2p xc1
    description T-PE1 MS-PW to 172.16.0.1 through 192.168.0.1
    interface gig0/1/0/0.1
    neighbor 192.168.0.1 pw-id 100
     pw-class dynamic_mpls
    !
  !

/* S-PE1 Configuration */
 l2vpn
  xconnect group MS-PW1
   p2p ms-pw1
    description S-PE1 MS-PW between 10.0.0.1 and 172.16.0.1
    neighbor 10.0.0.1 pw-id 100
     pw-class dynamic_mpls
     !
    neighbor 172.16.0.1 pw-id 300
     pw-class dynamic_mpls
    !
   !
l2vpn
 pw-class dynamic_mpls
  encapsulation mpls
  protocol ldp

  !
 !


/* T-PE2 Configuration */
Configure
 l2vpn
  pw-class dynamic_mpls
   encapsulation mpls
   protocol ldp

 !
  xconnect group XCON1
   p2p xc1
    description T-PE1 MS-PW to 10.0.0.1 through 192.168.0.1
    interface gig0/2/0/0.4
    neighbor 192.168.0.1 pw-id 300
     pw-class dynamic_mpls
    !
  !
```

Verification

Verify that you have configured Multisegment Pseudowire feature successfully.

```
Router:S-PE1#show l2vpn xocnnect
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
LU = Local Up, RU = Remote Up, CO = Connected

XConnect Group  Name     ST   Segment 1     ST    Segment 2     ST
                               Description         Description
--------------  -----    ---- ------------- ---  ------------  ---
MS-PW1          ms-pw1   UP   10.0.0.1      UP   172.16.0.1    UP
------------------------------------------------------------------


Router:S-PE1#show l2vpn xconnect detail
Group MS-PW1, XC ms-pw1, state is up; Interworking none
```

```
PW: neighbor 70.70.70.70, PW ID 100, state is up ( established )
  PW class not set
  Encapsulation MPLS, protocol LDP
  PW type Ethernet VLAN, control word enabled, interworking none
  PW backup disable delay 0 sec
  Sequencing not set
      MPLS         Local                          Remote
    ------------ ------------------------------ ------------------------------
    Label        16004                          16006
    Group ID     0x2000400                      0x2000700
    Interface    GigabitEthernet0/1/0/2.2       GigabitEthernet0/1/0/0.3
    MTU          1500                           1500
    Control word enabled                        enabled
    PW type      Ethernet VLAN                  Ethernet VLAN
    VCCV CV type 0x2                            0x2
                 (LSP ping verification)        (LSP ping verification)
    VCCV CC type 0x5                            0x7
                 (control word)                 (control word)
                                                (router alert label)
                 (TTL expiry)                   (TTL expiry)
    ------------ ------------------------------ ------------------------------
  Incoming PW Switching TLV:
    IP Address: 70.70.70.70, PW ID: 100
    Description: T-PE1 MS-PW to 172.16.0.1via 192.168.0.1
  Outgoing PW Switching TLV:
    IP Address: 90.90.90.70, PW ID: 300
    Description: T-PE2 MS-PW to 10.0.0.1via 192.168.0.1
    IP Address: 192.168.0.1, PW ID: 100
    Description: S-PE1 MS-PW between 10.0.0.1and 90.90.90.90
  Create time: 04/04/2008 23:18:24 (00:01:24 ago)
  Last time status changed: 04/04/2008 23:19:30 (00:00:18 ago)
  Statistics:
    packet totals: receive 0
    byte totals: receive 0
PW: neighbor 90.90.90.90, PW ID 300, state is up ( established )
  PW class not set
  Encapsulation MPLS, protocol LDP
  PW type Ethernet VLAN, control word enabled, interworking none
  PW backup disable delay 0 sec
  Sequencing not set
      MPLS         Local                          Remote
    ------------ ------------------------------ ------------------------------
    Label        16004                          16006
    Group ID     0x2000800                      0x2000200
    Interface    GigabitEthernet0/1/0/0.3       GigabitEthernet0/1/0/2.2
    MTU          1500                           1500
    Control word enabled                        enabled
    PW type      Ethernet VLAN                  Ethernet VLAN
    VCCV CV type 0x2                            0x2
                 (LSP ping verification)        (LSP ping verification)
    VCCV CC type 0x5                            0x7
                 (control word)                 (control word)
                                                (router alert label)
                 (TTL expiry)                   (TTL expiry)
    ------------ ------------------------------ ------------------------------
  Incoming PW Switching TLV:
    IP Address: 90.90.90.90, PW ID: 300
    Description: T-PE2 MS-PW to 10.0.0.1via 192.168.0.1
  Outgoing PW Switching TLV:
    IP Address: 70.70.70.70, PW ID: 100
    Description: T-PE1 MS-PW to 172.16.0.1via 192.168.0.1
    IP Address: 192.168.0.1, PW ID: 300
    Description: S-PE1 MS-PW between 10.0.0.1and 90.90.90.90
  Create time: 04/04/2008 23:18:24 (00:01:24 ago)
```

```
Last time status changed: 04/04/2008 23:19:30 (00:00:18 ago)
Statistics:
  packet totals: receive 0
  byte totals: receive 0
```

Related Topics

- Multisegment Pseudowire, on page 144
- Multisegment Pseudowire Redundancy, on page 146

Associated Commands

- show l2vpn xconnect
- show l2vpn xconnect detail
- show l2vpn xconnect summary

Split Horizon Groups

Cisco IOS XR bridge domain aggregates attachment circuits (ACs) in one of three groups called Split Horizon Groups. When applied to bridge domains, Split Horizon refers to the flooding and forwarding behavior between members of a Split Horizon group. The following table describes how frames received on one member of a split horizon group are treated and if the traffic is forwarded out to the other members of the same split horizon group.

Bridge Domain traffic is either unicast or multicast.

Flooding traffic consists of the following unknown unicast destination MAC address frames.

- The frames are sent to Ethernet multicast addresses (Spanning Tree BPDUs)
- Ethernet broadcast frames (MAC address FF-FF-FF-FF-FF-FF).

The known unicast traffic consists of frames sent to bridge ports that were learned from that port using MAC learning.

Traffic flooding is performed for broadcast, multicast and unknown unicast destination address.

Table 15: Split Horizon Groups Supported on Cisco IOS-XR

Split Horizon Group	Who belongs to this Group?	Multicast within Group	Unicast within Group
0	Default—any member not covered by groups 1 or 2.	Yes	Yes
1	Any PW configured under VFI.	No	No
2	Any AC configured with split-horizon keyword.	No	No

Important notes on Split Horizon Groups:

- All bridge ports or PWs that are members of a bridge domain must belong to one of the three groups.

- By default, all bridge ports or PWs are members of group 0.
- The VFI configuration submode under a bridge domain configuration indicates that members under this domain are included in group 1.
- A PW that is configured in group 0 is called an Access Pseudowire.
- The **split-horizon group** command is used to designate bridge ports as members of group 2.
- Known unicast is also filtered within the members of the group along with the Broadcast, Unknown unicast and Multicast (BUM) traffic.

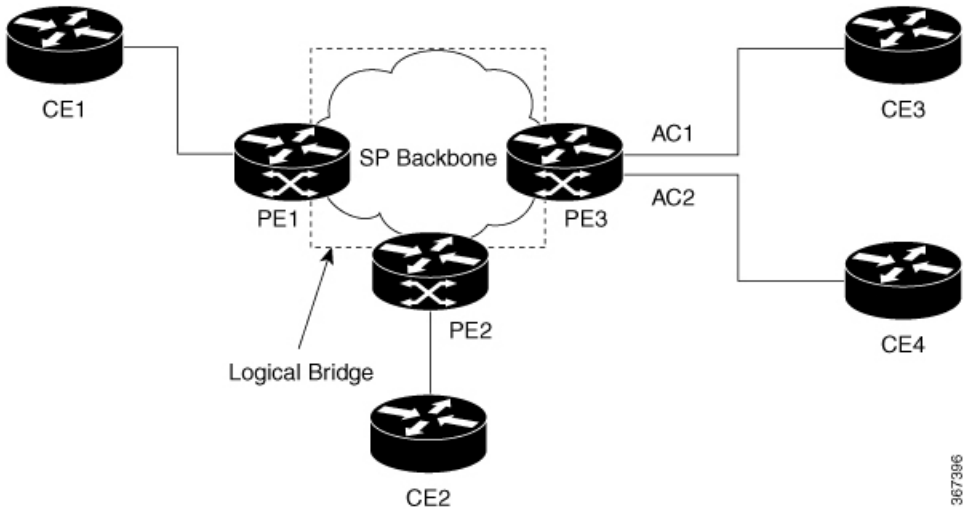
Split Horizon Group 2

The Split Horizon Group 2 feature allows you to prevent BUM and known unicast traffic to be flooded from one AC to other AC within the bridge domain. This feature enables efficient bandwidth allocation and resource optimization.

Consider the following topology in which AC1 and AC2 are part of the same VPLS bridge domain. When you configure split horizon group 2 over AC1, AC2 on PE3, BUM and known unicast traffic from AC1 is not flooded to AC2 and vice-versa.

However, BUM traffic coming from the pseduowire on PE3 to AC1 and AC2 that are part of group 2 is flooded. The known unicast traffic is sent to the corresponding AC.

Figure 18: Split Horizon Group 2



If AC1 is part of group 0 and AC2 is part of group 2, BUM and known unicast traffic is flooded between AC1 and AC2. Similarly, if AC2 is part of group 0 and AC1 is part of group 2, BUM and known unicast traffic is flooded between AC1 and AC2.

Configure Split Horizon Group 2

Perform this task to configure the Split Horizon Group 2 feature.

Configuration Example

This example shows how to configure interfaces for Layer 2 transport, add them to a bridge domain, and assign them to split horizon group 2.

```
/* Configure on PE3 */
Router#configure
Router(config)l2vpn
Router(config-l2vpn)#router-id 192.168.0.1
Router(config-l2vpn)#pw-class class1
Router(config-l2vpn-pwc)#encapsulation mpls
Router(config-l2vpn-pwc-encapmpls)#protocol ldp
Router(config-l2vpn-pwc-encapmpls)#ipv4 source 192.168.0.1
Router(config-l2vpn-pwc-encapmpls)#exit
Router(config-l2vpn-pwc)#exit
Router(config-l2vpn)#bridge goup bg1
Router(config-l2vpn-bg)#bridge-domain bd
Router(config-l2vpn-bg-bd)#exit
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface TenGigE
Router(config-l2vpn-bg-bd-ac)#split-horizon group
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#interface TenGigE

Router(config-l2vpn-bg-bd-ac)#split-horizon group
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#vfi vfi1
Router(config-l2vpn-bg-bd-vfi)#neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)#pw-class class1
Router(config-l2vpn-bg-bd-vfi-pw)#commit
```

Running Configuration

```
configure
 l2vpn
  router-id 192.168.0.1
  pw-class classs1
   encapsulation mpls
    protocol ldp
    ipv4 source 192.168.0.1
    !
   !
   bridge group bg1
    bridge-domain bd
    !
    bridge-domain bd1
     interface TenGigE
      split-horizon group
     !
     interface TenGigE
      split-horizon group
     !
     vfi vfi1
      neighbor 10.0.0.1 pw-id 1
       pw-class class1
      !
     !
    !
```

Verification

Verify whether the traffic is egressing out of the respective group 2 AC.

```
Router#show l2vpn bridge-domain bd-name bd1
Thu Jun 14 08:04:47.431 IST

  Legend: pp = Partially Programmed.
Bridge group: bg1, bridge-domain: bd1, id: 1, state: up, ShgId: 0, MSTi: 0
  Aging: 300s, MAC limit: 64000, Action: none, Notification: syslong
  Filter MAC addresses: 0
  ACs: 2 (2 up), VFIs: 1, PWs: 1 (up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    Te
, stage: up, Static MAC addresses: 0
    Te, stage: up, Static MAC addresses: 0
  List of Access PWs:
  List of VFIs:
    VFI vfi1 (up)
     Neighbor 10.0.0.1 pw-id 1, stage: up, Static MAC Addresses: 0
```

G.8032 Ethernet Ring Protection

The G.8032 Ethernet Ring Protection feature provides protection for Ethernet traffic in a ring topology. This feature prevents loops within the ring at the Ethernet layer by blocking either a pre-determined link or a failed link. You can configure this feature on physical and bundle interfaces.

Note You can configure HQoS on an AC interface that is part of the G.8032 ring. However, this functionality has a limitation on the G.8032 convergence. The convergence depends on the number of AC interfaces used in a G.8032 ring. This limitation is applicable when the HQOS mode is enabled at the system level or at the G.8032 AC level.

Overview

Each Ethernet ring node is connected to adjacent Ethernet ring nodes participating in the Ethernet ring using two independent links. A ring link never allows formation of loops that affect the network. The Ethernet ring uses a specific link to protect the entire Ethernet ring. This specific link is called the ring protection link (RPL). A ring link is bound by two adjacent Ethernet ring nodes and a port for a ring link (also known as a ring port).

Note The minimum number of Ethernet ring nodes in an Ethernet ring is two.

The fundamentals of ring protection switching are:

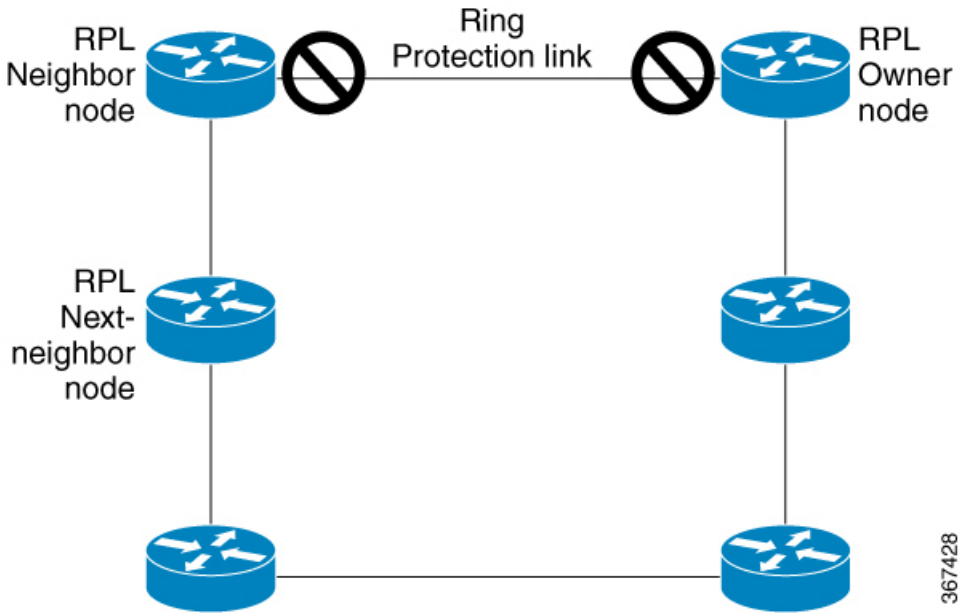
- The principle of loop avoidance.
- The utilization of learning, forwarding, and Filtering Database (FDB) mechanisms.

Loop avoidance in an Ethernet ring is achieved by ensuring that, at any time, traffic flows on all but one of the ring links which is the RPL. Multiple nodes are used to form a ring:

- RPL owner—It is responsible for blocking traffic over the RPL so that no loops are formed in the Ethernet traffic. There can be only one RPL owner in a ring.
- RPL neighbor node—The RPL neighbor node is an Ethernet ring node adjacent to the RPL. It is responsible for blocking its end of the RPL under normal conditions. This node type is optional and prevents RPL usage when protected.
- RPL next-neighbor node—The RPL next-neighbor node is an Ethernet ring node adjacent to RPL owner node or RPL neighbor node. It is mainly used for FDB flush optimization on the ring. This node is also optional.

The following figure illustrates the G.8032 Ethernet ring.

Figure 19: G.8032 Ethernet Ring



Nodes on the ring use control messages called RAPS to coordinate the activities of switching on or off the RPL link. Any failure along the ring triggers a RAPS signal fail (RAPS SF) message along both directions, from the nodes adjacent to the failed link, after the nodes have blocked the port facing the failed link. On obtaining this message, the RPL owner unblocks the RPL port.

Note A single link failure in the ring ensures a loop-free topology.

Line status and Connectivity Fault Management protocols are used to detect ring link and node failure. During the recovery phase, when the failed link is restored, the nodes adjacent to the restored link send RAPS no request (RAPS NR) messages. On obtaining this message, the RPL owner blocks the RPL port and sends RAPS no request, root blocked (RAPS NR, RB) messages. This causes all other nodes, other than the RPL owner in the ring, to unblock all blocked ports. The ERP protocol is robust enough to work for both unidirectional failure and multiple link failure scenarios in a ring topology.

A G.8032 ring supports these basic operator administrative commands:

- Force switch (FS)—Allows operator to forcefully block a particular ring-port.

- Effective even if there is an existing SF condition
- Multiple FS commands for ring supported
- May be used to allow immediate maintenance operations

- Manual switch (MS)—Allows operator to manually block a particular ring-port.
  - Ineffective in an existing FS or SF condition
  - Overridden by new FS or SF conditions
  - Clears all previous MS commands

- Clear—Cancels an existing FS or MS command on the ring-port
  - Used (at RPL Owner) to clear non-revertive mode

A G.8032 ring can support two instances. An instance is a logical ring running over a physical ring. Such instances are used for various reasons, such as load balancing VLANs over a ring. For example, odd VLANs may go in one direction of the ring, and even VLANs may go in the other direction. Specific VLANs can be configured under only one instance. They cannot overlap multiple instances. Otherwise, data traffic or RAPS packet can cross logical rings, and that is not desirable.

Timers

G.8032 ERP specifies the use of different timers to avoid race conditions and unnecessary switching operations:

- Delay Timers—used by the RPL Owner to verify that the network has stabilized before blocking the RPL
  - After SF condition, Wait-to-Restore (WTR) timer is used to verify that SF is not intermittent. The WTR timer can be configured by the operator, and the default time interval is 5 minutes. The time interval ranges from 1 to 12 minutes.
  - After FS/MS command, Wait-to-Block timer is used to verify that no background condition exists.

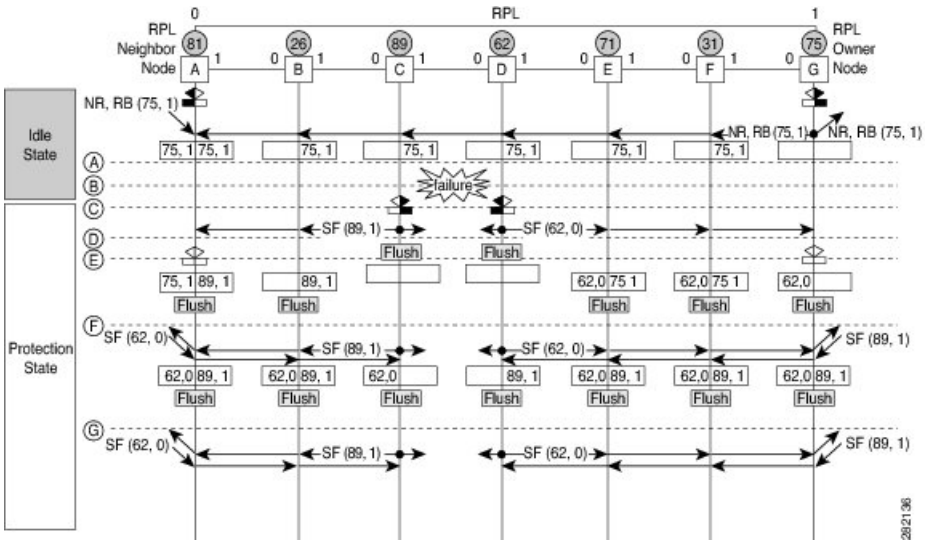
Note Wait-to-Block timer may be shorter than the Wait-to-Restore timer

- Guard Timer—used by all nodes when changing state; it blocks latent outdated messages from causing unnecessary state changes. The Guard timer can be configured and the default time interval is 500 ms. The time interval ranges from 10 to 2000 ms.
- Hold-off timers—used by underlying Ethernet layer to filter out intermittent link faults. The hold-off timer can be configured and the default time interval is 0 seconds. The time interval ranges from 0 to 10 seconds.
  - Faults are reported to the ring protection mechanism, only if this timer expires.

Single Link Failure

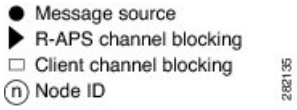
The following figure represents protection switching in case of a single link failure.

Figure 20: G.8032 Single Link Failure



The above figure represents an Ethernet ring composed of seven Ethernet ring nodes. The RPL is the ring link between Ethernet ring nodes A and G. In these scenarios, both ends of the RPL are blocked. Ethernet ring node G is the RPL owner node, and Ethernet ring node A is the RPL neighbor node.

These symbols are used:

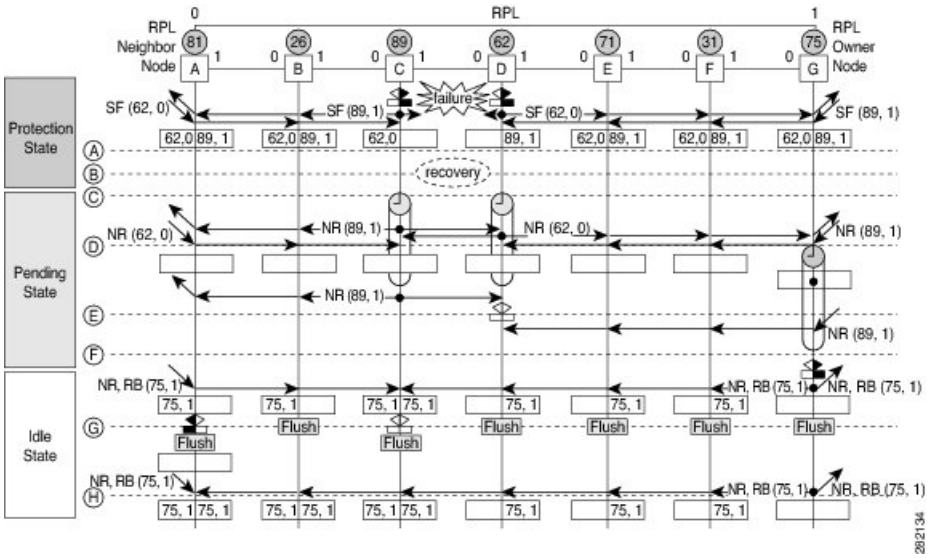


This sequence describes the steps in the single link failure:

1. Link operates in the normal condition.
2. A failure occurs.
3. Ethernet ring nodes C and D detect a local Signal Failure condition and after the holdoff time interval, block the failed ring port and perform the FDB flush.
4. Ethernet ring nodes C and D start sending RAPS (SF) messages periodically along with the (Node ID, BPR) pair on both ring ports, while the SF condition persists.
5. All Ethernet ring nodes receiving an RAPS (SF) message perform FDB flush. When the RPL owner node G and RPL neighbor node A receive an RAPS (SF) message, the Ethernet ring node unblocks it's end of the RPL and performs the FDB flush.
6. All Ethernet ring nodes receiving a second RAPS (SF) message perform the FDB flush again; this is because of the Node ID and BPR-based mechanism.
7. Stable SF condition—RAPS (SF) messages on the Ethernet Ring. Further RAPS (SF) messages trigger no further action.

The following figure represents reversion in case of a single link failure.

Figure 21: Single link failure Recovery (Revertive operation)



This sequence describes the steps in the single link failure recovery:

1. Link operates in the stable SF condition.
2. Recovery of link failure occurs.
3. Ethernet ring nodes C and D detect clearing of signal failure (SF) condition, start the guard timer and initiate periodical transmission of RAPS (NR) messages on both ring ports. (The guard timer prevents the reception of RAPS messages).
4. When the Ethernet ring nodes receive an RAPS (NR) message, the Node ID and BPR pair of a receiving ring port is deleted and the RPL owner node starts the WTR timer.
5. When the guard timer expires on Ethernet ring nodes C and D, they may accept the new RAPS messages that they receive. Ethernet ring node D receives an RAPS (NR) message with higher Node ID from Ethernet ring node C, and unblocks its non-failed ring port.
6. When WTR timer expires, the RPL owner node blocks its end of the RPL, sends RAPS (NR, RB) message with the (Node ID, BPR) pair, and performs the FDB flush.
7. When Ethernet ring node C receives an RAPS (NR, RB) message, it removes the block on its blocked ring ports, and stops sending RAPS (NR) messages. On the other hand, when the RPL neighbor node A receives an RAPS (NR, RB) message, it blocks its end of the RPL. In addition to this, Ethernet ring nodes A to F perform the FDB flush when receiving an RAPS (NR, RB) message, due to the existence of the Node ID and BPR based mechanism.

Configure G.8032 Ethernet Ring Protection

The ERP feature supports both revertive and non-revertive mode of operation. By default, ERP rings operate in revertive mode unless explicitly configured as non-revertive mode under ERP profile configuration.

Perform the following tasks to configure the Ethernet Ring Protection feature:

- Confiigure ERP Profile

- Configure an ERP Instance

Note Tag re-write, either push or pop on sub-interface being used as Ring Automatic Protection Switching (RAPS) channel is not supported.

Configure ERP Profile

Perform this task to configure Ethernet ring protection (ERP) profile.

Configuration Examle

```
Router#configure
Router(config)ethernet ring g8032 profile p1
Router(config-g8032-ring-profile)#timer wtr 5
Router(config-g8032-ring-profile)#non-revertive
Router(config-g8032-ring-profile)#commit
```

Revertive Mode—In this mode, RPL is blocked after a failed ERP link comes up and WTR timer has expired. There is no specific command or configuration to enable this mode. By default, ERP rings operate in revertive mode unless explicitly configured as non-revertive mode under ERP profile configuration.

Non-revertive Mode —In this mode, RPL remains in the blocked state and the recovered link also remains in a blocked state until you run **erp clear** command on the RPL owner node, or there is a new SF in the ring.

Running Confiugration

```
configure
 Ethernet ring g8032 profile p1
  timer wtr 5
  non-revertive
  !
 !
```

Configuring an ERP Instance

Perform this task to configure an ERP instance.

Configuration Examle

```
Router#configure
Router(config)l2vpn
Router(config-l2vpn)#ethernet ring g8032 ring1
Router(config-l2vpn-erp)#port0 interface TenGigE0/0/0/0
/* To configure an ERP on bundle interface, use the following command */
Router(config-l2vpn-erp)#port0 interface bundle-ether1
Router(config-l2vpn-erp-port0)#exit
Router(config-l2vpn-erp)#port1 interface TenGigE0/0/0/8
/* To configure an ERP on bundle interface, use the following command */
Router(config-l2vpn-erp)#port1 interface bundle-ether2
Router(config-l2vpn-erp-port1)#exit
Router(config-l2vpn-erp)#instance 1
Router(config-l2vpn-erp-instance)#profile p1
Router(config-l2vpn-erp-instance)#rpl port0 owner
```

```
Router(config-l2vpn-erp-instance)#inclusion-list vlan-ids 1,7-150
Router(config-l2vpn-erp-instance)#aps-channel
Router(config-l2vpn-erp-instance-aps)#port0 interface TenGigE
Router(config-l2vpn-erp-instance-aps)#port1 interface TenGigE
/* To configure an ERP instance on bundle sub-interfaces, use the following command */
Router(config-l2vpn-erp-instance-aps)#port0 interface bundle-ether1.1
Router(config-l2vpn-erp-instance-aps)#port1 interface bundle-ether2.1
Router(config-l2vpn-erp-instance-aps)#commit
```

Inclusion list vlan ids—ports of these vlans are protected and traffic is switched only for these ports.

Exclusion list vlan ids—these vlan ids are not protected by G.8032, traffic for these vlans is forwarded normally, ports of these vlans are not blocked by G.8032.

Vlans not part of either list—are part of default instance and traffic is dropped for these vlans.

Running Confiugration

```
configure
 l2vpn
  ethernet ring g8032 ring1
   port0 interface TenGigE0/0/0/0
  !
   port1 interface TenGigE0/0/0/8
  !
  instance 1
   profile fretta
   rpl port0 owner
   inclusion-list vlan-ids 1,7-150
   aps-channel
    port0 interface TenGigE
    port1 interface TenGigE
   !
  !

 !
```

Verification

Verify the status of Ethernet ring.

```
Router#show ethernet ring g8032 ring1
Thu Jun 14 08:04:47.431 IST

  R: Interface is the RPL-link
  F: Interface is faulty
  B: Interface is blocked
  N: Interface is not present
 FS: Local forced switch
 MS: Local manual switch

RingName                          Inst  NodeType  NodeState    Port0    Port1
-------------------------------------------------------------------------------
ring1                               1    Owner     Idle         R,B
-------------------------------------------------------------------------------------------

Router#show ethernet ring g8032 status
Thu Jun 14 08:05:35.263 IST

Ethernet ring ring1 instance 1 is RPL Owner node in Idle state
  Port0: TenGigE0/0/0/0 (Monitor: TenGigE0/0/0/0)
```

```
        APS-Channel: TenGigE0/0/0/0.1
        Status: RPL, blocked
        Remote R-APS NodeId: 0000.0000.0000, BPR: 0
  Port1: TenGigE0/0/0/8 (Monitor: TenGigE0/0/0/8)
        APS-Channel: TenGigE0/0/0/8.1
        Status: NonRPL
        Remote R-APS NodeId: 0000.0000.0000, BPR: 0
  APS Level: 7
  Open APS ring topology
  Profile: p1
    WTR interval: 1 minutes
    Guard interval: 500 milliseconds
    Hold-off interval: 0 seconds
    Revertive mode
```

Configuring G.8032 Ethernet Ring Protection: Example

This sample configuration illustrates the elements that a complete G.8032 configuration includes:

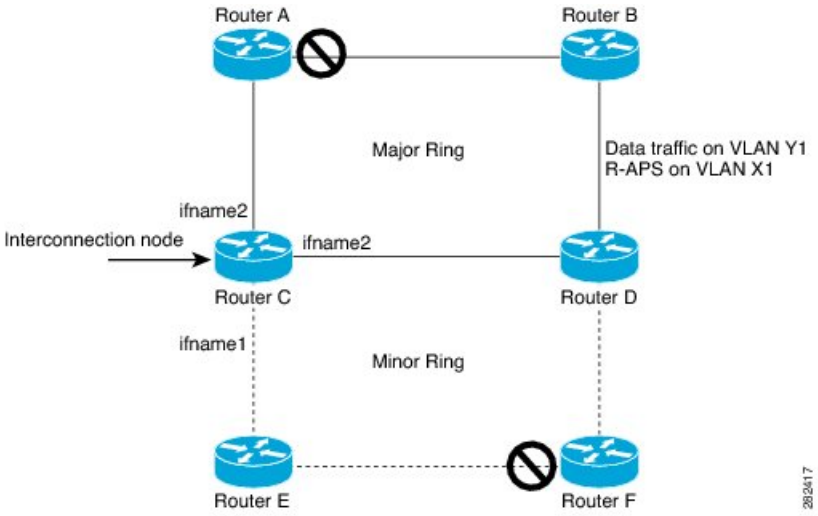
```
# Configure the ERP profile characteristics if ERP instance behaviors are non-default.
ethernet ring g8032 profile ERP-profile
  timer wtr 10
  timer guard 100
  timer hold-off 1
  non-revertive

# Configure CFM MEPs and configure to monitor the ring links.
ethernet cfm
  domain domain1
    service link1 down-meps
   continuity-check interval 100ms
      efd
   mep crosscheck
  mep-id 2
  domain domain2
    service link2 down-meps
   continuity-check interval 100ms
      efd protection-switching
   mep crosscheck
  mep id 2

Interface Gig 0/0/0/0
  ethernet cfm mep domain domain1 service link1 mep-id 1
Interface Gig
  ethernet cfm mep domain domain2 service link2 mep-id 1

# Configure the ERP instance under L2VPN
l2vpn
  ethernet ring g8032 RingA
    port0 interface g0/0/0/0
    port1 interface g
    instance 1
      description BD2-ring
      profile ERP-profile
      rpl port0 owner
      inclusion-list vlan-ids 10-100
      aps channel
        level 3
        port0 interface g0/0/0/0.1
        port1 interface g

# Set up the bridge domains
```

```
bridge group ABC
    bridge-domain BD2
      interface Gig

      interface Gig
      interface Gig

    bridge-domain BD2-APS
      interface Gig
      interface Gig

# EFPs configuration
interface Gig  l2transport
  encapsulation dot1q 5

interface Gig  l2transport
  encapsulation dot1q 5

interface g l2transport
  encapsulation dot1q 10-100

interface g  l2transport
  encapsulation dot1q 10-100

interface g  l2transport
  encapsulation dot1q 10-100
```

Configuring Interconnection Node: Example

This example shows you how to configure an interconnection node. The following figure illustrates an open ring scenario.

Figure 22: Open Ring Scenario - interconnection node



The minimum configuration required for configuring G.8032 at Router C (Open ring – Router C):

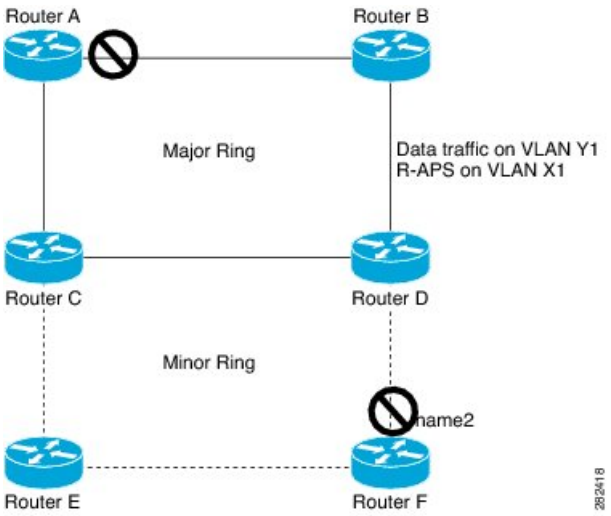
```
interface Gig 0/0/0/1.1 l2transport
 encapsulation dot1q 5
interface Gig 0/0/0/1.10 l2transport
 encapsulation dot1q 6
interface Gig 0/0/0/2.10 l2transport
 encapsulation dot1q 6
```

```
interface Gig 0/0/0/3.10 l2transport
 encapsulation dot1q 6
l2vpn
ethernet ring g8032 ring8
      port0 interface Gig 0/0/0/1
      port1 none  /* This router is connected to an interconnection node. */
      open-ring
     !
     instance 1
      inclusion-list vlan-ids 1,7-150
      aps-channel
       port0 interface Gig 0/0/0/1.1
       port1 none /* This router is connected to an interconnection node */
      !
 bridge group bg1
  bridge-domain BD2 /* Data traffic has its own bridge domain */
   interface Gig 0/0/0/1.10
   interface Gig 0/0/0/2.10
   interface Gig 0/0/0/3.10
  !
  bridge-domain BD2-APS /* APS-channel has its own bridge domain */
   interface Gig 0/0/0/1.1 /* There is only one APS-channel at the interconnection node */
```

Configuring the Node of an Open Ring: Example

This example shows you how to configure the node part of an open ring. The following figure illustrates an open ring scenario.

Figure 23: Open Ring Scenario



The minimum configuration required for configuring G.8032 at the node of the open ring (node part of the open ring at router F):

```
interface Gig 0/0/0/1.1 l2transport
 encapsulation dot1q 5
interface Gig 0/0/0/2.1 l2transport
 encapsulation dot1q 5
interface Gig 0/0/0/1.10 l2transport
 encapsulation dot1q 6
interface Gig 0/0/0/2.10 l2transport
```

```
 encapsulation dot1q 6
l2vpn
   ethernet ring g8032 ringB
      port0 interface Gig 0/0/0/1
      port1 interface Gig 0/0/0/2
      open-ring
     !
     instance 1
      inclusion-list vlan-ids 1,7-150
      rpl port0 owner  /* This node is RPL owner and interface Gig 0/0/0/2 is blocked
      aps-channel
       port0 interface Gig 0/0/0/1.1
       port1 interface Gig 0/0/0/2.1

/* Set up the bridge domain
 bridge group bg1
   bridge-domain BD2
    bridge-domain BD2-APS /* APS-channel has its own bridge domain */
     interface Gig 0/0/0/1.1
     interface Gig 0/0/0/2.1
   !
/* Data traffic has its own bridge domain */
   bridge-domain BD2
    interface Gig 0/0/0/1.10
    interface Gig 0/0/0/2.10
```

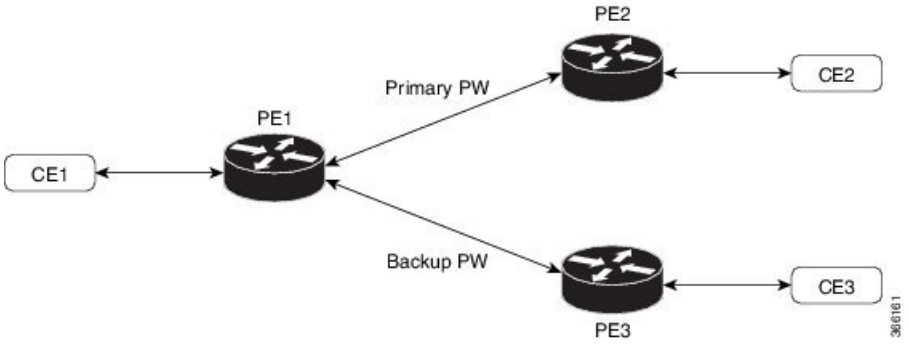
Pseudowire Redundancy

Table 16: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire Redundancy	Release 7.4.1	This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.

The Pseudowire Redundancy feature allows you to configure a redundant pseudowire that backs up the primary pseudowire. When the primary pseudowire fails, the PE router switches to the redundant pseudowire. You can elect to have the primary pseudowire resume operation after it becomes functional. The primary pseudowire fails when the PE router fails or when there is a network outage.

Figure 24: Pseudowire Redundancy



Forcing a Manual Switchover to the Backup Pseudowire

To force the router to switch over to the backup or switch back to the primary pseudowire, use the **l2vpn switchover** command in EXEC mode.

A manual switchover is made only if the peer specified in the command is actually available and the cross-connect moves to the fully active state when the command is entered.

Configure Pseudowire Redundancy

This section describes how you can configure pseudowire redundancy.

You must consider the following restrictions while configuring the Pseudowire Redundancy feature:

- 2000 active and 2000 backup PWs are supported.
- Only MPLS LDP is supported.

```
/* Configure PW on PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet
Router(config-l2vpn-xc-p2p)# neighbor ipv4 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# backup neighbor 192.168.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw-backup)# commit

/* Configure PW on PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit

/* Configure PW on PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* On PE1 */
!
l2vpn
 xconnect group XCON1
  p2p XCON1_P2P2
   interface GigabitEthernet
   neighbor ipv4 172.16.0.1 pw-id 1
    backup neighbor 192.168.0.1 pw-id 1
!

/* On PE2 */
!
l2vpn
```

```
 xconnect group XCON1
  p2p XCON1_P2P2
   interface GigabitEthernet
   neighbor ipv4 10.0.0.1 pw-id 1

!

/* On PE3 */
!
l2vpn
 xconnect group XCON1
  p2p XCON1_P2P2
   interface GigabitEthernet
   neighbor ipv4 10.0.0.1 pw-id 1

!
```

Verification

Verify that the configured pseudowire redundancy is up.

```
/* On PE1 */

Router#show l2vpn xconnect group XCON_1
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                   Segment 1                      Segment 2
Group      Name       ST   Description            ST       Description            ST
------------------------   -----------------------------   -----------------------------
XCON_1     XCON1_P2P2 UP   Gi0/1/0/0.1            UP       172.16.0.1     1000   UP
                                                           Backup
                                                           192.168.0.1     1000   SB
-------------------------------------------------------------------------------------

/* On PE2 */

Router#show l2vpn xconnect group XCON_1
Tue Jan 17 15:36:12.327 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                   Segment 1                      Segment 2
Group      Name       ST   Description            ST       Description            ST
------------------------   -----------------------------   -----------------------------
XCON_1     XCON1_P2P2 UP   BE100.1                UP       10.0.0.1     1000   UP
-------------------------------------------------------------------------------------

/* On PE3 */

Router#show l2vpn xconnect group XCON_1
Tue Jan 17 15:38:04.785 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                   Segment 1                      Segment 2
Group      Name       ST   Description            ST       Description            ST
------------------------   -----------------------------   -----------------------------
XCON_1     XCON1_P2P2 DN   BE100.1                UP       10.0.0.1     1000   SB
-------------------------------------------------------------------------------------

Router#show l2vpn xconnect summary
```

```
Number of groups: 3950
Number of xconnects: 3950
  Up: 3950  Down: 0  Unresolved: 0 Partially-programmed: 0
  AC-PW: 3950  AC-AC: 0  PW-PW: 0 Monitor-Session-PW: 0
Number of Admin Down segments: 0
Number of MP2MP xconnects: 0
  Up 0 Down 0
  Advertised: 0 Non-Advertised: 0
Number of CE Connections: 0
  Advertised: 0 Non-Advertised: 0
Backup PW:
  Configured   : 3950
  UP           : 0
  Down         : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby      : 3950
  Standby Ready: 0
Backup Interface:
  Configured   : 0
  UP           : 0
  Down         : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby      : 0
```

Configure Pseudowire Redundancy

Pseudowire redundancy allows you to configure your network to detect a failure in the network and reroute the Layer 2 service to another endpoint that can continue to provide service. This feature provides the ability to recover from a failure of either the remote provider edge (PE) router or the link between the PE and customer edge (CE) routers.

L2VPNs can provide pseudowire resiliency through their routing protocols. When connectivity between end-to-end PE routers fails, an alternative path to the directed LDP session and the user data takes over. However, there are some parts of the network in which this rerouting mechanism does not protect against interruptions in service.

Pseudowire redundancy enables you to set up backup pseudowires. You can configure the network with redundant pseudowires and redundant network elements.

Prior to the failure of the primary pseudowire, the ability to switch traffic to the backup pseudowire is used to handle a planned pseudowire outage, such as router maintenance.

Note Pseudowire redundancy is provided only for point-to-point Virtual Private Wire Service (VPWS) pseudowires.

Configuration

This section describes the configuration for pseudowire redundancy.

```
/* Configure a cross-connect group with a static point-to-point
cross connect */
Router# configure
Router(config)# l2vpn
```

```
Router(config-l2vpn)# xconnect group A
Router(config-l2vpn-xc)# p2p xc1
ROuter(config-l2vpn-xc-p2p)# interface tengige 0/0/0/0.2
Router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 2

/*Configure the pseudowire segment for the cross-connect group */
Router(config-l2vpn-xc-p2p-pw)#pw-class path1

/*Configure the backup pseudowire segment for the cross-connect group */
Router(config-l2vpn-xc-p2p-pw)# backup neighbor 10.2.2.2 pw-id 5
Router(config-l2vpn-xc-p2p-pw-backup)#end

/*Commit your configuration */
Router(config-l2vpn-xc-p2p-pw-backup)#commit
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]: yes
```

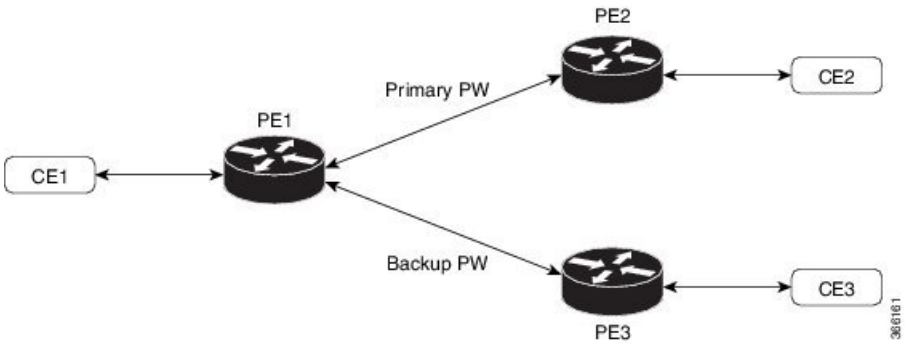
Running Configuration

```
Router# show-running configuration
...
l2vpn
 encapsulation mpls
 !
 xconnect group A
  p2p xc1
   interface tengige 0/0/0/0.2
   neighbor ipv4 10.1.1.2 pw-id 2
    pw-class path1
    backup neighbor 10.2.2.2 pw-id 5
    !
   !
...
```

Access Pseudowire Redundancy

The Access Pseudowire Redundancy feature allows you to configure a backup pseudowire under the bridge domain. When the primary pseudowire fails, the PE router switches to the backup pseudowire. The primary pseudowire resumes operation after it becomes functional. The primary pseudowire fails when the PE router fails or when there is a network outage.

Figure 25: Access Pseudowire Redundancy



Configure Access Pseudowire Redundancy

This section describes how you can configure access pseudowire redundancy.

Configuration Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group r1
Router(config-l2vpn-bg)# bridge-domain r1
Router(config-l2vpn-bg-bd)# interface TenGigE0/1/0/0.4
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# neighbor 10.0.0.1 pw-id 4
Router(config-l2vpn-bg-bd-pw)# backup neighbor 172.16.0.1 pw-id 4
Router(config-l2vpn-bg-bd-pw-backup)# commit
Router(config-l2vpn-bg-bd-pw-backup)# exit


Router# configure
Router(config)# interface TenGigE0/1/0/0.4 l2transport
Router(config-subif)# encapsulation dot1q 4
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# commit
```

Running Configuration

This section shows access pseudowire redundancy running configuration.

```
configure
 l2vpn
  bridge group r1
   bridge-domain r1
    interface TenGigE0/1/0/0.4
   !
   neighbor 10.0.0.1 pw-id 4
    backup neighbor 172.16.0.1 pw-id 4
    !
   !
  !
 !
!
interface TenGigE0/1/0/0.4 l2transport
 encapsulation dot1q 4
 rewrite ingress tag pop 1 symmetric
```

Verification

Verify the access pseudowire redundancy configuration.

```
Router# show l2vpn bridge-domain bd-name r1

Thu Apr 30 03:52:13.096 UTC
Legend: pp = Partially Programmed.
Bridge group: r1, bridge-domain: r1, id: 1, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 32000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 1 (1 up), VFIs: 0, PWs: 2 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    Te0/1/0/0.4, state: up, Static MAC addresses: 0
```

```
List of Access PWs:
    Neighbor 10.0.0.1 pw-id 4, state: up, Static MAC addresses: 0
    Neighbor 172.16.0.1 pw-id 4, state: standby, Static MAC addresses: 0, backup
List of VFIs:
List of Access VFIs:
```

Related Topics

- Access Pseudowire Redundancy, on page 167

Associated Commands

- show l2vpn bridge-domain

Virtual Circuit Connection Verification on L2VPN

Virtual Circuit Connection Verification (VCCV) is an L2VPN Operations, Administration, and Maintenance (OAM) feature that allows network operators to run IP-based provider edge-to-provider edge (PE-to-PE) keepalive protocol across a specified pseudowire to ensure that the pseudowire data path forwarding does not contain any faults. The disposition PE receives VCCV packets on a control channel, which is associated with the specified pseudowire. The control channel type and connectivity verification type, which are used for VCCV, are negotiated when the pseudowire is established between the PEs for each direction.

Two types of packets can arrive at the disposition egress:

- Type 1—Specifies normal Ethernet-over-MPLS (EoMPLS) data packets. This includes a) inband control word if negotiated during signalling and b) MPLS TTL expiry
- Type 2—Specifies a router alert label (label-0).

The router supports Label Switched Path (LSP) VCCV packets of Type 1. The VCCV echo reply is sent as an IPv4 packet, that is, the reply mode is IPv4.

The router does not support accounting of VCCV packets. .

GTP Load Balancing

The GPRS Tunneling Protocol (GTP) Load Balancing feature enables efficient distribution of traffic in mobile networks, and provides increased reliability and availability for the network.

GTP is a tunnel control and management protocol among General Packet Radio Service (GPRS) support nodes. Wireless networks use GTP tunnels to deliver mobile data. GTP includes GTP signaling (GTP-C) and data transfer (GTP-U) procedures. GTP-C specifies a tunnel control and management protocol, and creates, deletes and modifies tunnels. GTP-U uses a tunneling mechanism to provide a service for carrying user data packets over the network.

GTP load balancing is performed on IPv4 or IPv6 incoming packets with GTP payloads and on MPLS incoming labeled packets.

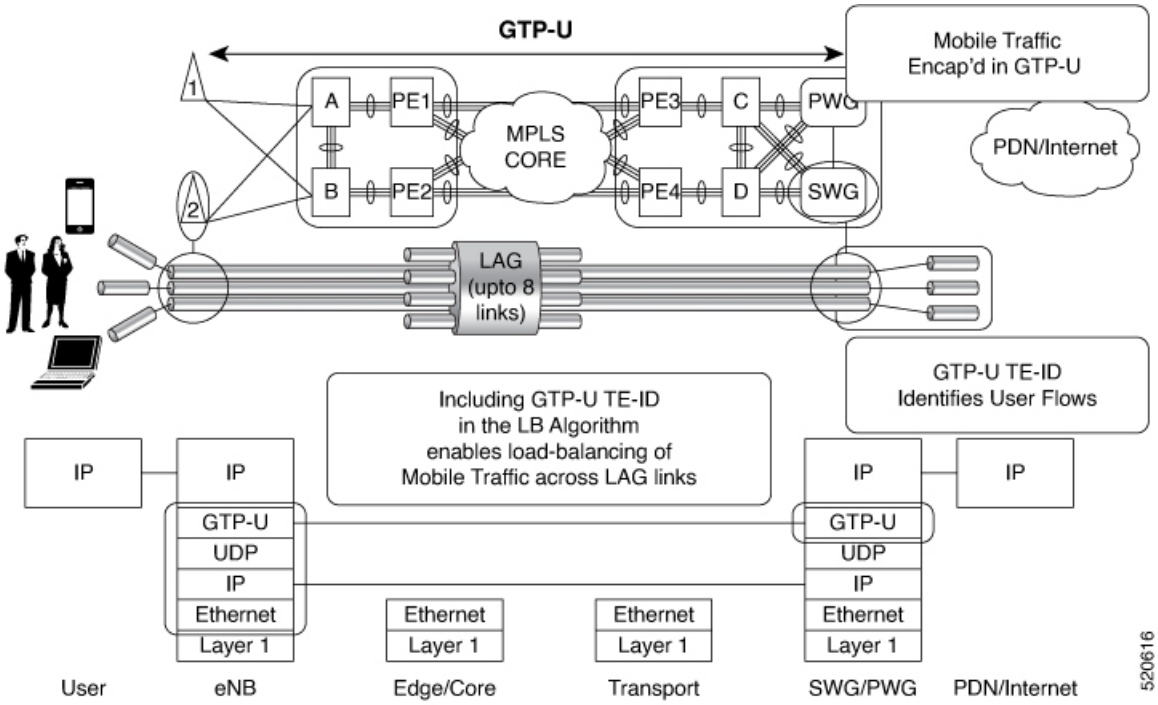
This feature supports GTP hashing only when the GTP UDP port is 2152.

The number of MPLS label stack in the transport layer is limited to three for GTP hashing. GTP hashing is not considered when the MPLS label stack exceeds three.

You need not reload the router after configuring or unconfiguring the **hw-module command** for GTP load balancing over MPLS to take effect.

The following figure shows an illustration of the mobile transport GTP-U load balancing.

Figure 26: Mobile Transport GTP-U Load-Balancing



The global L3 flow-based load balancing considers the following fields:

- source address
- destination address
- router ID
- source port
- destination port

For GTP traffic, however, the number of unique values for these fields is limited; this causes an uneven distribution of traffic. Sometimes, to facilitate redundancy and load balancing in a network, equal cost paths exist to different destinations. Load balancing does not occur in such scenarios as the source and destination IP addresses, as well as L4 ports, are the same. In order to achieve a greater distribution of traffic over equal cost links, load balancing (hashing) must occur on the GTP Tunnel Endpoint Identifier (TEID), which is unique for each traffic flow.

If the packet is UDP and the destination port is the GTP-U port (port number 2152), the GTP TEID is considered for load balancing. This provides GTP load balancing.

The TEID in the GTP header of a GTP packet identifies individual tunnel endpoints, thus achieving better mobile traffic load balancing within any given GRE tunnel. Additionally, this also helps in load balancing GTP traffic over Bundles at transit routers.

Load balancing based on tunnel endpoints is supported for Version 1 GTP packet and GTP version 2, if TEID is present. For GTP version 0, load balancing occurs in the same manner as before, as there is no TEID in version 0.

Note GTP load balancing is performed only for GTP-U (user data) packets. The GTP-C (control data) packets use a different destination port number of 2123 and hence, are subject to only the global L3 flow based load balancing.

By default, load balancing based on GTP-ID when GTP tunnel is over MPLS is disabled.

To enable GTP load balacing over MPLS, configure the **hw-module profile load-balance algorithm gtp-mpls** command.

VPLS over SR-TE and RSVP-TE

Table 17: Feature History Table

Feature Name	Release Information	Feature Description
VPLS over SR-TE and RSVP-TE	Release 7.5.1	For Traffic Engineering, VPLS traffic can be sent using MPLS-TE with RSVP or SR-TE. Resource Reservation Protocol (RSVP) is a signaling protocol that enables systems to request resource reservations from the network. MPLS Traffic Engineering (MPLS-TE) learns the topology and resources available in a network and then maps traffic flows to particular paths, based on resource requirements and network resources such as bandwidth. MPLS-TE uses RSVP to signal LSPs. Segment routing for traffic engineering (SR-TE) uses a “policy” to steer traffic through the network.

Configure VPLS over SR-TE and RSVP-TE

Virtual Private LAN Services (VPLS) enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider.

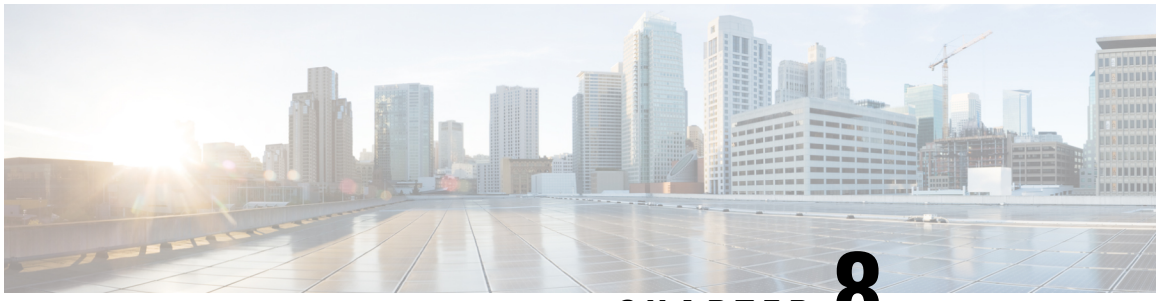
Segment routing for traffic engineering (SR-TE) uses a “policy” to steer traffic through the network. An SR-TE policy path is expressed as a list of segments that specifies the path, called a segment ID (SID) list. Each segment is an end-to-end path from the source to the destination, and instructs the routers in the network to follow the specified path instead of following the shortest path calculated by the IGP.

Resource Reservation Protocol (RSVP) is a signaling protocol that enables systems to request resource reservations from the network. RSVP processes protocol messages from other systems, processes resource requests from local clients, and generates protocol messages. As a result, resources are reserved for data flows on behalf of local and remote clients. RSVP creates, maintains, and deletes these resource reservations.

All L2VPN services such as VPLS, VPWS, and so on must use L2VPN preferred-path while using TE (SR-TE, and RSPV-TE) services as transport.

Perform the following tasks to configure VPLS over SR-TE and RSVP-TE:

- To configure VPLS over SR-TE, see *L2VPN Preferred Path* section in the *Segment Routing Configuration Guide for Cisco NCS Series Routers, IOS XR*
- To configure VPLS over RSVP-TE, see *Implementing RSVP for MPLS-TE* chapter in the *MPLS Configuration Guide for Cisco NCS Series Routers, IOS XR*



CHAPTER 8

Information About Multiple Spanning Tree Protocol

To configure Multiple Spanning Tree Protocol, you must understand these concepts:

- Spanning Tree Protocol Overview, on page 173
- Multiple Spanning Tree Protocol, on page 175
- MSTP Supported Features, on page 178
- Restrictions, on page 179
- Access Gateway, on page 180
- Multiple VLAN Registration Protocol, on page 186
- How to Implement Multiple Spanning Tree Protocol, on page 186
- Configuration Examples for Implementing MSTP, on page 210

Spanning Tree Protocol Overview

Ethernet is no longer just a link-layer technology used to interconnect network vehicles and hosts. Its low cost and wide spectrum of bandwidth capabilities coupled with a simple *plug and play* provisioning philosophy have transformed Ethernet into a legitimate technique for building networks, particularly in the access and aggregation regions of service provider networks.

Ethernet networks lacking a TTL field in the Layer 2 (L2) header and, encouraging or requiring multicast traffic network-wide, are susceptible to broadcast storms if loops are introduced. However, loops are a desirable property as they provide redundant paths. Spanning tree protocols (STP) are used to provide a loop free topology within Ethernet networks, allowing redundancy within the network to deal with link failures.

There are many variants of STP; however, they work on the same basic principle. Within a network that may contain loops, a sufficient number of interfaces are disabled by STP so as to ensure that there is a loop-free spanning tree, that is, there is exactly one path between any two devices in the network. If there is a fault in the network that affects one of the active links, the protocol recalculates the spanning tree so as to ensure that all devices continue to be reachable. STP is transparent to end stations which cannot detect whether they are connected to a single LAN segment or to a switched LAN containing multiple segments and using STP to ensure there are no loops.

STP Protocol Operation

All variants of STP operate in a similar fashion: STP frames (known as bridge protocol data units (BPDUs)) are exchanged at regular intervals over Layer 2 LAN segments, between network devices participating in STP. Such network devices do not forward these frames, but use the information to construct a loop free spanning tree.

The spanning tree is constructed by first selecting a device which is the *root* of the spanning tree (known as the root bridge), and then by determining a loop free path from the *root bridge* to every other device in the network. Redundant paths are disabled by setting the appropriate ports into a blocked state, where STP frames can still be exchanged but data traffic is never forwarded. If a network segment fails and a redundant path exists, the STP protocol recalculates the spanning tree topology and activates the redundant path, by unblocking the appropriate ports.

The selection of the root bridge within a STP network is determined by the lowest Bridge ID which is a combination of configured bridge priority and embedded mac address of each device. The device with the lowest priority, or with equal lowest priority but the lowest MAC address is selected as the root bridge.

Root port: is selected based on lowest root path cost to root bridge. If there is a tie with respect to the root path cost, port on local switch which receives BPDU with lowest sender bridge ID is selected as root port.

Designated port: Least cost port on local switch towards root bridge is selected as designated port. If there is a tie, lowest number port on local switch is selected as designated port.

The selection of the active path among a set of redundant paths is determined primarily by the port path cost. The port path cost represents the cost of transiting between that port and the root bridge - the further the port is from the root bridge, the higher the cost. The cost is incremented for each link in the path, by an amount that is (by default) dependent on the media speed. Where two paths from a given LAN segment have an equal cost, the selection is further determined by the lowest bridge ID of the attached devices, and in the case of two attachments to the same device, by the configured port priority and port ID of the neighboring attached ports.

Once the active paths have been selected, any ports that do not form part of the active topology are moved to the blocking state.

Topology Changes

Network devices in a switched LAN perform MAC learning; that is, they use received data traffic to associate unicast MAC addresses with the interface out of which frames destined for that MAC address should be sent. If STP is used, then a recalculation of the spanning tree (for example, following a failure in the network) can invalidate this learned information. The protocol therefore includes a mechanism to notify topology changes around the network, so that the stale information can be removed (flushed) and new information can be learned based on the new topology.

A *Topology Change* notification is sent whenever STP moves a port from the blocking state to the forwarding state. When it is received, the receiving device flushes the MAC learning entries for all ports that are not blocked other than the one where the notification was received, and also sends its own topology change notification out of those ports. In this way, it is guaranteed that stale information is removed from all the devices in the network.

Variants of STP

There are many variants of the Spanning Tree Protocol:

- Legacy STP (STP)—The original STP protocol was defined in IEEE 802.1D-1998. This creates a single spanning tree which is used for all VLANs and most of the convergence is timer-based.
- Rapid STP (RSTP)—This is an enhancement defined in IEEE 802.1D-2004 to provide more event-based, and hence faster, convergence. However, it still creates a single spanning tree for all VLANs.
- Multiple STP (MSTP)—A further enhancement was defined in IEEE 802.1Q-2005. This allows multiple spanning tree instances to be created over the same physical topology. By assigning different VLANs to the different spanning tree instances, data traffic can be load-balanced over different physical links. The number of different spanning tree instances that can be created is restricted to a much smaller number than the number of possible VLANs; however, multiple VLANs can be assigned to the same spanning tree instance. The BPDUs used to exchange MSTP information are always sent untagged; the VLAN and spanning tree instance data is encoded inside the BPDU.
- Per-Vlan Rapid Spanning Tree (PVRST)— This feature is the IEEE 802.1w (RSTP) standard implemented per VLAN, and is also known as Rapid PVST or PVST+. A single instance of STP runs on each configured VLAN (if you do not manually disable STP). Each Rapid PVST+ instance on a VLAN has a single root switch. You can enable and disable STP on a per-VLAN basis when you are running Rapid PVST+.

  PVRST uses point-to-point wiring to provide rapid convergence of the spanning tree. The spanning tree reconfiguration can occur in less than one second with PVRST (in contrast to 50 seconds with the default settings in the 802.1D STP).
- REP (Cisco-proprietary ring-redundancy protocol)— This is a Cisco-proprietary protocol for providing resiliency in rings. It is included for completeness, as it provides MSTP compatibility mode, using which, it interoperates with an MSTP peer.

Multiple Spanning Tree Protocol

The Multiple Spanning Tree Protocol (MSTP) is a Spanning Tree Protocols (STPs) variant that allows you to create multiple and independent spanning trees over the same physical network. You can configure the parameters for each spanning tree separately. You can select different network devices as the root bridge or different paths to form the loop-free topology. Therefore, you can block a given physical interface for some of the spanning trees and unblock for others.

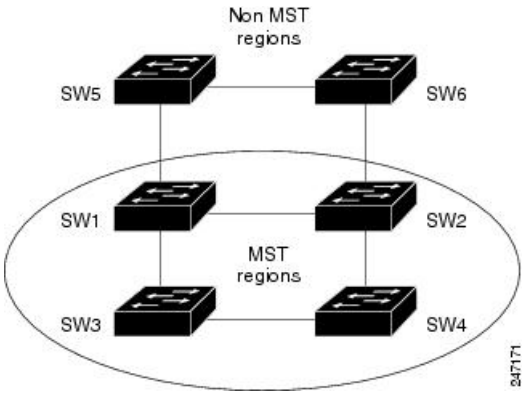
After setting up multiple spanning tree instances, you can partition the set of VLANs in use. For example, you can assign VLANs 1–100 to spanning tree instance 1, VLANs 101–200 to spanning tree instance 2, VLANs 201–300 to spanning tree instance 3, and so on. Since each spanning tree has a different active topology with different active links, this has the effect of dividing the data traffic among the available redundant links based on the VLAN—a form of load balancing.

MSTP Regions

Along with supporting multiple spanning trees, MSTP also introduces the concept of regions. A region is a group of devices under the same administrative control and have similar configuration. In particular, the configuration for the region name, revision, and the mapping of VLANs to spanning tree instances must be identical on all the network devices in the region. A digest of this information is included in the BPDUs sent by each device, so as to allow other devices to verify whether they are in the same region.

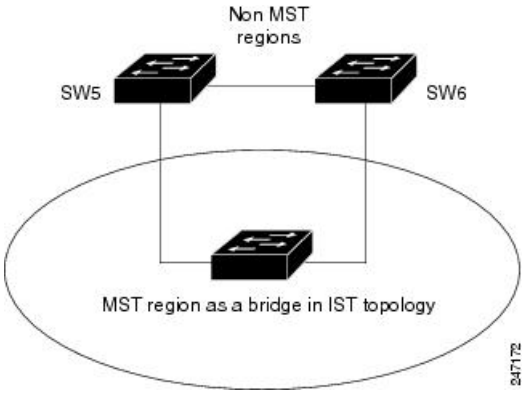
The following figure shows the operation of MST regions when bridges running MSTP are connected to bridges running legacy STP or RSTP. In this example, switches SW1, SW2, SW3, SW4 support MSTP, while switches SW5 and SW6 do not.

Figure 27: MST Interaction with Non-MST Regions



To handle this situation, an Internal Spanning Tree (IST) is used. This is always spanning tree instance 0 (zero). When communicating with non-MSTP-aware devices, the entire MSTP region is represented as a single switch. The logical IST topology in this case is shown in the following figure.

Figure 28: Logical Topology in MST Region Interacting with Non-MST Bridges



The same mechanism is used when communicating with MSTP devices in a different region. For example, SW5 in the above figure could represent a number of MSTP devices, all in a different region compared to SW1, SW2, SW3 and SW4.

MSTP Port Fast

MSTP includes a *Port Fast* feature for handling ports at the edge of the switched Ethernet network. For devices that only have one link to the switched network (typically host devices), there is no need to run MSTP, as there is only one available path. Furthermore, it is undesirable to trigger topology changes (and resultant MAC flushes) when the single link fails or is restored, as there is no alternative path.

By default, MSTP monitors ports where no BPDUs are received, and after a timeout, places them into *edge mode* whereby they do not participate in MSTP. However, this process can be speeded up (and convergence of the whole network thereby improved) by explicitly configuring edge ports as port fast.

Note

- You must disable and re-enable the port for Port Fast configuration to take effect. Use **shutdown** and **no shutdown** command (in interface configuration mode) to disable and re-enable the port.
- Port Fast is implemented as a Cisco-proprietary extension in Cisco implementations of legacy STP. However, it is encompassed in the standards for RSTP and MSTP, where it is known as Edge Port.

MSTP Root Guard

In networks with shared administrative control, it may be desirable for the network administrator to enforce aspects of the network topology and in particular, the location of the root bridge. By default, any device can become the root bridge for a spanning tree, if it has a lower priority or bridge ID. However, a more optimal forwarding topology can be achieved by placing the root bridge at a specific location in the centre of the network.

Note The administrator can set the root bridge priority to 0 in an effort to secure the root bridge position; however, this is no guarantee against another bridge which also has a priority of 0 and has a lower bridge ID.

The root guard feature provides a mechanism that allows the administrator to enforce the location of the root bridge. When root guard is configured on an interface, it prevents that interface from becoming a root port (that is, a port via which the root can be reached). If superior information is received via BPDUs on the interface that would normally cause it to become a root port, it instead becomes a backup or alternate port. In this case, it is placed in the blocking state and no data traffic is forwarded.

The root bridge itself has no root ports. Thus, by configuring root guard on every interface on a device, the administrator forces the device to become the root, and interfaces receiving conflicting information are blocked.

Note Root Guard is implemented as a Cisco-proprietary extension in Cisco implementations of legacy STP and RSTP. However, it is encompassed in the standard for MSTP, where it is known as Restricted Role.

MSTP Topology Change Guard

In certain situations, it may be desirable to prevent topology changes originating at or received at a given port from being propagated to the rest of the network. This may be the case, for example, when the network is not under a single administrative control and it is desirable to prevent devices external to the core of the network from causing MAC address flushing in the core. This behavior can be enabled by configuring Topology Change Guard on the port.

Note Topology Change Guard is known as *Restricted TCN* in the MSTP standard.

MSTP Supported Features

The routers support MSTP, as defined in IEEE 802.1Q-2005, on physical Ethernet interfaces and Ethernet Bundle interfaces.This includes the Port Fast, Backbone Fast, Uplink Fast and Root Guard features found in Cisco implementations of legacy STP, RSTP and PVST, as these are encompassed by the standard MSTP protocol. The routers can operate in either standard 802.1Q mode, or in Provide Edge (802.1ad) mode. In provider edge mode, a different MAC address is used for bridge protocol data units (BPDUs), and any BPDUs received with the 802.1Q MAC address are forwarded transparently.

When you have not configured the **allow-legacy-bpdu** command on MST default instance, and if one of the bridge ports receives legacy BPDU, the port enters **error-disable** state.

Note MSTP supports interoperation with RSTP as described in the 802.1Q standard. However, these features do not support interoperation with legacy STP.

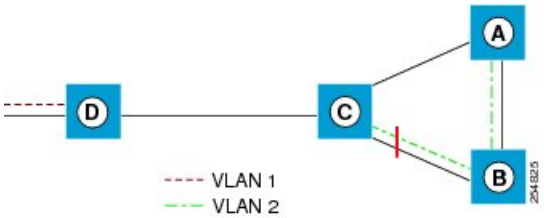
BPDU Guard

The BPDU Guard feature allows you to protect against misconfiguration of edge ports. It is an enhancement to the MSTP port fast feature. When you configure port fast on an interface, MSTP considers that interface to be an edge port and removes it from consideration when calculating the spanning tree. When you configure BPDU Guard, MSTP additionally shuts down the interface using error-disable when an MSTP BPDU is received.

Flush Containment

The Flush Containment feature allows you to prevent unnecessary MAC flushes due to unrelated topology changes in other areas of a network.The following figure shows a network containing four devices. Two VLANs are in use: VLAN 1 is only used on device D, while VLAN 2 spans devices A, B and C. The two VLANs are in the same spanning tree instance, but do not share any links.

Figure 29: Flush Containment



If the link BC goes down, then in normal operation, as C brings up its blocked port, it sends out a topology change notification on all other interfaces, including towards D. This causes a MAC flush to occur for VLAN 1, even though the topology change which has taken place only affects VLAN 2.

Flush containment helps deal with this problem by preventing topology change notifications from being sent on interfaces on which no VLANs are configured for the MSTI in question. In the example network this would mean no topology change notifications would be sent from C to D, and the MAC flushes which take place would be confined to the right hand side of the network.

Bringup Delay

The Bringup Delay feature allows you to stop MSTP from considering an interface when calculating the spanning tree when the interface is not yet ready to forward traffic. This is useful when a line card first boots up, as the system may declare that the interfaces on that card are *Up* before the dataplane is fully ready to forward traffic. According to the standard, MSTP considers the interfaces as soon as they are declared *Up*, and this may cause it to move other interfaces into the blocking state if the new interfaces are selected instead.

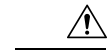
Bringup delay solves this problem by adding a configurable delay period which occurs as interfaces that are configured with MSTP first come into existence. Until this delay period ends, the interfaces remain in blocking state, and are not considered when calculating the spanning tree.

Bringup delay only takes place when interfaces which are already configured with MSTP are created, for example, on a card reload. No delay takes place if an interface which already exists is later configured with MSTP.

Restrictions

These restrictions apply when using MSTP:

- You must enable MSTP must only on interfaces where the interface itself (if it is in L2 mode) or all of the subinterfaces have a simple encapsulation configured. These encapsulation matching criteria are considered simple:
  - Single-tagged 802.1Q frames
  - Double-tagged Q-in-Q frames (only the outermost tag is examined)
  - 802.1ad frames (if MSTP is operating in Provider Bridge mode)
  - Ranges or lists of tags (any of the above)
- If an L2 interface or subinterface is configured with an encapsulation that matches multiple VLANs, then all of those VLANs must be mapped to the same spanning tree instance. There is therefore a single spanning tree instance associated with each L2 interface or subinterface.
- All the interfaces or subinterfaces in a given bridge domain must be associated with the same spanning tree instance.
- Multiple subinterfaces on the same interface must not be associated with the same spanning tree instance, unless those subinterfaces are in the same split horizon group. In other words, hair-pinning is not possible. Across the network, L2 interfaces or subinterfaces must be configured on all redundant paths for all the VLANs mapped to each spanning tree instance. This is to avoid inadvertent loss of connectivity due to STP blocking of a port.



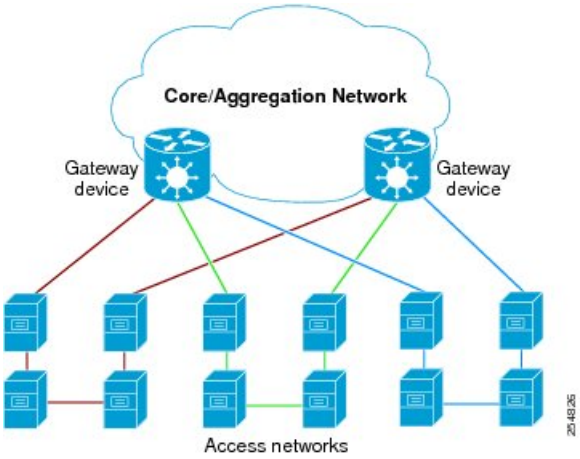
Caution A subinterface with a default or untagged encapsulation leads to an MSTP state machine failure.

- When you have not configured the **allow-legacy-bpdu** command on MST default instance, and if one of the bridge ports receives legacy BPDU, the port enters **error-disable** state.

Access Gateway

One common deployment scenario for router is as an nPE gateway device situated between a network of uPE access devices and a core or aggregation network. Each gateway device may provide connectivity for many access networks, as shown in following figure. The access networks (typically rings) have redundant links to the core or aggregation network, and therefore must use some variant of STP or a similar protocol to ensure the network remains loopfree.

Figure 30: Core or Aggregation Network



It is possible for the gateway devices to also participate in the STP protocol. However, since each gateway device may be connected to many access networks, this would result in one of two solutions:

- A single topology is maintained covering all of the access networks. This is undesirable as it means topology changes in one access network could impact all the other access networks.
- The gateway devices runs multiple instances of the STP protocol, one for each access network. This means a separate protocol database and separate protocol state machines are maintained for each access network, which is undesirable due to the memory and CPU resource that would be required on the gateway device.

It can be seen that both of these options have significant disadvantages.

Another alternative is for the gateway devices to tunnel protocol BPDUs between the *legs* of each access network, but not to participate in the protocol themselves. While this results in correct loopfree topologies, it also has significant downsides:

- Since there is no direct connection between the *legs* of the access ring, a failure in one of the leg links is not immediately detected by the access device connected to the other *leg*. Therefore, recovery from the failure must wait for protocol timeouts, which leads to a traffic loss of at least six seconds.
- As the gateway devices do not participate in the protocol, they are not aware of any topology changes in the access network. The aggregation network may therefore direct traffic destined for the access network over the wrong *leg*, following a topology change. This can lead to traffic loss on the order of the MAC learning timeout (5 minutes by default).

Access gateway is a Cisco feature intended to address this deployment scenario, without incurring the disadvantages of the solutions described above.

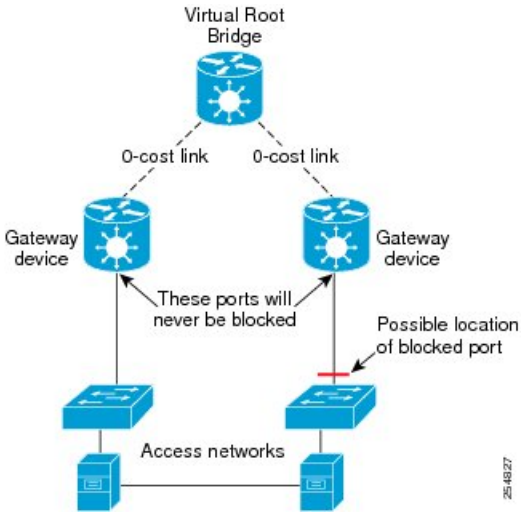
Overview of Access Gateway

Access gateway is based on two assumptions:

- Both gateway devices provide connectivity to the core or aggregation network at all times. Generally, resiliency mechanisms used within the core or aggregation network are sufficient to ensure this is the case. In many deployments, VPLS is used in the core or aggregation network to provide this resiliency.
- The desired root of all of the spanning trees for each access network is one of the gateway devices. This will be the case if (as is typical) the majority of the traffic is between an access device and the core or aggregation network, and there is little if any traffic between the access devices.

With these assumptions, an STP topology can be envisaged where for every spanning tree, there is a virtual root bridge behind (that is, on the core side of) the gateway devices, and both gateway devices have a zero cost path to the virtual root bridge. In this case, the ports that connect the gateway devices to the access network would never be blocked by the spanning tree protocol, but would always be in the forwarding state. This is illustrated in the following figure.

Figure 31: Access Network



With this topology, it can be observed that the BPDUs sent by the gateway devices are constant: since the root bridge never changes (as we assume the aggregation or core network always provides connectivity) and the ports are always forwarding, the information sent in the BPDUs never changes.

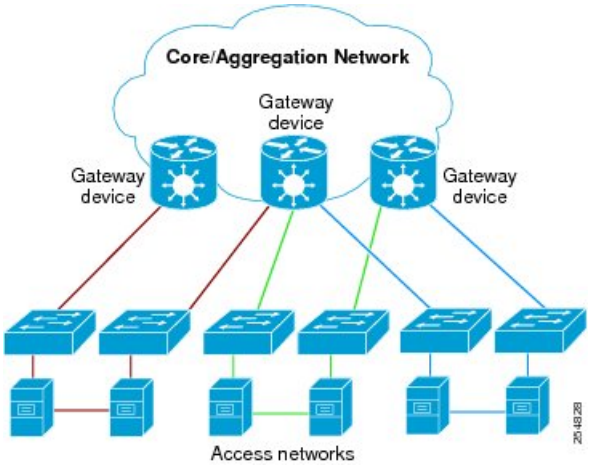
Access gateway makes use of this by removing the need to run the full STP protocol and associated state machines on the gateway devices, and instead just sends statically configured BPDUs towards the access network. The BPDUs are configured so as to mimic the behavior above, so that they contain the same information that would be sent if the full protocol was running. To the access devices, it appears that the gateway devices are fully participating in the protocol; however, since in fact the gateway devices are just sending static BPDUs, very little memory or CPU resource is needed on the gateway devices, and many access networks can be supported simultaneously.

For the most part, the gateway devices can ignore any BPDUs received from the access network; however, one exception is when the access network signals a topology change. The gateway devices can act on this appropriately, for example by triggering an LDP MAC withdrawal in the case where the core or aggregation network uses VPLS.

In many cases, it is not necessary to have direct connectivity between the gateway devices; since the gateway devices statically send configured BPDUs over the access links, they can each be configured independently

(so long as the configuration on each is consistent). This also means that different access networks can use different pairs of gateway devices, as shown in the following figure.

Figure 32: Access Networks



Note Although the above figure shows access rings, in general there are no restrictions on the access network topology or the number or location of links to the gateway devices.

Access gateway ensures loop-free connectivity in the event of these failure cases:

- Failure of a link in the access network.
- Failure of a link between the access network and the gateway device.
- Failure of an access device.
- Failure of a gateway device.

Topology Change Propagation

There is one case where the two gateway devices need to exchange BPDUs between each other, and this is to handle topology changes in the access network. If a failure in the access network results in a topology change that causes a previously blocked port to move to forwarding, the access device sends a topology change notification out on that port, so as to notify the rest of the network about the change and trigger the necessary MAC learning flushes. Typically, the topology change notification is sent towards the root bridge, in the case of access gateway, that means it is sent to one of the gateway devices.

As described above, this causes the gateway device itself to take any necessary action; however, if the failure caused the access network to become partitioned, it may also be necessary to propagate the topology change notification to the rest of the access network, that is, the portion connected to the other gateway device. This can be achieved by ensuring there is connectivity between the gateway devices, so that each gateway device can propagate any topology change notifications it receives from the access network to the other device. When a gateway device receives a BPDU from the other gateway device that indicates a topology change, it signals this in the static BPDUs (that it is sending towards the access network).

Topology Change Propagation is only necessary when these two conditions are met:

- The access network contains three or more access devices. If there are fewer than three devices, then any possible failure must be detected by all the devices.

- The access devices send traffic to each other, and not just to or from the core or aggregation network. If all the traffic is to or from the core or aggregation network, then all the access devices must either already be sending traffic in the right direction, or will learn about the topology change from the access device that originates it.

Preempt Delay

One of the assumptions underpinning access gateway is that the gateway devices are always available to provide connectivity to the core or aggregation network. However, there is one situation where this assumption may not hold, which is at bringup time. At bringup, it may be the case that the access facing interface is available before all of the necessary signaling and convergence has completed that means traffic can successfully be forwarded into the core or aggregation network. Since access gateway starts sending BPDUs as soon as the interface comes up, this could result in the access devices sending traffic to the gateway device before it is ready to receive it. To avoid this problem, the preempt delay feature is used.

The preempt delay feature causes access gateway to send out inferior BPDUs for some period of time after the interface comes up, before reverting to the normal values. These inferior BPDUs can be configured such that the access network directs all traffic to the other gateway device, unless the other gateway device is also down. If the other gateway device is unavailable, it is desirable for the traffic to be sent to this device, even if it is only partially available, rather than being dropped completely. For this reason, inferior BPDUs are sent during the preempt delay time, rather than sending no BPDUs at all.

Supported Access Gateway Protocols

Access Gateway is supported on routers when the following protocols are used in the access network

Table 18: Protocols

Access Network Protocol	Access Gateway Variant
MSTP	MST Access Gateway (MSTAG)
REP	REP Access gateway (REPAG)[1]
PVST+	PVST+ Access Gateway (PVSTAG) [2]
PVRST	PVRST Access Gateway (PVRSTAG) [3]

1. REP Access Gateway is supported when the access device interfaces that connect to the gateway devices are configured with REP MSTP Compatibility mode.
2. Topology Change Propagation is not supported for PVSTAG.
3. Topology Change Propagation is not supported for PVRSTAG.

MSTAG Edge Mode

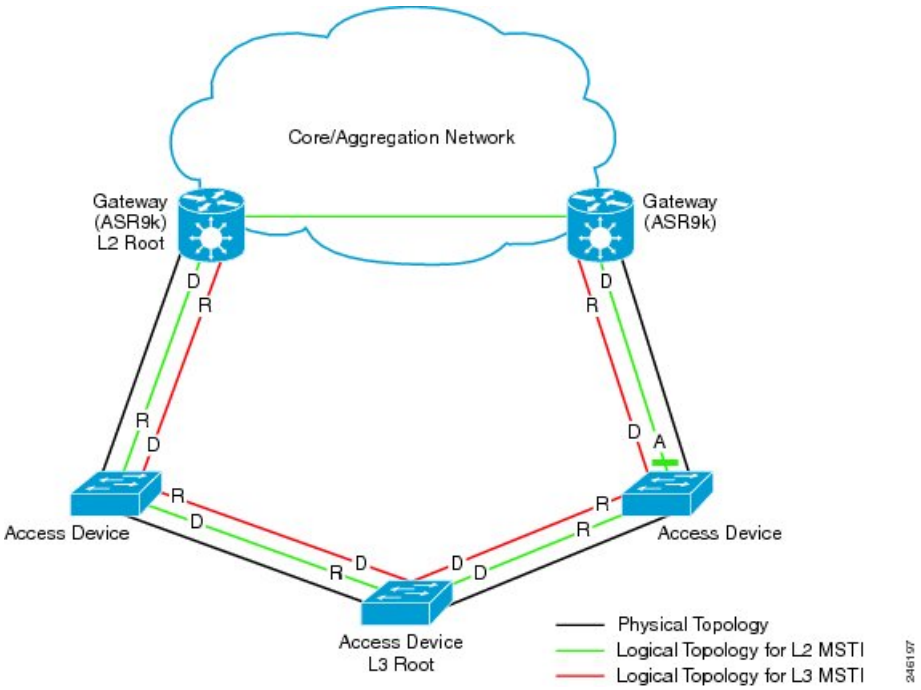
An access gateway is used in a Layer 2 (L2) environment to ensure that for each Multiple Spanning Tree Instance (MSTI), each access device has one path to the core or aggregation network. The core or aggregation network provides L2 (Ethernet) connectivity between two gateway devices. Therefore, when there are no failures, there must be at least one blocked port in the access network for each MSTI. In the case of an access

ring, there should be one blocked port in the access ring. For each MSTI – this is typically one of the uplink ports that connects to one of the gateway devices. This is achieved by configuring MSTAG in such a way that the gateway devices appear to have the best path to the best possible Multiple Spanning Tree Protocol (MSTP) root node. Thus, the access devices always use the gateway devices to reach the root, and the ports on the gateway devices are always in the designated forwarding state.

In a mixed Layer 2-Layer 3 environment, the L2 access network is used to provide a Layer 2 service on certain VLANs and a Layer 3 (L3) service on other VLANs. In the access network, a different MSTI is used for the L2 service and the L3 service. For the L2 VLANs, the core or aggregation network provides L2 connectivity between the gateway devices. However, for the L3 service, the gateway devices terminate the L2 network and perform L3 routing. Typically, an L3 redundancy mechanism such as HSRP or VRRP is used to allow the end hosts to route to the correct gateway.

In this scenario, the use of MSTAG alone does not achieve the desired behavior for the L3 MSTI. This is because it results in one of the ports in the access network being blocked, even though there is actually no loop. (This, in turn, is because there is no L2 connectivity between the gateway devices for the L3 VLANs.) In fact, because the gateway devices terminate the L2 network for the L3 VLANs, the desirable behavior is for the MSTP root to be located in the access network, and for the gateway devices to appear as leaf nodes with a single connection. This can be achieved by reversing the MSTAG configuration; that is, setting the gateway devices to advertise the worst possible path to the worst possible root. This forces the access devices to elect one of the access devices as the root, and therefore, no ports are blocked. In this case, the ports on the gateway devices are always in root forwarding state. The MSTAG Edge mode feature enables this scenario by changing the role advertised by the gateway devices from designated to root. The following figure illustrates this scenario.

Figure 33: MSTAG Edge Mode scenario



- D - Designated port (forwarding)
- R - Root port (forwarding)
- A - Alternate port (blocked)

For normal MSTAG, and for the L2 MSTIs, topology change notifications are propagated from one gateway device to the other, and re-advertised into the access network. However, for the L3 MSTI, this is not desirable. As there is no block for the L3 MSTI in the access network, the topology change notification could loop forever. To avoid that situation, MSTAG Edge mode completely disables handling of topology change notifications in the gateway devices.

PVSTAG on Bundle Interfaces

Per-VLAN Spanning Tree Access Gateway (PVSTAG) support has been extended on bundle interfaces, along with physical interfaces, to cater to an increasing number of customers that support PVST access networks.

For physical interfaces, bridge protocol data units (BDPUs) are sent from the line card that hosts the interfaces. However, for bundle interfaces BPDUs are sent from the route processor (RP). When an RP failover occurs, the data traffic flowing over the bundle interface is not affected; as a result, no BPDUs are sent until the failover is complete and the newly active RP takes over. If there is a delay, the peer device times out the BPDU information. This leads to a forwarding loop, which results in disruption in Ethernet networks. It is therefore important to ensure that, upon RP failover, the peer device does not time out the BPDU information.

Per-VLAN Rapid Spanning Tree

Per-VLAN Rapid Spanning Tree (PVRST) or Rapid PVST or PVST+ is the IEEE 802.1w (RSTP) standard implemented per VLAN. A single instance of STP runs on each configured VLAN (if you do not manually disable STP). Each Rapid PVST+ instance on a VLAN has a single root switch. You can enable and disable STP on a per-VLAN basis when you are running Rapid PVST+.

PVRST uses point-to-point wiring to provide rapid convergence of the spanning tree. The spanning tree reconfiguration can occur in less than 1 second with PVRST (in contrast to 50 seconds with the default settings in the 802.1D STP).

Note PVRST supports one STP instance for each VLAN.

Using PVRST, STP convergence occurs rapidly. Each designated or root port in the STP sends out a BPDU every 2 seconds by default. On a designated or root port in the topology, if hello messages are missed three consecutive times, or if the maximum age expires, the port immediately flushes all protocol information in the table. A port considers that it loses connectivity to its direct neighbor root or designated port if it misses three BPDUs or if the maximum age expires. This rapid aging of the protocol information allows quick failure detection.

PVRST achieves rapid transition to the forwarding state only on edge ports and point-to-point links. Although the link type is configurable, the system automatically derives the link type information from the duplex setting of the port. Full-duplex ports are assumed to be point-to-point ports, while half-duplex ports are assumed to be shared ports.

Disadvantages

- Increased load in terms of increased packet rate.
- Greater CPU and memory utilization due to one STP instance for every LAN.

Implementation of PVRST in IOS-XR

The implementation of PVRST in IOS-XR has the following characteristics:

- Configuration of the Forward Delay and Max Age timers is only supported globally and not per VLAN.
- Configuration of the Hello timer is supported per port and not per VLAN. The Hello timer configured on a port applies to all VLANs on that specific port.
- The cost of a spanning tree bundle port is always 10000. It is not affected by any of the following:
  - Number or speed of the bundle members
  - Logical or administrative operational status of the bundle member ports
  - Addition or deletion of bundle members
- Receiving BPDU on an interface configured with the BPDU Guard error-disables the physical interface as well as any layer-2 or layer-3 sub-interfaces configured on the physical interface.
- Only Ethernet Flow-points (EFPs) that are untagged or have a single VLAN tag can be protected by PVRST.
- If any one EFP in a bridge-domain is protected by PVRST, then all EFPs in that bridge domain must belong to the same VLAN.
- If any one EFP on a port is protected by PVRST, then all EFPs on that port must be protected by PVRST.

Multiple VLAN Registration Protocol

The Multiple VLAN Registration Protocol is defined in IEEE 802.1ak and is used in MSTP based networks to optimize the propagation of multicast and broadcast frames.

By default, multicast and broadcast frames are propagated to every point in the network, according to the spanning tree, and hence to every edge (host) device that is attached to the network. However, for a given VLAN, it may be the case that only certain hosts are interested in receiving the traffic for that VLAN. Furthermore, it may be the case that a given network device, or even an entire segment of the network, has no attached hosts that are interested in receiving traffic for that VLAN. In this case, an optimization is possible by avoiding propagating traffic for that VLAN to those devices that have no stake in it. MVRP provides the necessary protocol signaling that allows each host and device to indicate to its attached peers which VLANs it is interested in.

MVRP-enabled devices can operate in two modes:

- Static mode—In this mode, the device initiates MVRP messages declaring interest in a statically configured set of VLANs. Note that the protocol is still dynamic with respect to the MSTP topology; it is the set of VLANs that is static.
- Dynamic mode—In this mode, the device processes MVRP messages received on different ports, and aggregates them dynamically to determine the set of VLANs it is interested in. It sends MVRP messages declaring interest in this set. In dynamic mode, the device also uses the received MVRP messages to prune the traffic sent out of each port so that traffic is only sent for the VLANs that the attached device has indicated it is interested in.

The router supports operating in static mode. This is known as MVRP-lite.

How to Implement Multiple Spanning Tree Protocol

This section contains these procedures:

Configuring MSTP

This section describes the procedure for configuring MSTP:

Note This section does not describe how to configure data switching. Refer to the Implementing Multipoint Layer 2 Services module for more information.

Enabling MSTP

By default, STP is disabled on all interfaces. MSTP should be explicitly enabled by configuration on each physical or Ethernet Bundle interface. When MSTP is configured on an interface, all the subinterfaces of that interface are automatically MSTP-enabled.

Configuring MSTP parameters

The MSTP Standard defines a number of configurable parameters. The global parameters are:

- Region Name and Revision
- Bringup Delay
- Forward Delay
- Max Age or Hops
- Transmit Hold Count
- Provider Bridge mode
- Flush Containment
- VLAN IDs (per spanning-tree instance)
- Bridge Priority (per spanning-tree instance)

The per-interface parameters are:

- External port path cost
- Hello Time
- Link Type
- Port Fast and BPDU Guard
- Root Guard and Topology Change Guard
- Port priority (per spanning-tree instance)
- Internal port path cost (per spanning-tree instance)

Per-interface configuration takes place in an interface submode within the MST configuration submode.

Note The configuration steps listed in the following sections show all of the configurable parameters. However, in general, most of these can be retained with the default value.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **spanning-tree mst** *protocol instance identifier*

Example:

```
RP/0/RP0/CPU0:router(config)# spanning-tree mst a
RP/0/RP0/CPU0:router(config-mstp)#
```

Enters the MSTP configuration submode.

Step 3 **bringup delay for** *interval* { **minutes** | **seconds** }

Example:

```
RP/0/RP0/CPU0:router(config-mstp)#bringup delay for 10 minutes
```

Configures the time interval to delay bringup for.

Step 4 **flush containment disable**

Example:

```
RP/0/RP0/CPU0:router(config-mstp)#flush containment disable
```

Disable flush containment.

This command performs MAC flush on all instances regardless of the their state.

Step 5 **name** *name*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# name m1
```

Sets the name of the MSTP region.

The default value is the MAC address of the switch, formatted as a text string by means of the hexadecimal representation specified in IEEE Std 802.

Step 6 **revision** *revision* -number

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# revision 10
```

Sets the revision level of the MSTP region.

Allowed values are from 0 through 65535.

Step 7 **forward-delay** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# forward-delay 20
```

Sets the forward-delay parameter for the bridge.

Allowed values for bridge forward-delay time in seconds are from 4 through 30.

Step 8 **maximum** { **age** *seconds* | **hops** *hops* }

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# max age 40
RP/0/RP0/CPU0:router(config-mstp)# max hops 30
```

Sets the maximum age and maximum hops performance parameters for the bridge.

Allowed values for maximum age time for the bridge in seconds are from 6 through 40.

Allowed values for maximum number of hops for the bridge in seconds are from 6 through 40.

Step 9 **transmit hold-count** *count*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# transmit hold-count 8
```

Sets the transmit hold count performance parameter.

Allowed values are from 1 through 10.

Step 10 **provider-bridge**

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# provider-bridge
```

Places the current instance of the protocol in 802.1ad mode.

Step 11 **instance** *id*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# instance 101
```

```
RP/0/RP0/CPU0:router(config-mstp-inst)#
```

Enters the MSTI configuration submode.

Allowed values for the MSTI ID are from 0 through 4094.

Step 12 **priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-inst)# priority 8192
```

Sets the bridge priority for the current MSTI.

Allowed values are from 0 through 61440 in multiples of 4096.

Step 13 **vlan-id** *vlan-range* [,*vlan-range*][,*vlan-range*] [,*vlan-range*]

Example:

```
RP/0/RP0/CPU0:router(config-mstp-inst)# vlan-id 2-1005
```

Associates a set of VLAN IDs with the current MSTI.

List of VLAN ranges in the form a-b, c, d, e-f, g, and so on.

Note
Repeat steps 11 to 13 for each MSTI.

Step 14 **interface** { **Bundle-Ether** | **GigabitEthernet** | **TenGigE** | **FastEthernet** } *instance*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# interface FastEthernet 0/0/0/1
RP/0/RP0/CPU0:router(config-mstp-if)#
```

Enters the MSTP interface configuration submode, and enables STP for the specified port.

Forward interface in Rack/Slot/Instance/Port format.

Step 15 **instance** *id* **port-priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# instance 101 port-priority 160
```

Sets the port priority performance parameter for the MSTI.

Allowed values for the MSTI ID are from 0 through 4094.

Allowed values for port priority are from 0 through 240 in multiples of 16.

Step 16 **instance** *id* **cost** *cost*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# instance 101 cost 10000
```

Sets the internal path cost for a given instance on the current port.

Allowed values for the MSTI ID are from 0 through 4094.

Allowed values for port cost are from 1 through 200000000.

Repeat steps 15 and 16 for each MSTI for each interface.

Step 17 **external-cost** *cost*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# external-cost 10000
```

Sets the external path cost on the current port.

Allowed values for port cost are from 1 through 200000000.

Step 18 **link-type** { **point-to-point** | **multipoint** }

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# link-type point-to-point
```

Sets the link type of the port to point-to-point or multipoint.

Step 19 **hello-time** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# hello-time 1
```

Sets the port hello time in seconds.

Allowed values are 1 and 2.

Step 20 **portfast** [**bpdu-guard**]

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# portfast
RP/0/RP0/CPU0:router(config-mstp-if)# portfast bpduguard
```

Enables PortFast on the port, and optionally enables BPDU guard.

Step 21 **guard root**

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# guard root
```

Enables RootGuard on the port.

Step 22 **guard topology-change**

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# guard topology-change
```

Enables TopologyChangeGuard on the port.

Note
Repeat steps 14 to 22 for each interface.

Step 23 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Verifying MSTP

These show commands allow you to verify the operation of MSTP:

- **show spanning-tree mst** *mst-name*
- **show spanning-tree mst** *mst-name* **interface** *interface-name*
- **show spanning-tree mst** *mst-name* **errors**
- **show spanning-tree mst** *mst-name* **configuration**
- **show spanning-tree mst** *mst-name* **bpdu interface** *interface-name*
- **show spanning-tree mst** *mst-name* **topology-change flushes**

Configuring MSTAG or REPAG

This section describes the procedures for configuring MSTAG:

Note The procedures for configuring REPAG are identical.

This section does not describe how to configure data switching. Refer to the Implementing Multipoint Layer 2 Services module *Implementing Multipoint Layer 2 Services module* for more information.

Configuring an untagged subinterface

In order to enable MSTAG on a physical or Bundle Ethernet interface, an L2 subinterface must first be configured which matches untagged packets, using the encapsulation untagged command.

Enabling MSTAG

MSTAG is enabled on a physical or Bundle Ethernet interface by explicitly configuring it on the corresponding untagged subinterface. When MSTAG is configured on the untagged subinterface, it is automatically enabled on the physical or Bundle Ethernet interface and on all other subinterfaces on that physical or Bundle Ethernet subinterface.

Configuring MSTAG parameters

MSTAG parameters are configured separately on each interface, and MSTAG runs completely independently on each interface. There is no interaction between the MSTAG parameters on different interfaces (unless they are connected to the same access network).

These parameters are configurable for each interface:

- Region Name and Revision
- Bridge ID
- Port ID
- External port path cost
- Max Age
- Provide Bridge mode
- Hello Time

The following MSTAG parameters are configurable for each interface, for each spanning tree instance:

- VLAN IDs
- Root Bridge Priority and ID
- Bridge Priority
- Port Priority
- Internal Port Path Cost

To ensure consistent operation across the access network, these guidelines should be used when configuring:

- Both gateway devices should be configured with a Root Bridge Priority and ID (for each spanning tree instance) that is better (lower) than the Bridge Priority and Bridge ID of any device in the access network. It is recommended to set the Root Bridge Priority and ID to 0 on the gateway devices.

Note To avoid an STP dispute being detected by the access devices, the same root priority and ID should be configured on both gateway devices.

- Both gateway devices should be configured with a Port Path Cost of 0.
- For each spanning tree instance, one gateway device should be configured with the bridge priority and ID that is higher than the root bridge priority and ID, but lower than the bridge priority and ID of any other device in the network (including the other gateway device). It is recommended to set the bridge priority to 0.

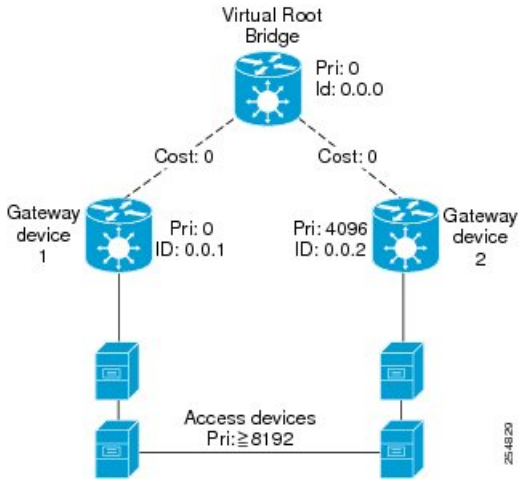
- For each spanning tree instance, the second gateway device should be configured with a bridge priority and ID that is higher than the root bridge priority and ID and the first gateway device bridge priority and ID, but lower than the bridge priority and ID of any device in the access network. It is recommended to set the bridge priority to 4096 (this is the lowest allowable value greater than 0).
- All of the access devices should be configured with a higher bridge priority than the gateway devices. It is recommended to use values of 8192 or higher.
- For each spanning tree instance, the port path cost and other parameters may be configured on the access devices so as to ensure the desired port is put into the blocked state when all links are up.

Caution There are no checks on MSTAG configuration—misconfiguration may result in incorrect operation of the MSTP protocol in the access devices (for example, an STP dispute being detected).

The guidelines above are illustrated in the following figure.

Note These guidelines do not apply to REPAG, as in that case the access devices ignore the information received from the gateway devices apart from when a topology change is signalled.

Figure 34: MSTAG Guidelines



Note The configuration steps listed in the following sections show all of the configurable parameters. However, in general, most of these can be retained with the default values.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **spanning-tree mstag** *protocol instance identifier*

Example:

```
RP/0/RP0/CPU0:router(config)# spanning-tree mstag a
RP/0/RP0/CPU0:router(config-mstag)#
```

Enters the MSTAG configuration submode.

Step 3 **preempt delay for** interval{ **seconds** | **minutes** | **hours** }

Example:

```
RP/0/RP0/CPU0:router(config-mstag)# preempt delay for 10 seconds
```

Specifies the delay period during which startup BPDUs should be sent, before preempting.

Step 4 **interface** { **Bundle-Ether** | **GigabitEthernet** | **TenGigE** | **FastEthernet** } *instance.subinterface*

Example:

```
RP/0/RP0/CPU0:router(config-mstag)# interface GigabitEthernet0/2/0/30.1
RP/0/RP0/CPU0:router(config-mstag-if)#
```

Enters the MSTAG interface configuration submode, and enables MSTAG for the specified port.

Step 5 **name** *name*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# name leo
```

Sets the name of the MSTP region.

The default value is the MAC address of the switch, formatted as a text string using the hexadecimal representation specified in IEEE Standard 802.

Step 6 **revision** *revision* -number

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# revision 1
```

Sets the revision level of the MSTP region.

Allowed values are from 0 through 65535.

Step 7 **max age** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# max age 20
```

Sets the maximum age performance parameters for the bridge.

Allowed values for the maximum age time for the bridge in seconds are from 6 through 40.

Step 8 **provider-bridge**

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# provider-bridge
```

Places the current instance of the protocol in 802.1ad mode.

Step 9 **bridge-id** *id*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# bridge-id 001c.0000.0011
```

Sets the bridge ID for the current switch.

Step 10 **port-id** *id*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# port-id 111
```

Sets the port ID for the current switch.

Step 11 **external-cost** *cost*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# external-cost 10000
```

Sets the external path cost on the current port.

Allowed values for port cost are from 1 through 200000000.

Step 12 **hello-time** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# hello-time 1
```

Sets the port hello time in seconds.

Allowed values are from 1 through 2.

Step 13 **instance** *id*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if)# instance 1
```

Enters the MSTI configuration submode.

Allowed values for the MSTI ID are from 0 through 4094.

Step 14 **edge mode**

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if-inst)# edge mode
```

Enables access gateway edge mode for this MSTI.

Step 15 **vlan-id** *vlan-range* [, *vlan-range*] [,*vlan-range*] [,*vlan-range*]

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if-inst)# vlan-id 2-1005
```

Associates a set of VLAN IDs with the current MSTI.

List of VLAN ranges in the form a-b, c, d, e-f, g, and so on.

Step 16 **priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if-inst)# priority 4096
```

Sets the bridge priority for the current MSTI.

Allowed values are from 0 through 61440 in multiples of 4096.

Step 17 **port-priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if-inst)# port-priority 160
```

Sets the port priority performance parameter for the MSTI.

Allowed values for port priority are from 0 through 240 in multiples of 16.

Step 18 **cost** *cost*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if-inst)# cost 10000
```

Sets the internal path cost for a given instance on the current port.

Allowed values for port cost are from 1 through 200000000.

Step 19 **root-bridge** *id*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if-inst)# root-id 001c.0000.0011
```

Sets the root bridge ID for the BPDUs sent from the current port.

Step 20 **root-priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-mstag-if-inst)# root-priority 4096
```

Sets the root bridge priority for the BPDUs sent from this port.

Note
Repeat steps 4 to 19 to configure each interface, and repeat steps 13 to 19 to configure each MSTI for each interface.

Step 21 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring MSTAG Topology Change Propagation

MSTAG Topology Change Propagation is configured simply by configuring connectivity between the MSTAG-enabled interfaces on the two gateway devices:

1. Configure MSTAG as described above. Take note of the untagged subinterface that is used.
2. Configure connectivity between the gateway devices. This may be via an MPLS Pseudowire, or may be a VLAN subinterface if there is a direct physical link.
3. Configure a point-to-point (P2P) cross-connect on each gateway device that contains the untagged subinterface and the link (PW or subinterface) to the other gateway device.

Once the untagged subinterface that is configured for MSTAG is added to the P2P cross-connect, MSTAG Topology Change Propagation is automatically enabled. MSTAG forwards BDPUs via the cross-connect to the other gateway device, so as to signal when a topology change has been detected.

For more information on configuring MPLS pseudowire or P2P cross-connects, refer to the Implementing Point to Point Layer 2 Services module.

Verifying MSTAG

These show commands allow you to verify the operation of MSTAG:

- **show spanning-tree mstag** *mst-name*

- **show spanning-tree mstag** *mst-name* **bpdu interface** *interface-name*
- **show spanning-tree mstag** *mst-name* **topology-change flushes**

Analogous commands are available for REPAG.

MSTAG Uplink Tracking

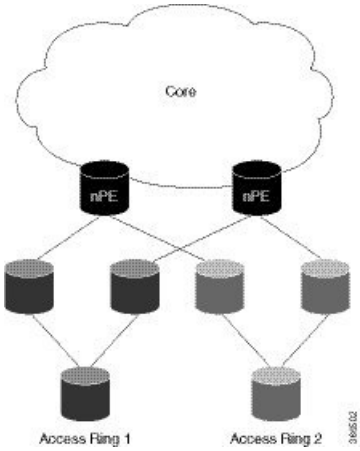
The MSTAG Uplink Tracking feature monitors the connectivity of an nPE gateway device to the core or aggregation network. This feature prevents traffic loss if there is a connectivity failure between a gateway router and the core network. This feature also ensures reduction in the frequency of traffic outages and reduced need for redundancy in connectivity from the gateway to the core network.

Multiple Spanning Tree Access Gateway (MSTAG) tracks the connectivity of interfaces that face the core. When an nPE device loses connectivity to the core, it sends start-up BPDUs indicating that all core-facing interfaces are down. This allows the access ring to switch the traffic to go through the other nPE device.

The core connectivity is based on a per-protocol instance. This is because each access ring corresponds to an access ring, and they may use different interfaces to forward traffic to the core. Therefore, it is possible for different protocol instances to have different core connectivity status.

In this topology there are two access rings. Each one is connected to the core through a pair of nPE devices in which MSTAG is configured. MSTAG allows each access ring to run the STP independently. When one of the nPE devices lose core connectivity, it starts sending start-up BPDUs indicating that traffic must flow through the other side of the access ring. Once core connectivity is available, the nPE device starts sending the standard BPDUs. If pre-empt delay is set, it continues to send the start-up BPDUs until the timer expires. For example, if pre-empt delay is configured as ten seconds, the nPE device sends startup BPDUs for the first ten seconds after core connectivity becomes available. After ten seconds, it starts sending standard BPDUs.

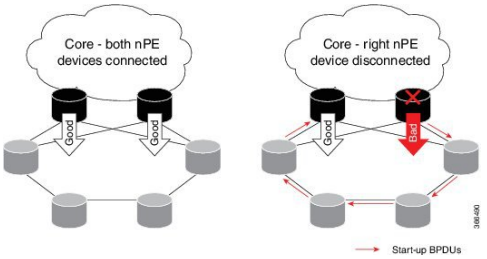
Figure 35: MSTAG Uplink Tracking



Core Connectivity Failure

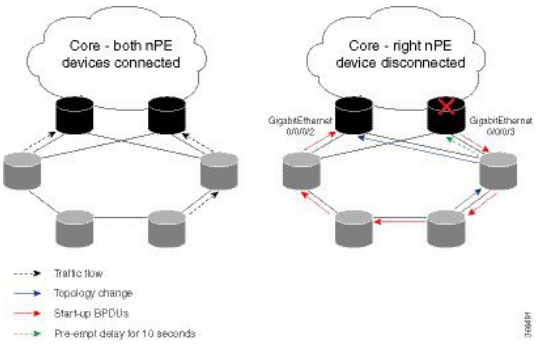
The following diagram illustrates how the nPE device sends start-up BPDUs when it loses core connectivity.

Figure 36: Core connectivity Failure



The device that loses core connectivity starts sending start-up BPDUs. The following diagram illustrates how traffic flows in the access network when an nPE device loses core connectivity.

Figure 37: Active Topology Change



Start-up BPDUs are sent from the disconnected router to ensure that it is not used as a path to the core from the access ring. This changes the active topology in the access rings depending on the STP configuration.

Benefits

The MSTAG Uplink Tracking feature has these benefits:

- Reduction in the frequency of traffic outages
- Reduced need for redundancy in connectivity from the gateway to the core network

Prerequisites

- Configure MSTAG

Restrictions

Only physical and bundle interfaces, and their sub interfaces can be tracked for core connectivity. Psuedowires themselves cannot be tracked, only the underlying interface carrying the psuedowire traffic can be tracked.

Configure MSTAG Uplink Tracking

Only physical and bundle interfaces, and their sub-interfaces can be tracked for core connectivity. Pseudowires themselves cannot be tracked, only the underlying interface carrying the pseudowire traffic can be tracked.

Perform this task to configure MSTAG Uplink Tracking feature.

```
/* Configure MSTAG for a protocol instance called 'foo', with two interfaces facing the
access ring and pre-empt delay of ten seconds */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# spanning-tree mstag foo
RP/0/RSP0/CPU0:router(config-mstag)# preempt delay for 10 seconds
RP/0/RSP0/CPU0:router(config-mstag)# interface GigabitEthernet0/0/0/0
RP/0/RSP0/CPU0:router(config-mstag-if)# exit
RP/0/RSP0/CPU0:router(config-mstag)# interface GigabitEthernet0/0/0/1
RP/0/RSP0/CPU0:router(config-mstag-if)# commit
RP/0/RSP0/CPU0:router(config-mstag-if)# root

/* Configure Uplink Tracking by adding core-facing interfaces under the 'track' keyword */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# spanning-tree mstag foo
RP/0/RSP0/CPU0:router(config-mstag)#track
RP/0/RSP0/CPU0:router(config-mstag-track)# interface GigabitEthernet0/0/0/2
RP/0/RSP0/CPU0:router(config-mstag-track)# interface GigabitEthernet0/0/0/3
RP/0/RSP0/CPU0:router(config-mstag-track)# commit

/* When pre-empt timer is zero, traffic flows through interface GigabitEthernet0/0/0/3 as
soon as this interface comes up.*/

/* In this configuration, pre-empt delay is set to 10 seconds. Traffic flows through interface
 GigabitEthernet0/0/0/3, ten seconds after this interface comes up. */
```

Running Configuration

```
configure
  spanning-tree mstag foo
    interface GigabitEthernet 0/0/0/0
    !
    interface GigabitEthernet 0/0/0/1
    !
    track
      interface GigabitEthernet 0/0/0/2
      interface GigabitEthernet 0/0/0/3
    !
    pre-empt delay for 10 seconds
  !
!
```

Verification

Verify the core connectivity status. The following example shows the state of the interfaces that connect the core.

```
RP/0/0/CPU0:ios#show spanning-tree mstag foo tracked
Core Connectivity Available: True
Tracked Items:               1/2 up

Interface Name                        State
-----------------------------------------------
GigabitEthernet0/0/0/3                Down
GigabitEthernet0/0/0/2                Up
-----------------------------------------------
```

Configuring PVSTAG or PVRSTAG

This section describes the procedures for configuring PVSTAG:

The procedures for configuring PVRSTAG are identical.

Note This section does not describe how to configure data switching. Refer to the *Implementing Multipoint Layer 2 Services* module for more information.

Enabling PVSTAG

PVSTAG is enabled for a particular VLAN, on a physical interface, by explicit configuration of that physical interface and VLAN for PVSTAG.

Configuring PVSTAG parameters

The configurable PVSTAG parameters for each interface on each VLAN are:

- Root Priority and ID
- Root cost
- Bridge Priority and ID
- Port priority and ID
- Max Age
- Hello Time

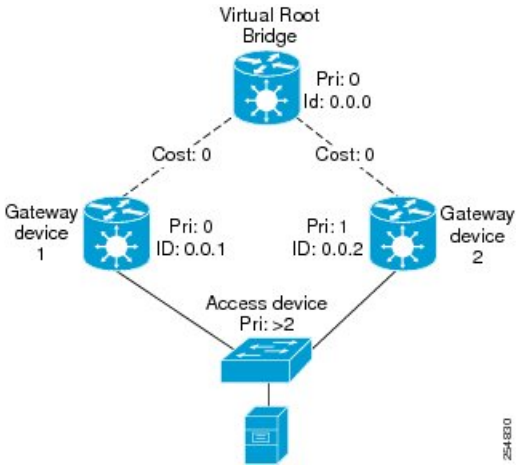
For correct operation, these guidelines must be followed when configuring PVSTAG.

- Both gateway devices should be configured with a root bridge priority and ID that is better (lower) than the bridge priority and Bridge ID of any device in the access network. It is recommended that you set the root bridge priority and ID to 0 on the gateway devices.
- Both gateway devices should be configured with a root cost of 0.
- One gateway device should be configured with the bridge priority and ID that is higher than the root bridge priority and ID, but lower than the bridge priority and ID of any other device in the network (including the other gateway device). It is recommended that you set the bridge priority to 0.
- The second gateway device should be configured with a bridge priority and ID that is higher than the root bridge priority and ID and the first gateway device bridge priority and ID, but lower than the bridge priority and ID of any device in the access network. It is recommended that you set the bridge priority to 1 for PVSTAG or 4096 for PVRSTAG. (For PVRSTAG, this is the lowest allowable value greater than 0.)
- All access devices must be configured with a higher bridge priority than the gateway devices. It is recommended that you use values of 2 or higher for PVSTAG, or 8192 or higher for PVRSTAG.
- For each spanning tree instance, the port path cost and other parameters may be configured on the access devices, so as to ensure the desired port is placed into the blocked state when all links are up.

Caution There are no checks on PVSTAG configuration—misconfiguration may result in incorrect operation of the PVST protocol in the access devices (for example, an STP dispute being detected).

These guidelines are illustrated in the following figure:

Figure 38: PVSTAG Guidelines



Note The configuration steps listed in the following sections show all of the configurable parameters. However, in general, most of these can be retained with the default values.

PVSTAG Topology Restrictions

These restrictions are applicable to PVSTAG topology:

- Only a single access device can be attached to the gateway devices.
- Topology change notifications on a single VLAN affect all VLANs and bridge domains on that physical interface.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **spanning-tree mstag pvstag** *protocol instance identifier*

Example:

```
RP/0/RP0/CPU0:router(config)# spanning-tree pvstag a
RP/0/RP0/CPU0:router(config-pvstag)#
```

Enters the PVSTAG configuration submode.

Step 3 **preempt delay for** *interval* { **seconds | minutes | hours** }

Example:

```
RP/0/RP0/CPU0:router(config-pvstag)# preempt delay for 10 seconds
```

Specifies the delay period during which startup BPDUs should be sent, before preempting.

Step 4 **interface type** *interface-path-id* or **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag)# interface GigabitEthernet0/2/0/30.1
RP/0/RP0/CPU0:router(config-pvstag-if)#

or

RP/0/RP0/CPU0:router(config-pvstag)# interface Bundle-Ether 100
RP/0/RP0/CPU0:router(config-pvstag-if)#
```

Enters the PVSTAG interface configuration submode, and enables PVSTAG for the specified port.

Step 5 **vlan** *vlan-id*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if)# vlan 200
```

Enables and configures a VLAN on this interface.

Step 6 **root-priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# root-priority 4096
```

Sets the root bridge priority for the BPDUs sent from this port.

Step 7 **root-id** *id*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# root-id 0000.0000.0000
```

Sets the identifier of the root bridge for BPDUs sent from a port.

Step 8 **root-cost** *cost*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# root-cost 10000
```

Set the root path cost to sent in BPDUs from this interface.

Step 9 **priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# priority 4096
```

Sets the bridge priority for the current MSTI.

For PVSTAG, allowed values are from are 0 through 65535; for PVRSTAG, the allowed values are from 0 through 61440 in multiples of 4096.

Step 10 **bridge-id** *id*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# bridge-id 001c.0000.0011
```

Sets the bridge ID for the current switch.

Step 11 **port-priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# port-priority 160
```

Sets the port priority performance parameter for the MSTI.

For PVSTAG, allowed values for port priority are from 0 through 255; for PVRSTAG, the allowed values are from 0 through 240 in multiples of 16.

Step 12 **port-id** *id*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# port-id 111
```

Sets the port ID for the current switch.

Step 13 **hello-time** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# hello-time 1
```

Sets the port hello time in seconds.

Allowed values are from 1 through 2.

Step 14 **max age** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-pvstag-if-vlan)# max age 20
```

Sets the maximum age performance parameters for the bridge.

Allowed values for the maximum age time for the bridge in seconds are from 6 through 40.

Note
Repeat steps 4 to 14 to configure each interface; repeat steps 5 to 14 to configure each VLAN on each interface.

Step 15 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Subinterfaces

For each VLAN that is enabled for PVSTAG on an interface, a corresponding subinterface that matches traffic for that VLAN must be configured. This is used both for data switching and for PVST BPDUs. Follow these guidelines when configuring subinterfaces:

- VLAN 1 is treated as the native VLAN in PVST. Therefore, for VLAN 1, a subinterface that matches untagged packets (**encapsulation untagged**) must be configured. It may also be necessary to configure a subinterface that matches packets tagged explicitly with VLAN 1 (**encapsulation dot1q 1**).
- Only dot1q packets are allowed in PVST; Q-in-Q and dot1ad packets are not supported by the protocol, and therefore subinterfaces configured with these encapsulation will not work correctly with PVSTAG.
- Subinterfaces that match a range of VLANs are supported by PVSTAG; it is not necessary to configure a separate subinterface for each VLAN, unless it is desirable for provisioning the data switching.
- PVSTAG does not support:
  - Physical interfaces configured in L2 mode
  - Subinterface configured with a default encapsulation (**encapsulation default**)
  - Subinterfaces configured to match any VLAN (**encapsulation dot1q any**)

For more information about configuring L2 subinterfaces, refer to the Implementing Point to Point Layer 2 Services *Implementing Point to Point Layer 2 Services* module.

Verifying PVSTAG

These show commands allow you to verify the operation of PVSTAG or PVRSTAG:

- **show spanning-tree pvstag** *mst-name*
- **show spanning-tree pvstag** *mst-name*

In particular, these commands display the subinterface that is being used for each VLAN.

Configuring PVRST

Before you begin

Ensure that:

- L2transport subinterfaces with VLAN encapsulation are defined.
- L2VPN bridge domains under bridge group for every VLAN running spanning tree are configured, and corresponding l2transport subinterfaces are configured in the bridge-domain.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **spanning-tree pvrst** *protocol-instance-name*

Example:

```
RP/0/RP0/CPU0:router(config)# spanning-tree pvrst stp
```

Enters the PVRST configuration submode.

Step 3 **apply-group** *group_name group_name*

Example:

```
RP/0/RP0/CPU0:router(config-pvrst)# apply-group groupA groupB
```

Allows you to apply configuration from one group to another.

Step 4 **forward-delay** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-pvrst)# forward-delay 10
```

Allows you to configure bridge forward delay time in seconds.

The forward delay is the number of seconds a port waits before changing from its spanning-tree learning and listening states to the forwarding state. The delay time range is from 4 to 30.

Step 5 **maximum age** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-pvrst)# maximum age 10
```

Specifies maximum age for the bridge in seconds.

The maximum-aging time is the number of seconds a switch waits without receiving spanning-tree configuration messages before attempting a reconfiguration. The maximum age range is from 6 to 40 seconds.

Step 6 **transmit hold-count** *count*

Example:

```
RP/0/RP0/CPU0:router(config-pvrst)# transmit hold-count 4
```

Allows you to configure bridge transmit hold count. The hold count range is from 1 to 10.

Step 7 **vlan** *vlan_id*

Example:

```
RP/0/RP0/CPU0:router(config-pvrst)#
```

Allows you to configure PVRST on a VLAN. The VLAN ID range is from 1 to 4094.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring MVRP-lite

This section describes the procedure for configuring MVRP-lite:

Configuring MVRP-lite

This section describes the procedure for configuring MVRP-lite:

Configuring MVRP-lite parameters

The configurable MVRP-lite parameters are:

- Periodic Transmission
- Join Time
- Leave Time

- Leave-all Time

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters XR Config mode.

Step 2 **spanning-tree mst** *protocol instance identifier*

Example:

```
RP/0/RP0/CPU0:router(config)#
spanning-tree mst aRP/0/RSP0/CPU0:router(config-mstp)#
```

Enters the MSTP configuration submode.

Step 3 **mvrp static**

Example:

```
RP/0/RP0/CPU0:router(config-mstp)#mvrp static
```

Configures MVRP to run over this MSTP protocol instance.

Step 4 **periodic transmit** [**interval** *seconds*]

Example:

```
RP/0/RP0/CPU0:router(config-mvrp)#
periodic transmit
```

Sends periodic Multiple VLAN Registration Protocol Data Unit (MVRPDU) on all active ports.

Step 5 **join-time** *milliseconds*

Example:

```
RP/0/RP0/CPU0:router(config-mvrp)#
 hello-time 1
```

Sets the join time for all active ports.

Step 6 **leave-time** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-mvrp)# leave-time 20
```

Sets the leave time for all active ports.

Step 7 **leaveall-time** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-mvrp)# leaveall-time 20
```

Sets the leave all time for all active ports.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Verifying MVRP-lite

These show commands allow you to verify the operation of MVRP-lite:

- **show ethernet mvrp mad**
- **show ethernet mvrp status**
- **show ethernet mvrp statistics**

Configuration Examples for Implementing MSTP

This section provides configuration examples for the following:

Configuring MSTP: Examples

This example shows MSTP configuration for a single spanning-tree instance with MSTP enabled on a single interface:

```
config
spanning-tree mst example
    name m1
    revision 10
    forward-delay 20
    maximum hops 40
    maximum age 40
    transmit hold-count 8
    provider-bridge
    bringup delay for 60 seconds
    flush containment disable
    instance 101
        vlans-id 101-110
        priority 8192
    !
```

```
    interface GigabitEthernet0/0/0/0
        hello-time 1
        external-cost 10000
        link-type point-to-point
        portfast
        guard root
        guard topology-change
        instance 101 cost 10000
        instance 101 port-priority 160
    !
!
```

This example shows the output from the **show spanning-tree mst** command, which produces an overview of the spanning tree protocol state:

```
# show spanning-tree mst example
Role:  ROOT=Root, DSGN=Designated, ALT=Alternate, BKP=Backup, MSTR=Master
State: FWD=Forwarding, LRN=Learning, BLK=Blocked, DLY=Bringup Delayed

Operating in dot1q mode


MSTI 0 (CIST):

  VLANS Mapped: 1-9,11-4094

  CIST Root  Priority    4096
             Address     6262.6262.6262
             This bridge is the CIST root
             Ext Cost    0

  Root ID    Priority    4096
             Address     6262.6262.6262
             This bridge is the root
             Int Cost    0
             Max Age 20 sec, Forward Delay 15 sec


  Bridge ID  Priority    4096 (priority 4096 sys-id-ext 0)
             Address     6262.6262.6262
             Max Age 20 sec, Forward Delay 15 sec
             Max Hops 20, Transmit Hold count  6
Interface    Port ID           Role State Designated           Port ID
             Pri.Nbr Cost                 Bridge ID            Pri.Nbr
------------ ------- --------- ---- ----- -------------------- -------
Gi0/0/0/0    128.1   20000     DSGN FWD   4096  6262.6262.6262 128.1
Gi0/0/0/1    128.2   20000     DSGN FWD   4096  6262.6262.6262 128.2
Gi0/0/0/2    128.3   20000     DSGN FWD   4096  6262.6262.6262 128.3
Gi0/0/0/3    128.4   20000     ---- BLK   ----- -------------- -------


MSTI 1:

  VLANS Mapped: 10

  Root ID    Priority    4096
             Address     6161.6161.6161
             Int Cost    20000
             Max Age 20 sec, Forward Delay 15 sec


  Bridge ID  Priority    32768 (priority 32768 sys-id-ext 0)
             Address     6262.6262.6262
             Max Age 20 sec, Forward Delay 15 sec
```

```
              Max Hops 20, Transmit Hold count  6

Interface    Port ID           Role State Designated           Port ID
             Pri.Nbr Cost                 Bridge ID            Pri.Nbr
------------ ------- --------- ---- ----- -------------------- -------
Gi0/0/0/0    128.1   20000     ROOT FWD   4096  6161.6161.6161 128.1
Gi0/0/0/1    128.2   20000     ALT  BLK   4096  6161.6161.6161 128.2
Gi0/0/0/2    128.3   20000     DSGN FWD   32768 6262.6262.6262 128.3
Gi0/0/0/3    128.4   20000     ---- BLK   ----- -------------- -------

================================================================================
```

In the **show spanning-tree mst** example output, the first line indicates whether MSTP is operating in dot1q or the Provider Bridge mode, and this information is followed by details for each MSTI.

For each MSTI, the following information is displayed:

- The list of VLANs for the MSTI.
- For the CIST, the priority and bridge ID of the CIST root, and the external path cost to reach the CIST root. The output also indicates if this bridge is the CIST root.
- The priority and bridge ID of the root bridge for this MSTI, and the internal path cost to reach the root. The output also indicates if this bridge is the root for the MSTI.
- The max age and forward delay times received from the root bridge for the MSTI.
- The priority and bridge ID of this bridge, for this MSTI.
- The maximum age, forward delay, max hops and transmit hold-count for this bridge (which is the same for every MSTI).
- A list of MSTP-enabled interfaces. For each interface, the following information is displayed:
  - The interface name
  - The port priority and port ID for this interface for this MSTI.
  - The port cost for this interface for this MSTI.
  - The current port role:
    - DSGN—Designated: This is the designated port on this LAN, for this MSTI
    - ROOT—Root: This is the root port for the bridge for this MSTI.
    - ALT—Alternate: This is an alternate port for this MSTI.
    - BKP—Backup: This is a backup port for this MSTI
    - MSTR—Master: This is a boundary port that is a root or alternate port for the CIST.

    The interface is down, or the bringup delay timer is running and no role has been assigned yet.
  - The current port state:
    - BLK—The port is blocked.
    - LRN—The port is learning.
    - FWD—The port is forwarding.

- DLY—The bringup-delay timer is running.

- If the port is a boundary port, and not CIST and the port is not designated, then only the BOUNDARY PORT is displayed and the remaining information is not displayed.
- If the port is not up, or the bringup delay timer is running, no information is displayed for the remaining fields. Otherwise, the bridge priority and bridge ID of the designated bridge on the LAN that the interface connects to is displayed, followed by the port priority and port ID of the designated port on the LAN. If the port role is Designated, then the information for this bridge or port is displayed.

The following example shows the output from the **show spanning-tree mst** command, which produces more detailed information regarding interface state than the standard command as described above:

```
# show spanning-tree mst a interface GigabitEthernet0/1/2/1
GigabitEthernet0/1/2/1
Cost: 20000
link-type: point-to-point
hello-time 1
Portfast: no
BPDU Guard: no
Guard root: no
Guard topology change: no
BPDUs sent 492, received 3

MST 3:
 Edge port:
 Boundary : internal
 Designated forwarding
 Vlans mapped to MST 3: 1-2,4-2999,4000-4094
 Port info port id 128.193 cost 200000
 Designated root address 0050.3e66.d000 priority 8193 cost 20004
 Designated bridge address 0002.172c.f400 priority 49152 port id 128.193
 Timers: message expires in 0 sec, forward delay 0, forward transitions 1
 Transitions to reach this state: 12
```

The output includes interface information about the interface which applies to all MSTIs:

- Cost
- link-type
- hello-time
- portfast (including whether BPDU guard is enabled)
- guard root
- guard topology change
- BPDUs sent, received.

It also includes information specific to each MSTI:

- Port ID, priority, cost
- BPDU information from root (bridge ID, cost, and priority)
- BPDU information being sent on this port (Bridge ID, cost, priority)

- State transitions to reach this state.
- Topology changes to reach this state.
- Flush containment status for this MSTI.

This example shows the output of **show spanning-tree mst** errors, which produces information about interfaces that are configured for MSTP but where MSTP is not operational. Primarily this shows information about interfaces which do not exist:

```
# show spanning-tree mst a errors
Interface               Error
------------------------------
GigabitEthernet1/2/3/4   Interface does not exist.
```

This example shows the output of **show spanning-tree mst** configuration, which displays the VLAN ID to MSTI mapping table. It also displays the configuration digest which is included in the transmitted BPDUs—this must match the digest received from other bridges in the same MSTP region:

```
# show spanning-tree mst a configuration
Name          leo
Revision      2702
Config Digest 9D-14-5C-26-7D-BE-9F-B5-D8-93-44-1B-E3-BA-08-CE
Instance  Vlans mapped
--------  ------------------------------
0         1-9,11-19,21-29,31-39,41-4094
1         10,20,30,40
---------------------------------------
```

This example shows the output of **show spanning-tree mst**, which produces details on the BPDUs being output and received on a given local interface:

Note Several received packets can be stored in case of MSTP operating on a shared LAN.

```
# show spanning-tree mst a bpdu interface GigabitEthernet0/1/2/2 direction
 transmit
MSTI 0 (CIST):
 Root ID : 0004.9b78.0800
 Path Cost : 83
 Bridge ID : 0004.9b78.0800
 Port ID : 12
 Hello Time : 2
 ...
```

This example shows the output of **show spanning-tree mst**, which displays details about the topology changes that have occurred for each MSTI on each interface:

```
# show spanning-tree mst M topology-change flushes instance$
MSTI 1:

Interface     Last TC              Reason                           Count
------------  -------------------  -------------------------------- -----
Te0/0/0/1     04:16:05 Mar 16 2010  Role change: DSGN to ----           10
#
#
# show spanning-tree mst M topology-change flushes instance$
MSTI 0 (CIST):

Interface     Last TC              Reason                           Count
```

```
------------  -------------------  -------------------------------- -----
Te0/0/0/1     04:16:05 Mar 16 2010  Role change: DSGN to ----           10
#
```

Configuring MSTAG: Examples

This example shows MSTAG configuration for a single spanning-tree instance on a single interface:

```
config
interface GigabitEthernet0/0/0/0.1 l2transport
    encapsulation untagged
!
spanning-tree mstag example
    preempt delay for 60 seconds
    interface GigabitEthernet0/0/0/0.1
        name m1
        revision 10
        external-cost 0
        bridge-id 0.0.1
        port-id 1
        maximum age 40
        provider-bridge
        hello-time 1
        instance 101
              edge-mode
            vlans-id 101-110
            root-priority 0
            root-id 0.0.0
            cost 0
            priority 0
            port-priority 0
        !
    !
!
```

This example shows additional configuration for MSTAG Topology Change Propagation:

```
l2vpn
    xconnect group example
        p2p mstag-example
            interface GigabitEthernet0/0/0/0.1
            neighbor 123.123.123.1 pw-id 100
        !
    !
!
```

This example shows the output of **show spanning-tree mstag**:

```
# show spanning-tree mstag A
GigabitEthernet0/0/0/1
  Preempt delay is disabled.
  Name:            6161:6161:6161
  Revision:        0
  Max Age:         20
  Provider Bridge: no
  Bridge ID:       6161.6161.6161
  Port ID:         1
  External Cost:   0
  Hello Time:      2
  Active:          no
  BPDUs sent:      0
    MSTI 0 (CIST):
    VLAN IDs:         1-9,32-39,41-4094
    Role:             Designated
```

```
    Bridge Priority:  32768
    Port Priority:    128
    Cost:             0
    Root Bridge:      6161.6161.6161
    Root Priority:    32768
    Topology Changes: 123
  MSTI 2
    VLAN IDs:         10-31
    Role:             Designated
    Bridge Priority:  32768
    Port Priority:    128
    Cost:             0
    Root Bridge:      6161.6161.6161
    Root Priority:    32768
    Topology Changes: 123
  MSTI 10
 VLAN IDs:         40
    Role:             Root (Edge mode)
    Bridge Priority:  32768
    Port Priority:    128
    Cost:             200000000
    Root Bridge:      6161.6161.6161
    Root Priority:    61440
    Topology Changes: 0
```

This example shows the output of **show spanning-tree mstag bpdu interface**, which produces details on the BPDUs being output and received on a given local interface:

```
RP/0/RSP0/CPU0:router#show spanning-tree mstag foo bpdu interface GigabitEthernet 0/0/0/0
Transmitted:
  MSTI 0 (CIST):
ProtocolIdentifier: 0
ProtocolVersionIdentifier: 3
BPDUType: 2
CISTFlags: Top Change Ack  0
           Agreement       1
           Forwarding      1
           Learning        1
           Role            3
           Proposal        0
           Topology Change 0
CISTRootIdentifier: priority 8, MSTI 0, address 6969.6969.6969
CISTExternalPathCost: 0
CISTRegionalRootIdentifier: priority 8, MSTI 0, address 6969.6969.6969
CISTPortIdentifierPriority: 8
CISTPortIdentifierId: 1
MessageAge: 0
MaxAge: 20
HelloTime: 2
ForwardDelay: 15
Version1Length: 0
Version3Length: 80
FormatSelector: 0
Name: 6969:6969:6969
Revision: 0
MD5Digest: ac36177f 50283cd4 b83821d8 ab26de62
CISTInternalRootPathCost: 0
CISTBridgeIdentifier: priority 8, MSTI 0, address 6969.6969.6969
CISTRemainingHops: 20
  MSTI 1:
MSTIFlags: Master          0
           Agreement       1
           Forwarding      1
           Learning        1
           Role            3
```

```
          Proposal        0
          Topology Change 0
MSTIRegionalRootIdentifier: priority 8, MSTI 1, address 6969.6969.6969
MSTIInternalRootPathCost: 0
MSTIBridgePriority: 1
MSTIPortPriority: 8
MSTIRemainingHops: 20
```

This example shows the output of **show spanning-tree mstag topology-change flushes**, which displays details about the topology changes that have occurred for each interface:

```
#show spanning-tree mstag b topology-change flushes

MSTAG Protocol Instance b

Interface     Last TC              Reason                              Count
------------  -------------------  ---------------------------------- -----
Gi0/0/0/1     18:03:24 2009-07-14  Gi0/0/0/1.10 egress TCN          65535
Gi0/0/0/2     21:05:04 2009-07-15  Gi0/0/0/2.1234567890 ingress TCN     2
```

Configuring PVSTAG: Examples

This example shows PVSTAG configuration for a single VLAN on a single interface:

```
config
spanning-tree pvstag example
    preempt delay for 60 seconds
    interface GigabitEthernet0/0/0/0
        vlan 10
            root-priority 0
            root-id 0.0.0
            root-cost 0
            priority 0
            bridge-id 0.0.1
            port-priority 0
            port-id 1
            max age 40
            hello-time 1
        !
    !
!
```

This example shows the output of **show spanning-tree pvstag**:

```
# show spanning-tree pvstag interface GigabitEthernet0/0/0/1
GigabitEthernet0/0/0/1
  VLAN 10
    Preempt delay is disabled.
    Sub-interface:    GigabitEthernet0/0/0/1.20 (Up)
    Max Age:          20
    Root Priority:    0
    Root Bridge:      0000.0000.0000
    Cost:             0
    Bridge Priority:  32768
    Bridge ID:        6161.6161.6161
    Port Priority:    128
    Port ID:          1
    Hello Time:       2
    Active:           no
    BPDUs sent:       0
    Topology Changes: 123
    VLAN 20
```

Configuring PVRST: Example

This example shows a sample PVRST configuration.

```
(config)# spanning-tree pvrst stp1
(config-pvrst)# forward-delay 6
(config-pvrst)# interface GigabitEthernet 0/1/1/2 hello-time 2
(config-pvrst)# maximum age 35
(config-pvrst)# transmit hold-count 9
(config-pvrst)# vlan 666 priority 4096
(config-pvrst)# commit
```

Configuring MVRP-Lite: Examples

This example shows MVRP-lite configuration:

```
config
spanning-tree mst example
    mvrp static
        periodic transmit
        join-time 200
        leave-time 30
        leaveall-time 10
    !
!
```

This example shows the output of **show ethernet mvrp mad**:

```
RP/0/RSP0/CPU0:router# show ethernet mvrp mad interface GigabitEthernet 0/1/0/1
GigabitEthernet0/1/0/1
  Participant Type: Full; Point-to-Point: Yes
  Admin Control: Applicant Normal; Registrar Normal

  LeaveAll Passive (next in 5.92s); periodic disabled
  Leave in 25.70s; Join not running
  Last peer 0293.6926.9585; failed registrations: 0

VID   Applicant              Registrar
----  --------------------   ---------
   1  Very Anxious Observer  Leaving
 283  Quiet Passive          Empty
```

This example shows the output of **show ethernet mvrp status**:

```
RP/0/RSP0/CPU0:router# show ethernet mvrp status interface GigabitEthernet 0/1/0/1
GigabitEthernet0/1/0/1
  Statically declared:  1-512,768,980-1034
  Dynamically declared: 2048-3084
  Registered:           1-512
```

This example shows the output of **show ethernet mvrp statistics**:

```
RP/0/RSP0/CPU0:router# show ethernet mvrp statistics interface GigabitEthernet 0/1/0/1
GigabitEthernet0/1/0/1
  MVRPDUs TX:    1245
  MVRPDUs RX:       7
  Dropped TX:       0
  Dropped RX:      42
```

```
Invalid RX:      12
```

CHAPTER 9

EVPN Features

This chapter describes how to configure Layer 2 Ethernet VPN (EVPN) features on the router.

- EVPN Overview, on page 222
- EVPN Concepts, on page 223
- EVPN Operation, on page 224
- EVPN Route Types, on page 225
- EVPN Timers, on page 226
- Configure EVPN L2 Bridging Service, on page 226
- EVPN Single-Active Multi-Homing Mode, on page 228
- EVPN Software MAC Learning , on page 234
- EVPN Out of Service, on page 242
- CFM Support for EVPN, on page 246
- Control Word Support for ELAN, on page 246
- EVPN Multiple Services per Ethernet Segment, on page 247
- EVPN Single-Flow-Active Multihoming Load-Balancing Mode, on page 253
- EVPN Convergence Using NTP Synchronization, on page 258
- **EVPN MPLS Seamless Integration with VPLS** , on page 261
- Configure EVPN on the Existing VPLS Network, on page 262
- EVI Configuration Under L2VPN Bridge-Domain, on page 264
- Verify EVPN Configuration, on page 265
- Clear Forwarding Table, on page 269
- Hierarchical EVPN Access Pseudowire, on page 269
- EVPN Seamless Integration with VPWS, on page 271
- Network Convergence using Core Isolation Protection, on page 277
- Configurable Recovery Time for EVPN Core Isolation Group, on page 283
- Conditional Advertisement of Default-Originate, on page 290
- EVPN Single-Active Multihoming for Anycast Gateway IRB, on page 294
- EVPN Core Isolation Protection, on page 296
- EVPN Routing Policy, on page 299
- CFM on EVPN ELAN, on page 314
- EVPN Bridging and VPWS Services over BGP-LU Underlay, on page 322
- Support for DHCPv4 and DHCPv6 Client over BVI, on page 334

EVPN Overview

Ethernet VPN (EVPN) is a solution that provides Ethernet multipoint services over MPLS networks. EVPN operates in contrast to the existing Virtual Private LAN Service (VPLS) by enabling control-plane based MAC learning in the core. In EVPN, PEs participating in the EVPN instances learn customer MAC routes in control-plane using MP-BGP protocol. Control-plane MAC learning brings a number of benefits that allow EVPN to address the VPLS shortcomings, including support for multihoming with per-flow load balancing.

EVPN provides the solution for network operators for the following emerging needs in their network:

- Data center interconnect operation (DCI)
- Cloud and services virtualization
- Remove protocols and network simplification
- Integration of L2 and L3 services over the same VPN
- Flexible service and workload placement
- Multi-tenancy with L2 and L3 VPN
- Optimal forwarding and workload mobility
- Fast convergence
- Efficient bandwidth utilization

EVPN Benefits

The EVPN provides the following benefits:

- Integrated Services: Integrated L2 and L3 VPN services, L3VPN-like principles and operational experience for scalability and control, all-active multihoming and PE load-balancing using ECMP, and enables load balancing of traffic to and from CEs that are multihomed to multiple PEs.
- Network Efficiency: Eliminates flood and learn mechanism, fast-reroute, resiliency, and faster reconvergence when the link to dual-homed server fails, optimized Broadcast, Unknown-unicast, Multicast (BUM) traffic delivery.
- Service Flexibility: MPLS data plane encapsulation, support existing and new services types (E-LAN, E-Line), peer PE auto-discovery, and redundancy group auto-sensing.

EVPN Modes

The following EVPN modes are supported:

- Single-homing - Enables you to connect a customer edge (CE) device to one provider edge (PE) device.
- Multihoming - Enables you to connect a customer edge (CE) device to more than one provider edge (PE) device. Multihoming ensures redundant connectivity. The redundant PE device ensures that there is no traffic disruption when there is a network failure. Following are the types of multihoming:
  - Single-Active - In single-active mode only a single PE among a group of PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.

- All-Active - In all-active mode all the PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.

EVPN Restrictions

When paths of different technologies are resolved over ECMP, it results in *heterogeneous* ECMP, leading to severe network traffic issues. Don't use ECMP for any combination of the following technologies:

- LDP.
- BGP-LU, including services over BGP-LU loopback peering or recursive services at Level-3
- VPNv4.
- 6PE and 6VPE.
- EVPN.
- Recursive static routing.

EVPN Concepts

To implement EVPN features, you need to understand the following concepts:

- Ethernet Segment (ES): An Ethernet segment is a set of Ethernet links that connects a multihomed device. If a multi-homed device or network is connected to two or more PEs through a set of Ethernet links, then that set of links is referred to as an Ethernet segment. The Ethernet segment route is also referred to as Route Type 4. This route is used for designated forwarder (DF) election for BUM traffic.
- Ethernet Segment Identifier (ESI): Ethernet segments are assigned a unique non-zero identifier, which is called an Ethernet Segment Identifier (ESI). ESI represents each Ethernet segment uniquely across the network.
- EVI: The EVPN instance (EVI) is represented by the virtual network identifier (VNI). An EVI represents a VPN on a PE router. It serves the same role of an IP VPN Routing and Forwarding (VRF), and EVIs are assigned import/export Route Targets (RTs). Depending on the service multiplexing behaviors at the User to Network Interface (UNI), all traffic on a port (all-to-one bundling), or traffic on a VLAN (one-to-one mapping), or traffic on a list/range of VLANs (selective bundling) can be mapped to a Bridge Domain (BD). This BD is then associated to an EVI for forwarding towards the MPLS core.
- EAD/ES: Ethernet Auto Discovery Route per ES is also referred to as Route Type 1. This route is used to converge the traffic faster during access failure scenarios. This route has Ethernet Tag of 0xFFFFFFFF.
- EAD/EVI: Ethernet Auto Discovery Route per EVI is also referred to as Route Type 1. This route is used for aliasing and load balancing when the traffic only hashes to one of the switches. This route cannot have Ethernet tag value of 0xFFFFFFFF to differentiate it from the EAD/ES route.
- Aliasing: It is used for load balancing the traffic to all the connected switches for a given Ethernet segment using the Route Type 1 EAD/EVI route. This is done irrespective of the switch where the hosts are actually learned.
- Mass Withdrawal: It is used for fast convergence during the access failure scenarios using the Route Type 1 EAD/ES route.

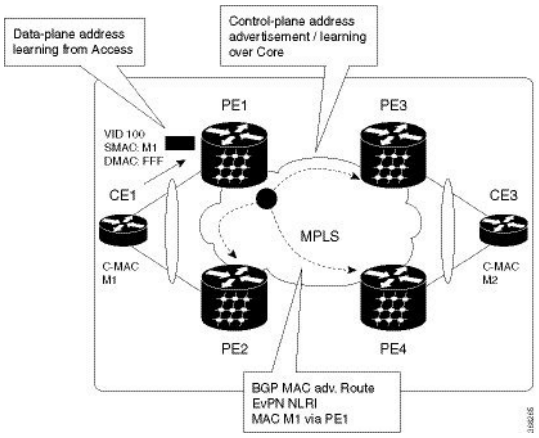
- DF Election: It is used to prevent forwarding of the loops. Only a single router is allowed to decapsulate and forward the traffic for a given Ethernet Segment.

EVPN Operation

At startup, PEs exchange EVPN routes in order to advertise the following:

- **VPN membership**: The PE discovers all remote PE members of a given EVI. In the case of a multicast ingress replication model, this information is used to build the PEs flood list associated with an EVI. BUM labels and unicast labels are exchanged when MAC addresses are learned.
- **Ethernet segment reachability**: In multihoming scenarios, the PE auto-discovers remote PE and their corresponding redundancy mode (all-active or single-active). In case of segment failures, PEs withdraw the routes used at this stage in order to trigger fast convergence by signaling a MAC mass withdrawal on remote PEs.
- **Redundancy Group membership**: PEs connected to the same Ethernet segment (multihoming) automatically discover each other and elect a Designated Forwarder (DF) that is responsible for forwarding Broadcast, Unknown unicast and Multicast (BUM) traffic for a given EVI.

Figure 39: EVPN Operation



EVPN can operate in single-homing or dual-homing mode. Consider single-homing scenario, when EVPN is enabled on PE, Route Type 3 is advertised where each PE discovers all other member PEs for a given EVPN instance. When an unknown unicast (or BUM) MAC is received on the PE, it is advertised as EVPN Route Type 2 to other PEs. MAC routes are advertised to the other PEs using EVPN Route Type 2. In multihoming scenarios, Route Types 1, 3, and 4 are advertised to discover other PEs and their redundancy modes (single-active or all-active). Use of Route Type 1 is to auto-discover other PE which hosts the same CE. The other use of this route type is to fast route unicast traffic away from a broken link between CE and PE. Route Type 4 is used for electing designated forwarder. For instance, consider the topology when customer traffic arrives at the PE, EVPN MAC advertisement routes distribute reachability information over the core for each customer MAC address learned on local Ethernet segments. Each EVPN MAC route announces the customer MAC address and the Ethernet segment associated with the port where the MAC was learned from and its associated MPLS label. This EVPN MPLS label is used later by remote PEs when sending traffic destined to the advertised MAC address.

Behavior Change due to ESI Label Assignment

To adhere to RFC 7432 recommendations, the encoding or decoding of MPLS label is modified for extended community. Earlier, the lower 20 bits of extended community were used to encode the split-horizon group (SHG) label. Now, the SHG label encoding uses from higher 20 bits of extended community.

According to this change, routers in same ethernet-segment running old and new software release versions decodes extended community differently. This change causes inconsistent SHG labels on peering EVPN PE routers. Almost always, the router drops BUM packets with incorrect SHG label. However, in certain conditions, it may cause remote PE to accept such packets and forward to CE potentially causing a loop. One such instance is when label incorrectly read as NULL.

To overcome this problem, Cisco recommends you to:

- Minimize the time both PEs are running different software release versions.
- Before upgrading to a new release, isolate the upgraded node and shutdown the corresponding AC bundle.
- After upgrading both the PEs to the same release, you can bring both into service.

Similar recommendations are applicable to peering PEs with different vendors with SHG label assignment that does not adhere to RFC 7432.

EVPN Route Types

The EVPN network layer reachability information (NLRI) provides different route types.

Table 19: EVPN Route Types

Route Type	Name	Usage
1	Ethernet Auto-Discovery (AD) Route	Few routes are sent per ES, carries the list of EVIs that belong to ES
2	MAC/IP Advertisement Route	Advertise MAC, address reachability, advertise IP/MAC binding
3	Inclusive Multicast Ethernet Tag Route	Multicast Tunnel End point discovery
4	Ethernet Segment Route	Redundancy group discovery, DF election
5	IP Prefix Route	Advertise IP prefixes.

Route Type 1: Ethernet Auto-Discovery (AD) Route

The Ethernet Auto-Discovery (AD) routes are advertised on per EVI and per ESI basis. These routes are sent per ES. They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed. This route type is used for mass withdrawal of MAC addresses and aliasing for load balancing.

Route Type 2: MAC/IP Advertisement Route

These routes are per-VLAN routes, so only PEs that are part of a VNI require these routes. The host's IP and MAC addresses are advertised to the peers within NRLI. The control plane learning of MAC addresses reduces unknown unicast flooding.

Route Type 3: Inclusive Multicast Ethernet Tag Route

This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.

Route Type 4: Ethernet Segment Route

Ethernet segment routes enable to connect a CE device to two or PE devices. ES route enables the discovery of connected PE devices that are connected to the same Ethernet segment.

Route Type 5: IP Prefix Route

The IP prefixes are advertised independently of the MAC-advertised routes. With EVPN IRB, host route /32 is advertised using RT-2 and subnet /24 is advertised using RT-5.

Note With EVPN IRB, host route /32 are advertised using RT-2 and subnet /24 are advertised using RT-5.

EVPN Timers

The following table shows various EVPN timers:

Configure EVPN L2 Bridging Service

Perform the following steps to configure EVPN L2 bridging service.

Note Always ensure to change the label mode from per-prefix to per-VRF label mode. Since L2FIB and VPNv4 route (labels) shares the same resource, BVI ping fails when you exhaust the resources.

Note Traffic to directly connected neighbor on EVPN or VPLS bridge won't work in the following scenarios:

- If neighbor doesn't advertise MPLS explicit null.
- If imposition node has a mix of implicit-null and labeled paths in ECMP or LFA deployment.

Note A device can contain up to 128K MAC address entries. A bridge domain on a device can contain up to 64K MAC address entries.

Note Flooding disable isn't supported on EVPN bridge domains.

```
/* Configure address family session in BGP */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 209.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn

/* Configure EVI and define the corresponding BGP route targets */
```

Note EVI route target used for multicast EVPN supports only extcomm type sub-type 0xA for EVI route target, the two-octet Autonomous System (AS) specific Extended Community. This means that when using a 4-byte AS number for BGP, you must additionally configure BGP import and export route targets under the EVPN configuration.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 6005
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# rd 200:50
Router(config-evpn-evi-bgp)# route-target import 100:6005
Router(config-evpn-evi-bgp)# route-target export 100:6005
Router(config-evpn-evi-bgp)# exit
Router(config-evpn-evi)# advertise-mac

/* Configure a bridge domain */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 1
Router(config-l2vpn-bg)# bridge-domain 1-1
Router(config-l2vpn-bg-bd)# interface GigabitEthernet
Router(config-l2vpn-bg-bd-ac)# evi 6005
Router(config-l2vpn-bg-bd-ac-evi)# commit
Router(config-l2vpnbg-bd-ac-evi)# exit
```

Running Configuration

```
router bgp 200 bgp
 router-id 209.165.200.227
```

```
 address-family l2vpn evpn
 neighbor 10.10.10.10
  remote-as 200 description MPLS-FACING-PEER
  updatesource Loopback0
  addressfamily l2vpn evpn
!

configure
evpn
 evi 6005
  bgp
   rd 200:50
   route-target import 100:6005
   route-target export 100:6005
!
  advertise-mac


configure
l2vpn
 bridge group 1
  bridge-domain 1-1
   interface GigabitEthernet

    evi 6005
!
```

EVPN Single-Active Multi-Homing Mode

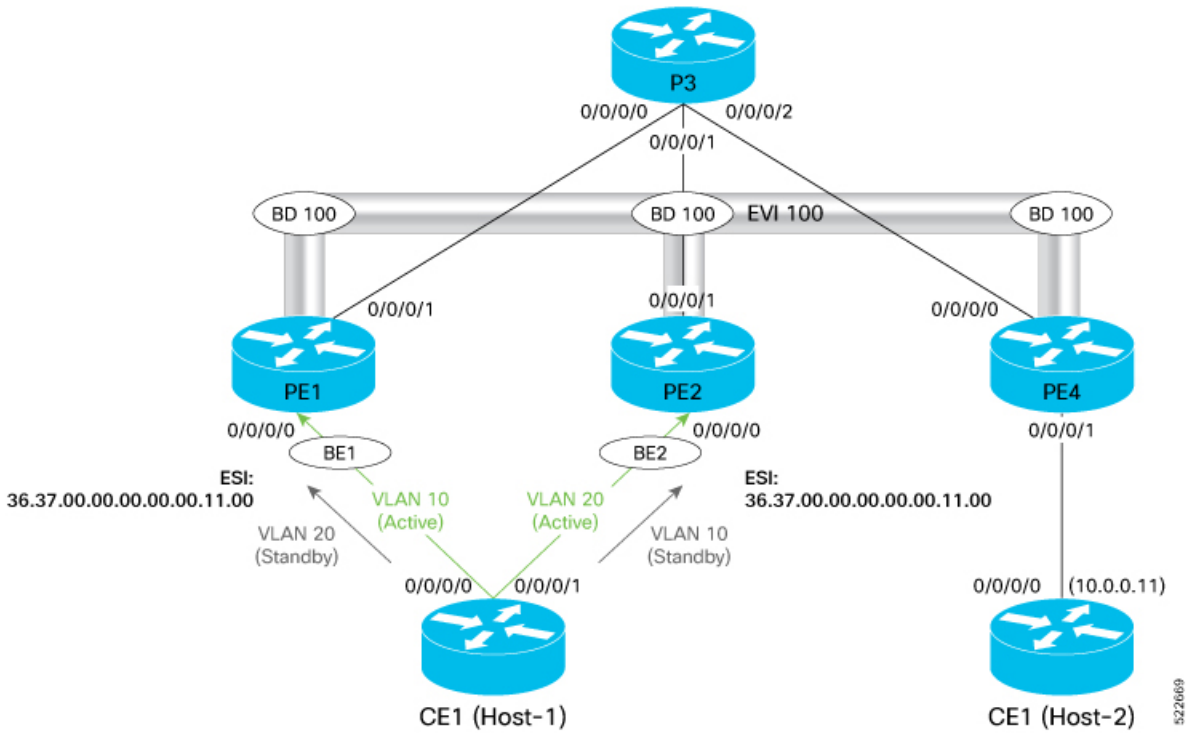
In single-active multihoming mode, only a single edge (PE) Router among a group of PE Routers attached to a host is allowed to send and recieve traffic on a given VLAN.

The single-active mode offers redundant connectivity for a VLAN on a single link at a time with failover to the second link in case the active link fails. The single-active mode directs the traffic to a single uplink. This mode is useful for network scenarios where policing, metering, and billing are required.

In Single-Active mode, Cisco IOS XR sends a topology change notification on the Ethernet segment links when a service carving update occurs, so that CEs flush their MAC tables and redirect traffic to the new DF-Elected PE.

Topology

Let's understand how the single-active mode works with this sample topology.



In this topology,

- The CE Router is multihomed to PE1 and PE2. Only one active uplink is allowed to send and receive traffic at any given time.
- In this mode, each link towards PE is in a unique ethernet bundle interface. In this example, BE1 is the ethernet bundle interface connecting CE1 and PE1. BE2 is the ethernet bundle interface connecting CE1 and PE2.
- As both the links are in a separate ethernet bundle interface, CE1 floods traffic at first to both the PE devices, but only the PE that is the Designated Forwarder (DF) forwards the traffic.
- In this mode, the uplinks to PE1 and PE2 are individual links and by default, the host chooses the DF uplink for forwarding for a given VLAN.

Configure EVPN Single-Active Multi-Homing

Perform the following tasks to configure EVPN single-active multi-homing:

- Configure Ethernet bundles on CE1 for multi-homing.
- Configure EVPN based single-active multi-homing.

Note Sub-interface shutdown is not supported in Single-Active load-balancing mode

Configure Ethernet bundles on CE1 for Multihoming:

```
Router#configure
Router(config)#interface Bundle-Ether1
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface Bundle-Ether2
Router(config-if)# no shutdown
Router(config)#exit
Router(config)#interface HundredGigE0/0/0/0
Router(config-if)#bundle id 1 mode active
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface HundredGigE0/0/0/1
Router(config-if)#bundle id 2 mode active
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface HundredGigE0/0/0/2
Router(config-if)#exit
Router(config)#interface HundredGigE0/0/0/3
Router(config-if)#no shutdown
Router(config-if)#commit
Router(config-if)#exit
Router(config)#interface Bundle-Ether1.10 l2transport
Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#rewrite ingress tag pop 1 symmetric
Router(config-subif)#commit
Router(config-subif)#exit
Router(config)#interface Bundle-Ether2.10 l2transport
Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#rewrite ingress tag pop 1 symmetric
Router(config-subif)#commit
Router(config-subif)#root
Router(config)#interface BVI10
Router(config-if)#ipv4 address 10.0.0.10 255.255.255.0
Router(config-if)exit
Router(config)#interface BVI10
Router(config-if)#ipv4 address 10.0.0.10 255.255.255.0
Router(config-if)#exit
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd-10
Router(config-l2vpn-bg-bd)#interface Bundle-Ether1.10
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#interface Bundle-Ether2.10
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#routed interface BVI10
Router(config-l2vpn-bg-bd-bvi)#commit
```

Configure EVPN based single-active multi-homing on PE Routers.

PE1 Configuration:

```
Router#configure
Router(config)#interface Bundle-Ether1
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface Bundle-Ether2
Router(config-if)#no shutdown
Router(config)#exit
Router(config)#interface Bundle-Ether1.10 l2transport
Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#rewrite ingress tag pop 1 symmetric
```

```
Router(config-subif)#exit
Router(config)#evpn
Router(config-evpn)#evi 100
Router(config-evpn-instance)#advertise-mac
Router(config-evpn-instance-mac)#exit
Router(config-evpn-instance)#exit
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 36.37.00.00.00.00.00.11.00
Router(config-evpn-ac-es)#load-balancing-mode single-active
Router(config-evpn-ac-es)#root
Router(config)#l2vpn
Router(config-l2vpn)#bridge group 100
Router(config-l2vpn-bg)#bridge-domain 100
Router(config-l2vpn-bg-bd)#interface Bundle-Ether2.10
Router(config-l2vpn-bg-bd)#evi 100
Router(config-l2vpn-bg-bd-evi)#commit
```

PE2 Configuration:

```
Router#configure
Router(config)#interface Bundle-Ether1
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface Bundle-Ether2
Router(config-if)#no shutdown
Router(config)#exit
Router(config)#interface Bundle-Ether2.10 l2transport
Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#rewrite ingress tag pop 1 symmetric
Router(config-subif)#exit
Router(config)#evpn
Router(config-evpn)#evi 100
Router(config-evpn-instance)#advertise-mac
Router(config-evpn-instance-mac)#exit
Router(config-evpn-instance)#exit
Router(config-evpn)#interface Bundle-Ether2
Router(config-evpn-ac)#ethernet-segment 0 36.37.00.00.00.00.00.11.00
Router(config-evpn-ac-es)#load-balancing-mode single-active
Router(config-evpn-ac-es)#root
Router(config)#l2vpn
Router(config-l2vpn)#bridge group 100
Router(config-l2vpn-bg)#bridge-domain 100
Router(config-l2vpn-bg-bd)#interface Bundle-Ether2.10
Router(config-l2vpn-bg-bd)#evi 100
Router(config-l2vpn-bg-bd-evi)#exit
```

Running Configuration

This section shows the single-active running configuration.

```
/* CE1 Configuration */
interface Bundle-Ether1
!
interface Bundle-Ether1.10 l2transport
 encapsulation dot1q 10
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether2
!
interface Bundle-Ether2.10 l2transport
```

```
 encapsulation dot1q 10
 rewrite ingress tag pop 1 symmetric
!
interface Loopback0
 ipv4 address 200.0.0.7 255.255.255.255
!
interface MgmtEth0/RSP0/CPU0/0
 ipv4 address dhcp
!
interface BVI10
 description "Host-1 IP"
 ipv4 address 10.0.0.10 255.255.255.0
!
interface HundredGigE0/0/0/0
 bundle id 1 mode active
!
interface HundredGigE0/0/0/1
 description "Link to Leaf-2"
 bundle id 2 mode active
!
l2vpn
 bridge group bg1
  bridge-domain bd-10
   interface Bundle-Ether1.10
   !
   interface Bundle-Ether2.10
   !
   routed interface BVI10
   !
  !


/* PE1 Configuration */
evpn
 evi 100
  advertise-mac
  !
 !
 interface Bundle-Ether1
  ethernet-segment
  identifier type 0 36.37.00.00.00.00.00.11.00
   load-balancing-mode single-active
  !
 !
!
l2vpn
 bridge group 100
  bridge-domain 100
   interface Bundle-Ether1.10
   !
   evi 100
   !
  !
 !
!
commit
root
exit


/* PE2 Configuration */
evpn
 evi 100
  advertise-mac
  !
```

```
 !
 interface Bundle-Ether2
  ethernet-segment
   identifier type 0 36.37.00.00.00.00.00.11.00
   load-balancing-mode single-active
  !
 !
!
l2vpn
 bridge group 100
  bridge-domain 100
   interface Bundle-Ether2.10
   !
   evi 100
   !
  !
 !
!
```

Verification

The following output shows that the EVPN single-active mode is enabled:

```
Router#show evpn ethernet-segment detail

Legend:
  B   - No Forwarders EVPN-enabled,
  C   - Backbone Source MAC missing (PBB-EVPN),
  RT  - ES-Import Route Target missing,
  E   - ESI missing,
  H   - Interface handle missing,
  I   - Name (Interface or Virtual Access) missing,
  M   - Interface in Down state,
  O   - BGP End of Download missing,
  P   - Interface already Access Protected,
  Pf  - Interface forced single-homed,
  R   - BGP RID not received,
  S   - Interface in redundancy standby state,
  X   - ESI-extracted MAC Conflict
  SHG - No local split-horizon-group label allocated

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0036.3700.0000.0000.1100 BE1                               10.1.1.1
                                                           10.2.2.2
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether1
     Interface MAC  : 0008.3302.3208
     IfHandle       : 0x02000160
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 36.3700.0000.0000.1100
  ES Import RT      : 3637.0000.0000 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, Single-active
     Configured     : Single-active (AApS)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       :
```

```
   Mobility-Flush : Count 0, Skip 0, Last n/a
  Peering Details   : 2 Nexthops
     10.1.1.1 [MOD:P:00]
     10.2.2.2 [MOD:P:00]
  Service Carving Results:
     Forwarders     : 1
     Elected        : 1
     Not Elected    : 0
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : 24007
  Remote SHG labels : 1
              24007 : nexthop 10.2.2.2
  Access signal mode: Bundle OOS (Default)
```

The following output shows that Bundle-Ether1 is up:

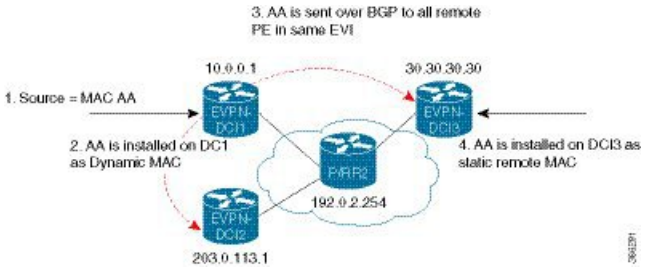
```
Router:PE1#show bundle bundle-ether 1

Bundle-Ether1
  Status:                                  Up
  Local links <active/standby/configured>:   1 / 0 / 1
  Local bandwidth <effective/available>:     100000000 (100000000) kbps
  MAC address (source):                    0008.3532.0137 (Chassis pool)
  Inter-chassis link:                      No
  Minimum active links / bandwidth:        1 / 1 kbps
  Maximum active links:                    64
  Wait while timer:                        2000 ms
  Load balancing:
    Link order signaling:                  Not configured
    Hash type:                             Default
    Locality threshold:                    None
  LACP:                                    Operational
    Flap suppression timer:                Off
    Cisco extensions:                      Disabled
    Non-revertive:                         Disabled
  mLACP:                                   Not configured
  IPv4 BFD:                                Not configured
  IPv6 BFD:                                Not configured

  Port                Device           State        Port ID         B/W, kbps
  ------------------  ---------------  -----------  --------------  ----------
  Hu0/0/0/0           Local            Active       0x8000, 0x0001   100000000
      Link is Active
```

EVPN Software MAC Learning

The MAC addresses learned on one device needs to be learned or distributed on the other devices in a VLAN. EVPN Software MAC Learning feature enables the distribution of the MAC addresses learned on one device to the other devices connected to a network. The MAC addresses are learnt from the remote devices using BGP.

Note A device can contain up to 128K MAC address entries. A bridge domain on a device can contain up to 64K MAC address entries.

Figure 40: EVPN Software MAC Learning



The above figure illustrates the process of software MAC learning. The following are the steps involved in the process:

1. Traffic comes in on one port in the bridge domain.
2. The source MAC address (AA) is learnt on the PE and is stored as a dynamic MAC entry.
3. The MAC address (AA) is converted into a type-2 BGP route and is sent over BGP to all the remote PEs in the same EVI.
4. The MAC address (AA) is updated on the PE as a remote MAC address.

Configure EVPN Software MAC Learning

The following section describes how you can configure EVPN Software MAC Learning:

Note From Release 7.4.1 Control word is enabled by default. If the **control-word-disable** command is not configured, ensure to configure it under EVPN or EVI configuration mode before an upgrade to avoid inconsistent behaviour with routers running before Release 7.4.2.

If you want to enable **control-word** command for EVPN Bridging feature, then you must configure it only when both the endpoints run Release 7.4.1 or later.

If you want to disable control word command, use **control-word-disable** before Release 7.8.1, it needed a router to reload to take effect.

Note The router does not support flow-aware transport (FAT) pseudowire.

```
/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_SH
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
```

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface TenGigE
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface BundleEther 20.2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# storm-control broadcast pps 10000 ← Enabling
 storm-control is optional
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-evi)# commit

/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 200
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 209.165.200.227
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp-nbr)# description MPLSFACINGPEER
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
```

Supported Modes for EVPN Software MAC Learning

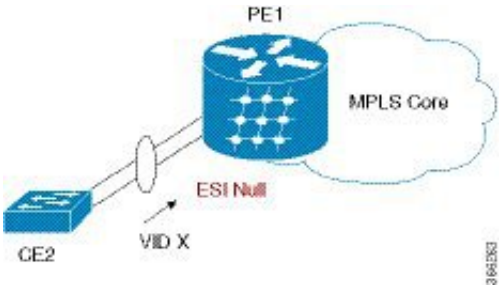
The following are the modes in which EVPN Software MAC Learning is supported:

- Single Home Device (SHD) or Single Home Network (SHN)
- Dual Home Device (DHD)—All Active Load Balancing

Single Home Device or Single Home Network Mode

The following section describes how you can configure EVPN Software MAC Learning feature in single home device or single home network (SHD/SHN) mode:

Figure 41: Single Home Device or Single Home Network Mode



In the above figure, the PE (PE1) is attached to Ethernet Segment using bundle or physical interfaces. Null Ethernet Segment Identifier (ESI) is used for SHD/SHN.

Configure EVPN in Single Home Device or Single Home Network Mode

This section describes how you can configure EVPN Software MAC Learning feature in single home device or single home network mode.

```
/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
```

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface Bundle-Ether1.2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001


/* Configure advertisement of MAC routes. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac


/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 09.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn
```

Running Configuration

```
l2vpn
bridge group EVPN_ALL_ACTIVE
 bridge-domain EVPN_2001
  interface BundleEther1.2001
  evi 2001
!
evpn
 evi 2001
  advertise-mac
!
router bgp 200 bgp
 router-id 40.40.40.40
 address-family l2vpn evpn
 neighbor 10.10.10.10
  remote-as 200 description MPLS-FACING-PEER
  updatesource Loopback0
  addressfamily l2vpn evpn
```

Verification

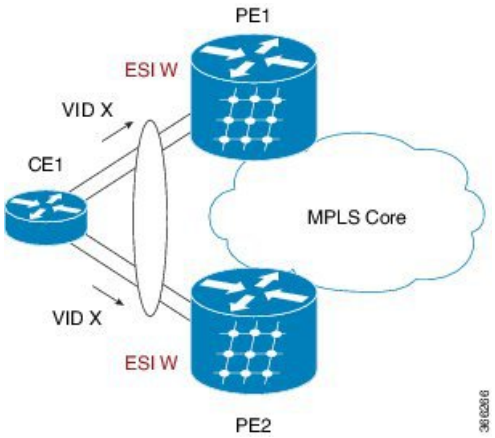
Verify EVPN in single home devices.

```
RP/0/RSP0/CPU0:router# show evpn ethernet-segment interface Te0/4/0/10 detail

Ethernet Segment Id    Interface   Nexthops
---------------------  ----------  ----------
N/A                    Te0/4/0/10  20.20.20.20
……
 Topology :
 Operational : SH
 Configured : Single-active (AApS) (default)
```

Dual Home Device—All-Active Load Balancing Mode

The following section describes how you can configure EVPN Software MAC Learning feature in dual home device (DHD) in all-active load balancing mode:

Figure 42: Dual Home Device —All-Active Load Balancing Mode



All-active load-balancing is known as Active/Active per Flow (AApF). In the above figure, identical Ethernet Segment Identifier is used on both EVPN PEs. PEs are attached to Ethernet Segment using bundle interfaces. In the CE, single bundles are configured towards two EVPN PEs. In this mode, the MAC address that is learnt is stored on both PE1 and PE2. Both PE1 and PE2 can forward the traffic within the same EVI.

Configure EVPN Software MAC Learning in Dual Home Device—All-Active Mode

This section describes how you can configure EVPN Software MAC Learning feature in dual home device—all-active mode:

```
/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001


/* Configure advertisement of MAC routes. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
RP/0/RSP0/CPU0:router(config-evpn-evi)# exit
RP/0/RSP0/CPU0:router(config-evpn)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# identifier type 0 01.11.00.00.00.00.00.00.01


/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 209.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
```

```
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLS-FACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn


/* Configure Link Aggregation Control Protocol (LACP) bundle. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-if)# lacp switchover suppress-flaps 300
RP/0/RSP0/CPU0:router(config-if)# exit


/* Configure VLAN Header Rewrite.*/

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1 l2transport
RP/0/RSP0/CPU0:router(config-if)# encapsulation dot1q 10
RP/0/RSP0/CPU0:router(config-if)# rewrite ingress tag pop 1 symmetric
```

Note Configure the same mlacp system priority <id> for both the dual homed PE routers to enable all-active load balancing.

Running Configuration

```
l2vpn
bridge group EVPN_ALL_ACTIVE
 bridge-domain EVPN_2001
 interface Bundle-Ether1
 !
 evi 2001
 !
!
evpn
 evi 2001
 !
 advertise-mac
 !
 interface Bundle-Ether1
  ethernet-segment
  identifier type 0 01.11.00.00.00.00.00.00.01
  !
 !
router bgp 200
bgp router-id  209.165.200.227
address-family l2vpn evpn
!
neighbor 10.10.10.10
 remote-as 200
 description MPLS-FACING-PEER
 update-source Loopback0
 address-family l2vpn evpn
!
interface Bundle-Ether1
lacp switchover suppress-flaps 300
load-interval 30
!
interface Bundle-Ether1 l2transport
```

```
 encapsulation dot1aq 2001
 rewrite ingress tag pop 1 symmetric
!
```

Verification

Verify EVPN in dual home devices in All-Active mode.

Note With the EVPN IRB, the supported label mode is per-VRF.

```
RP/0/RSP0/CPU0:router# show evpn ethernet-segment interface Bundle-Ether 1 carvin$

Ethernet Segment Id        Interface  Nexthops
-------- ----------        --------   --------
0100.211b.fce5.df00.0b00   BE1       10.10.10.10
 209.165.201.1
Topology :
 Operational : MHN
 Configured : All-active (AApF) (default)
 Primary Services : Auto-selection
 Secondary Services: Auto-selection
 Service Carving Results:
 Forwarders : 4003
 Elected : 2002
 EVI E : 2000, 2002, 36002, 36004, 36006, 36008
 ........
 Not Elected : 2001
 EVI NE : 2001, 36001, 36003, 36005, 36007, 36009

 MAC Flushing mode : Invalid

Peering timer : 3 sec [not running]
 Recovery timer : 30 sec [not running]
 Local SHG label : 34251
 Remote SHG labels : 1
  38216 : nexthop 209.165.201.1
```

Verify EVPN Software MAC Learning

Verify the packet drop statistics.

Note Disable CW configuration if any in EVPN peer nodes, as CW is not supported in EVPN Bridging.

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain bd-name EVPN_2001 details

Bridge group: EVPN_ALL_ACTIVE, bridge-domain: EVPN_2001, id: 1110,
state: up, ShgId: 0, MSTi: 0
 List of EVPNs:
 EVPN, state: up
 evi: 2001
 XC ID 0x80000458
 Statistics:
 packets: received 28907734874 (unicast 9697466652), sent
76882059953
 bytes: received 5550285095808 (unicast 1861913597184), sent
```

```
14799781851396
 MAC move: 0
 List of ACs:
 AC: TenGigE0/0/0/1, state is up
 Type VLAN; Num Ranges: 1
...
 Statistics:
 packets: received 0 (multicast 0, broadcast 0, unknown
unicast 0, unicast 0), sent 45573594908
 bytes: received 0 (multicast 0, broadcast 0, unknown unicast
0, unicast 0), sent 8750130222336
 MAC move: 0
 ........
```

Verify the EVPN EVI information with the VPN-ID and MAC address filter.

```
RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 neighbor

Neighbor IP    vpn-id
-----------   --------
209.165.200.225   2001
209.165.201.30   2001
```

Verify the BGP L2VPN EVPN summary.

```
RP/0/RSP0/CPU0:router# show bgp l2vpn evpn summary
...
Neighbor        Spk   AS     MsgRcvd  MsgSent  TblVer    InQ  OutQ  Up/Down  St/PfxRcd
209.165.200.225  0    200     216739  229871   200781341  0    0     3d00h   348032
209.165.201.30   0    200     6462962 4208831  200781341  10   0     2d22h   35750
```

Verify the MAC updates to the L2FIB table in a line card.

```
RP/0/RSP0/CPU0:router# show l2vpn mac mac all location 0/6/CPU0

 Topo ID     Producer     Next Hop(s)      Mac Address       IP Address
 --------    --------     -----------      --------------    ----------
     1112    0/6/CPU0     Te0/6/0/1.36001  00a3.0001.0001
```

Verify the MAC updates to the L2FIB table in a route switch processor (RSP).

```
RP/0/RSP0/CPU0:router# show l2vpn mac mac all location 0/6/CPU0

 Topo ID     Producer     Next Hop(s)      Mac Address       IP Address
 --------    --------     -----------      --------------    ----------
     1112    0/6/CPU0     Te0/6/0/1.36001  00a3.0001.0001
```

Verify the summary information for the MAC address.

```
RP/0/RP0/CPU0:router# show l2vpn forwarding bridge-domain EVPN_ALL_ACTIVE:EVPN_2001
mac-address location 0/6/CPU0


    Mac Address    Type    Learned from/Filtered on    LC learned Resync Age/Last Change
Mapped to
    -------------- ------- -------------------------- ---------- ----------------------
--------------
    00a3.0001.0001 dynamic Te0/6/0/1.36001            N/A        01 Sep 10:09:17
N/A
    0010.0400.0003 dynamic Te0/0/0/10/0.1             N/A        Remotely Aged
```

```
N/A
    2000.3000.4000 static  Te0/0/0/10/0.2              N/A        N/A
N/A
```

Verify the EVPN EVI information with the VPN-ID and MAC address filter.

```
RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 mac
VPN-ID     Encap      MAC address    IP address     Nexthop
  Label
---------- ---------- -------------- ------------- ---------------------------------------
 --------
  2001               00a9.2002.0001 ::             10.10.10.10
  34226    <-- Remote MAC
  2001               00a9.2002.0001 ::             209.165.201.30
  34202

  2001               00a3.0001.0001 20.1.5.55      TenGigE0/6/0/1.36001
  34203    <-- Local MAC


RP/0/RSP0/CPU0:router# RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 mac 00a9.2002.0001
 detail

EVI     MAC address      IP address  Nexthop      Label
----    --------------   ----------  -------      -----
2001    00a9.2002.0001   ::          10.10.10.10  34226

2001    00a9.2002.0001   ::          209.165.201.30  34202

 Ethernet Tag : 0
 Multi-paths Resolved : True <--- aliasing to two remote PE with All-Active load balancing

 Static : No
 Local Ethernet Segment : N/A
 Remote Ethernet Segment : 0100.211b.fce5.df00.0b00
 Local Sequence Number : N/A
 Remote Sequence Number : 0
 Local Encapsulation : N/A
 Remote Encapsulation : MPLS
```

Verify the BGP routes associated with EVPN with bridge-domain filter.

```
RP/0/RSP0/CPU0:router# show bgp l2vpn evpn bridge-domain EVPN_2001 route-type 2

*> [2][0][48][00bb.2001.0001][0]/104
                        0.0.0.0           0 i <------ locally learnt MAC
*>i[2][0][48][00a9.2002.00be][0]/104
                        10.10.10.10 100     0 i <----- remotely learnt MAC
* i 209.165.201.30 100 0 i
```

EVPN Out of Service

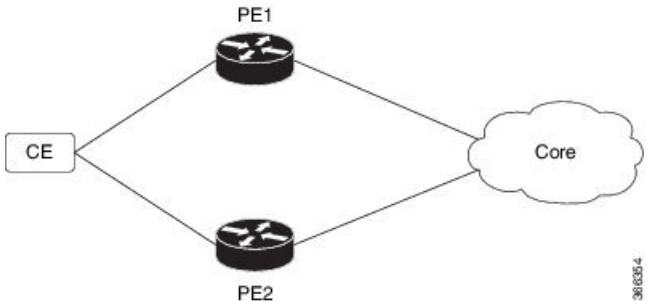
The EVPN Out of Service feature enables you to control the state of bundle interfaces that are part of an Ethernet segment that have Link Aggregation Control protocol (LACP) configured. This feature enables you to put a node out of service (OOS) without having to manually shutdown all the bundles on their provider edge (PE).

Use the **cost-out** command to bring down all the bundle interfaces belonging to an Ethernet VPN (EVPN) Ethernet segment on a node. The Ethernet A-D Ethernet Segment (ES-EAD) routes are withdrawn before shutting down the bundles. The PE signals to the connected customer edge (CE) device to bring down the corresponding bundle member. This steers away traffic from this PE node without traffic disruption. The traffic that is bound for the Ethernet segment from the CE is directed to the peer PE in a multi-homing environment.

Note EVPN cost-out is supported only on manually configured ESIs.

In the following topology, the CE is connected to PE1 and PE2. When you configure the **cost-out** command on PE1, all the bundle interfaces on the Ethernet segment are brought down. Also, the corresponding bundle member is brought down on the CE. Hence, the traffic for this Ethernet segment is now sent to PE2 from the CE.

Figure 43: EVPN Out of Service



To bring up the node into service, use **no cost-out** command. This brings up all the bundle interfaces belonging to EVPN Ethernet segment on the PE and the corresponding bundle members on the CE.

When the node is in cost-out state, adding a new bundle Ethernet segment brings that bundle down. Similarly, removing the bundle Ethernet segment brings that bundle up.

Use **startup-cost-in** command to bring up the node into service after the specified time on reload. The node will cost-out when EVPN is initialized and remain cost-out until the set time. If you execute **evpn no startup-cost-in** command while timer is running, the timer stops and node is cost-in.

The 'cost-out' configuration always takes precedence over the 'startup-cost-in' timer. So, if you reload with both the configurations, cost-out state is controlled by the 'cost-out' configuration and the timer is not relevant. Similarly, if you reload with the startup timer, and configure 'cost-out' while timer is running, the timer is stopped and OOS state is controlled only by the 'cost-out' configuration.

If you do a proc restart while the startup-cost-in timer is running, the node remains in cost-out state and the timer restarts.

Configure EVPN Out of Service

This section describes how you can configure EVPN Out of Service.

```
/* Configuring node cost-out on a PE */

Router# configure
Router(config)# evpn
```

```
Router(config-evpn)# cost-out
Router(config-evpn)commit

/* Bringing up the node into service */

Router# configure
Router(config)# evpn
Router(config-evpn)# no cost-out
Router(config-evpn)commit

/* Configuring the timer to bring up the node into service after the specified time on
reload */

Router# configure
Router(config)# evpn
Router(config-evpn)# startup-cost-in 6000
Router(config-evpn)commit
```

Running Configuration

```
configure
evpn
 cost-out
!

configure
evpn
 startup-cost-in 6000
!
```

Verification

Verify the EVPN Out of Service configuration.

```
/* Verify the node cost-out configuration */

Router# show evpn summary
Fri Apr  7 07:45:22.311 IST
Global Information
----------------------------
Number of EVIs                : 2
Number of Local EAD Entries   : 0
Number of Remote EAD Entries  : 0
Number of Local MAC Routes    : 0
Number of Local MAC Routes    : 5
          MAC                 : 5
          MAC-IPv4            : 0
          MAC-IPv6            : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes   : 7
          MAC                 : 7
          MAC-IPv4            : 0
          MAC-IPv6            : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels     : 5
Number of ES Entries          : 9
Number of Neighbor Entries    : 1
EVPN Router ID                : 192.168.0.1
BGP Router ID                 : ::
```

```
BGP ASN                         : 100
PBB BSA MAC address             : 0207.1fee.be00
Global peering timer            :      3 seconds
Global recovery timer           :     30 seconds
EVPN cost-out                   : TRUE
      startup-cost-in timer     : Not configured


/* Verify the no cost-out configuration */

Router# show evpn summary
Fri Apr  7 07:45:22.311 IST
Global Information
----------------------------
Number of EVIs                  : 2
Number of Local EAD Entries     : 0
Number of Remote EAD Entries    : 0
Number of Local MAC Routes      : 0
Number of Local MAC Routes      : 5
          MAC                   : 5
          MAC-IPv4              : 0
          MAC-IPv6              : 0
Number of Local ES:Global MAC   : 12
Number of Remote MAC Routes     : 7
          MAC                   : 7
          MAC-IPv4              : 0
          MAC-IPv6              : 0
Number of Local IMCAST Routes   : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels       : 5
Number of ES Entries            : 9
Number of Neighbor Entries      : 1
EVPN Router ID                  : 192.168.0.1
BGP Router ID                   : ::
BGP ASN                         : 100
PBB BSA MAC address             : 0207.1fee.be00
Global peering timer            :      3 seconds
Global recovery timer           :     30 seconds
EVPN cost-out                   : FALSE
      startup-cost-in timer     : Not configured


/* Verify the startup-cost-in timer configuration */

Router# show evpn summary
Fri Apr  7 07:45:22.311 IST
Global Information
----------------------------
Number of EVIs                  : 2
Number of Local EAD Entries     : 0
Number of Remote EAD Entries    : 0
Number of Local MAC Routes      : 0
Number of Local MAC Routes      : 5
          MAC                   : 5
          MAC-IPv4              : 0
          MAC-IPv6              : 0
Number of Local ES:Global MAC   : 12
Number of Remote MAC Routes     : 7
          MAC                   : 7
          MAC-IPv4              : 0
          MAC-IPv6              : 0
Number of Local IMCAST Routes   : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels       : 5
```

```
Number of ES Entries          : 9
Number of Neighbor Entries    : 1
EVPN Router ID                : 192.168.0.1
BGP Router ID                 : ::
BGP ASN                       : 100
PBB BSA MAC address           : 0207.1fee.be00
Global peering timer          :      3 seconds
Global recovery timer         :     30 seconds
EVPN node cost-out            : TRUE
        startup-cost-in timer : 6000
```

CFM Support for EVPN

Ethernet Connectivity Fault Management (CFM) is a service-level OAM protocol that provides tools for monitoring and troubleshooting end-to-end Ethernet services per VLAN. This includes proactive connectivity monitoring, fault verification, and fault isolation. CFM can be deployed in an EVPN network. You can monitor the connections between the nodes using CFM in an EVPN network.

Restrictions

CFM for EVPN is supported with the following restrictions:

- In an active-active multi-homing scenario, when monitoring the connectivity between a multi-homed CE device and the PE devices to which it is connected, CFM can only be used across each individual link between a CE and a PE. Attempts to use CFM on the bundle between CE and PE devices cause sequence number errors and statistical inaccuracies.
- There is a possibility of artefacts in loopback and linktrace results. Either a loopback or linktrace may report multiple results for the same instance, or consecutive instances of a loopback and linktrace between the same two endpoints may produce different results.

Control Word Support for ELAN

Table 20: Feature History Table

Feature Name	Release Information	Feature Description

Control-word support for EVPN Bridge-Mode (E-LAN)	Release 7.4.1	Control word is now supported and enabled by default in ELAN mode. If the **control-word-disable** command is not configured, ensure to configure it under EVPN or EVI configuration mode before an upgrade to avoid inconsistent behaviour with routers before this release.
		`Router# configure`
		`Router(config)# evpn`
		`Router(config-evpn)# evi 1`
		`Router(config-evpn-instance)# control-word-disable`
		`// Apply to interop with older releases EVPN ELAN`
		If you want to enable **control-word** command for EVPN Bridging feature, then you must configure it only when both the endpoints run Release 7.4.1 or later.

Note Control word is enabled by default in ELAN mode as well. If the **control-word-disable** command is not configured, ensure to configure it under EVPN or EVI configuration mode before an upgrade to avoid inconsistent behaviour with routers before Release 7.4.1.

If you want to enable **control-word** command for EVPN Bridging feature, then you must configure it only when both the endpoints run Release 7.4.1 or later.

EVPN Multiple Services per Ethernet Segment

EVPN Multiple Services per Ethernet Segment feature allows you to configure multiple services over single Ethernet Segment (ES). Instead of configuring multiple services over multiple ES, you can configure multiple services over a single ES.

You can configure the following services on a single Ethernet Bundle; you can configure one service on each sub-interface.

- Flexible cross-connect (FXC) service. It supports VLAN Unaware, VLAN Aware, and Local Switching modes.

  For more information, see *Configure Point-to-Point Layer 2 Services* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS Series Routers*.

- EVPN-VPWS Xconnect service

  For more information, see *EVPN Virtual Private Wire Service (VPWS)* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS Series Routers*.

- EVPN Integrated Routing and Bridging (IRB)

  For more information, see *Configure EVPN IRB* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS Series Routers*.

- Native EVPN

  For more information see, *EVPN Features* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS Series Routers*.

All these services are supported only on all-active multihoming scenario.

Configure EVPN Multiple Services per Ethernet Segment

Consider a customer edge (CE) device connected to two provider edge (PE) devices through Ethernet Bundle interface 22001. Configure multiple services on Bundle Ethernet sub-interfaces.

Configuration Example

Consider Bundle-Ether22001 ES, and configure multiple services on sub-interface.

```
/* Configure attachment circuits */
Router# configure
Router(config)# interface Bundle-Ether22001.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 12
```

```
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.13 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 13
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.14 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 14
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 1
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.2 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 2
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.3 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 3
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.4 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 4
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit

/*Configure VLAN Unaware FXC Service */
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc_mh1
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether22001.1
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether22001.2
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether22001.3
Router(config-l2vpn-fxs-vu)# neighbor evpn evi 21006 target 22016
Router(config-l2vpn-fxs-vu)# commit

/* Configure VLAN Aware FXC Service */
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 24001
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.12
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.13
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.14
Router(config-l2vpn-fxs-va)# commit

/* Configure Local Switching - Local switching is supported only on VLAN-aware FXC */
PE1
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 31400
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.1400
Router(config-l2vpn-fxs-va)# interface Bundle-Ether23001.1400
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs)# exit
PE2
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 31401
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.1401
Router(config-l2vpn-fxs-va)# interface Bundle-Ether23001.1401
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs)# exit
```

```
/* Configure EVPN-VPWS xconnect service and native EVPN with IRB */

Router# configure
Router(config)# interface Bundle-Ether22001.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 11
Router(config-l2vpn-subif)# rewrite ingress tag pop 2 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

Router# configure
Router(config)# interface Bundle-Ether22001.21 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q  21
Router(config-l2vpn-subif)# rewrite ingress tag pop 2 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

Router# configure
Route(config)# l2vpn
Router(config-l2vpn)# xconnect group xg22001
Router(config-l2vpn-xc)# p2p evpn-vpws-mclag-22001
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether22001.11
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 22101 target 220101 source 220301
Router(config-l2vpn-xc-p2p-pw)# commit
Router(config-l2vpn-xc-p2p-pw)# exit

Router # configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group native_evpn1
Router (config-l2vpn-bg)#  bridge-domain bd21
Router (config-l2vpn-bg-bd)# interface Bundle-Ether22001.21
Router (config-l2vpn-bg-bd-ac)# routed interface BVI21
Router (config-l2vpn-bg-bd-bvi)# evi 22021
Router (config-l2vpn-bg-bd-bvi)# commit
Router (config-l2vpn-bg-bd-bvi)# exit

/* Configure Native EVPN */
Router # configure
Router (config)# evpn
Router (config-evpn)# interface Bundle-Ether22001
Router (config-evpn-ac)# ethernet-segment identifier type 0 ff.ff.ff.ff.ff.ff.ff.ff.ee
Router (config-evpn-ac-es)# bgp route-target 2200.0001.0001
Router (config-evpn-ac-es)# exit
Router (config-evpn)# evi 24001
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64:24001
Router (config-evpn-evi-bgp)# route-target export 64:24001
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 21006
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target route-target 64:10000
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22101
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64:22101
Router (config-evpn-evi-bgp)# route-target export 64:22101
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22021
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64: 22021
```

```
Router (config-evpn-evi-bgp)# route-target export 64: 22021
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn-evi)# advertise-mac
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22022
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64: 22022
Router (config-evpn-evi-bgp)# route-target export 64: 22022
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# advertise-mac
Router (config-evpn-evi)# commit
Router (config-evpn-evi)# exit
```

Running Configuration

```
/* Configure attachment circuits */
interface Bundle-Ether22001.12 l2transport
encapsulation dot1q 1 second-dot1q 12
!
interface Bundle-Ether22001.13 l2transport
encapsulation dot1q 1 second-dot1q 13
!
interface Bundle-Ether22001.14 l2transport
encapsulation dot1q 1 second-dot1q 14
!
interface Bundle-Ether22001.1 l2transport
 encapsulation dot1q 1 second-dot1q 1
!
interface Bundle-Ether22001.2 l2transport
encapsulation dot1q 1 second-dot1q 2
!
interface Bundle-Ether22001.3 l2transport
encapsulation dot1q 1 second-dot1q 3
!
interface Bundle-Ether22001.4 l2transport
encapsulation dot1q 1 second-dot1q 4

/*Configure VLAN Unaware FXC Service */
flexible-xconnect-service vlan-unaware fxc_mh1
  interface Bundle-Ether22001.1
  interface Bundle-Ether22001.2
  interface Bundle-Ether22001.3
 neighbor evpn evi 21006 target 22016
!
/*Configure VLAN Aware FXC Service */
l2vpn
 flexible-xconnect-service vlan-aware evi 24001
   interface Bundle-Ether22001.12
   interface Bundle-Ether22001.13
   interface Bundle-Ether22001.14

/* Configure Local Switching */
flexible-xconnect-service vlan-aware evi 31400
 interface Bundle-Ether22001.1400
 interface Bundle-Ether23001.1400
!
flexible-xconnect-service vlan-aware evi 31401
 interface Bundle-Ether22001.1401
 interface Bundle-Ether23001.1401
!
```

```
/* Configure EVPN-VPWS xconnect service and native EVPN with IRB */
interface Bundle-Ether22001.11 l2transport
  encapsulation dot1q 1 second-dot1q 11
  rewrite ingress tag pop 2 symmetric
 !
interface Bundle-Ether22001.21 l2transport
  encapsulation dot1q 1 second-dot1q  21
  rewrite ingress tag pop 2 symmetric
 !
!
l2vpn
xconnect group xg22001
p2p evpn-vpws-mclag-22001
 interface Bundle-Ether22001.11
 neighbor evpn evi 22101 target 220101 source 220301
!
bridge group native_evpn1
   bridge-domain bd21
   interface Bundle-Ether22001.21
    routed interface BVI21
     evi 22021
  !
/* Configure Native EVPN */
Evpn
 interface Bundle-Ether22001
  ethernet-segment   identifier type 0 ff.ff.ff.ff.ff.ff.ff.ff.ee
  bgp route-target 2200.0001.0001
  !
  evi 24001
   bgp
    route-target import 64:24001
    route-target export 64:24001
   !
   evi 21006
    bgp
      route-target 64:100006
   !
    evi 22101
     bgp
       route-target import 64:22101
       route-target export 64:22101
     !
   evi 22021
    bgp
      route-target import 64:22021
      route-target export 64:22021
    !
    advertise-mac
  !
  evi 22022
   bgp
    route-target import 64:22022
    route-target export 64:22022
   !
    advertise-mac
  !
```

Verification

Verify if each of the services is configured on the sub-interface.

```
Router# show l2vpn xconnect summary
```

```
Number of groups: 6
Number of xconnects: 505  Up: 505  Down: 0  Unresolved: 0 Partially-programmed: 0
AC-PW: 505  AC-AC: 0  PW-PW: 0 Monitor-Session-PW: 0
Number of Admin Down segments: 0
Number of MP2MP xconnects: 0
 Up 0 Down 0
Advertised: 0 Non-Advertised: 0


Router# show l2vpn xconnect-service summary
Number of flexible xconnect services: 74
 Up: 74

Router# show l2vpn flexible-xconnect-service summary
Number of flexible xconnect services: 74
 Up: 74

Router# show l2vpn flexible-xconnect-service name fxc_mh1
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
Flexible XConnect Service  Segment
Name    ST  Type  Description  ST
----------------------- ----------------------------- -----------------------------
fxc_mh1 UP  AC:   BE22001.1    UP
            AC:   BE22001.2    UP
            AC:   BE22001.3    UP
-------------------------------------------------------------------------------------

Router# show l2vpn flexible-xconnect-service evi 24001
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
Flexible XConnect Service  Segment
Name      ST  Type  Description  ST
----------------------- ----------------------------- -----------------------------
evi:24001 UP  AC:   BE22001.11    UP
              AC:   BE22001.12    UP
              AC:   BE22001.13    UP
              AC:   BE22001.14    UP
-------------------------------------------------------------------------------------

Router# show l2vpn flexible-xconnect-service name evi:24001

Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
Flexible XConnect Service  Segment
Name      ST  Type  Description  ST
----------------------- ----------------------------- -----------------------------
evi:24001 UP  AC:   BE22001.11    UP
              AC:   BE22001.12    UP
              AC:   BE22001.13    UP
              AC:   BE22001.14    UP
-------------------------------------------------------------------------------------


Router# show l2vpn xconnect group xg22001 xc-name evpn-vpws-mclag-22001
Fri Sep 1 17:28:58.259 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
XConnect                               Segment 1           Segment 2
Group     Name                   ST     Description ST      Description                 ST
----------------------- ----------------------------- -----------------------------------
xg22001  evpn-vpws-mclag-22001   UP    BE22001.101  UP      EVPN 22101, 220101,64.1.1.6 UP
-------------------------------------------------------------------------------------
```

Associated Commands

- evpn
- evi
- ethernet-segment
- advertise-mac
- show evpn ethernet-segment
- show evpn evi
- show evpn summary
- show l2vpn xconnect summary
- show l2vpn flexible-xconnect-service
- show l2vpn xconnect group

EVPN Single-Flow-Active Multihoming Load-Balancing Mode

Table 21: Feature History Table

Feature Name	Release Information	Feature Description
Single-Flow Active (for Access Rings) - VPNv4 Hosts	Release 7.4.1	This feature extends the current implementation of EVPN Single-Flow-Active Multihoming Load-Balancing Mode, on Cisco routers with VPNv4 routes.
EVPN Single-Flow-Active Multihoming Load-Balancing Mode	Release 7.3.1	This feature introduces EVPN Single-Flow-Active multihoming mode to connect PE devices in an access network that run Layer 2 access gateway protocols. In this mode, only the PE that first advertises the host MAC address in a VLAN forwards the traffic in a specific flow. When the primary link fails, the traffic quickly switches to the standby PE that learns the MAC address from the originated path, thereby providing fast convergence. A keyword, **single-flow-active** is added to the **load-balancing-mode** command.

In a ring topology, only one of the PEs, which is the active PE, sends and receives the traffic to prevent a traffic loop. When the link to the active PE fails, the traffic switches over to the standby PE. Traffic switchover

takes a while because the standby PE has to learn the MAC addresses of the connected hosts. There's a traffic loss until the traffic switch over happens.

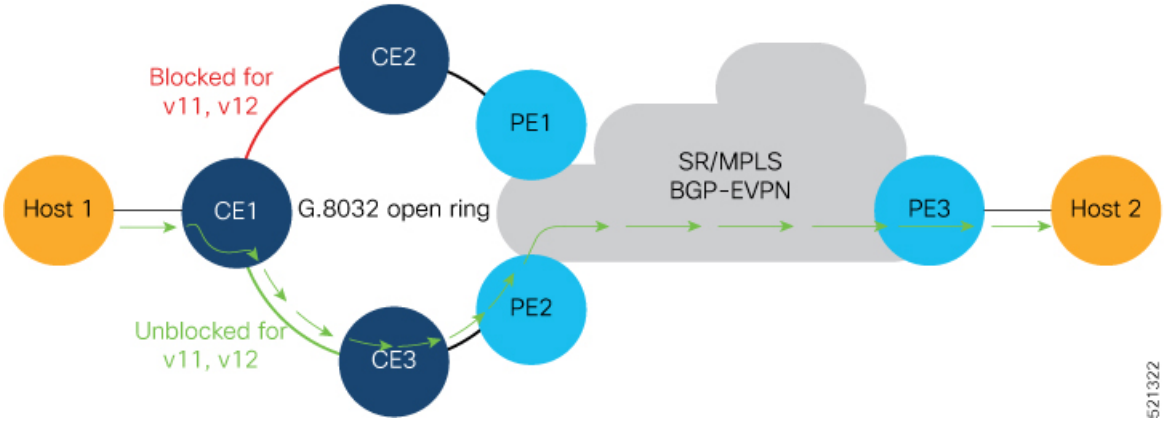
The EVPN Single-Flow-Active multihoming mode connects PE devices in an access network, and in the event of active link failure the switchover happens immediately and reduces the traffic loss.

Both active and standby PEs learn the MAC addresses of the connected host. The PE that learns the MAC address of the host directly is called the Primary (active) PE. The primary PE advertises the learnt MAC addresses to the peer PE, which is referred as standby PE. As the standby PE learns the MAC address of the host through the active PE, this learnt path is referred to as the reoriginated path.

When the primary link fails, the convergence happens fast and the traffic is sent through the standby PE (reoriginated path).

Let us understand how EVPN single flow-active mode helps in fast convergence:

- In this topology, the access network devices are connected through a ring topology. The access network uses Layer-2 gateway protocols such as G.8032, MPLS-TP, STP,REP-AG or MSTP-AG to prevent traffic loop due to continuous flooding.The access protocols are not supported on Cisco NC57 line cards, but only xconnect is supported in access network.



- Host 1 is connected to CE1.
- CE1 is connected to both PE1 and PE2, thus is multihomed.
- PE1 and PE2 are Multihoming devices.
- Both PE1 and PE2 is configured with the same non-zero Ethernet Segment ID (ESI) number `0 36.37.00.00.00.00.00.11.00` for the bundle interface to enable multihoming of the host (CE1).
- PE1 and PE2 belongs to te same VLAN and hence configured with the same EVPN instance (EVI) 100.

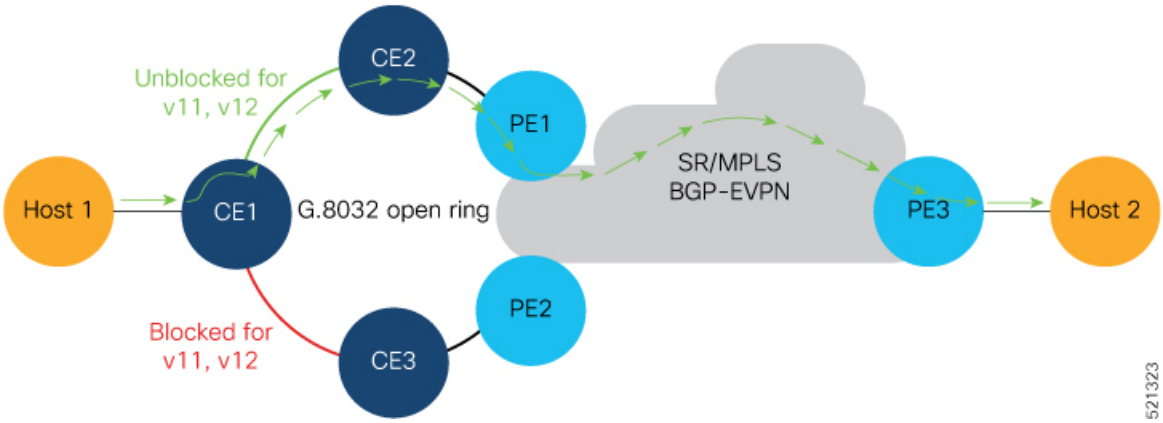
Traffic Flow

- Consider a traffic flow from Host 1 to Host 2. The traffic is sent from Host 1 to CE1.
- In this ring topology, the link between CE1 to CE2 is in the blocked state; the link between CE1 to CE3 is in the forwarding state. Hence, CE1 sends the traffic to PE2 through CE3.
- PE2 first learns the MAC address of Host1 through CE1. PE2 advertises the learnt MAC address to the peering PE1.

- As PE2 has learnt the MAC address directly from Host 1, and acts as an active PE.
- The PE which originates the MAC route due to access learning sets the default BGP local preference attribute value to 100.
- PE1 learns the MAC address from PE2 and acts as a stand-by PE. As PE1 gets the reoriginated MAC route from PE2, PE1 sets the BGP local preference attribute value to 80.
- The PE that has the higher local preference always sends and receives the traffic. Thus PE1 sends the traffic to PE3. PE3 sends the traffic to Host 2.

Failure Scenario

When the link between CE1 and CE3 is down or when the link between CE3 and PE2 is down, traffic is sent through PE1.



- When the link fails, the link CE1-CE2 changes to the forwarding state.
- PE1 learns the MAC address of Host 1 directly and advertises the learnt MAC address to PE2.
- PE1 sends the traffic to Host 2 through the remote PE3 with a BGP local preference value of 100.
- PE3 sends and receives the traffic from PE1 until the access link between CE1 and CE2 changes to the blocked state.

Restrictions

Single-Flow Active is not supported for EVPN VPWS.

Configuration Example

- Configure both PE1 and PE2 with the same EVI of 100.
- Configure both PE1 and PE2 with the same ESI 0 36.37.00.00.00.00.00.11.01.

Perform these tasks on both PE1and PE2.

```
/* Configure advertisement of MAC routes */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
```

```
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# root

/* Configure single-flow-active load-balancing mode */
Router(config)# evpn
Router(config-evpn)# interface bundle-ether 1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 36.37.00.00.00.00.00.11.01
Router(config-evpn-ac-es)# load-balancing-mode single-flow-active
Router(config-evpn-ac-es)# convergence
Router(config-evpn-ac-es-conv)# mac-mobility
Router(config-evpn-ac-es-conv)# exit

Router(config-evpn-ac-es)# root

/* Configure bridge domain and associating the evi to the bridge domain */
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 100
Router(config-l2vpn-bg)# bridge-domain 100
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1.2
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)# evi 100
Router(config-l2vpn-bg-bd-evi)# root
Router(config)# interface Bundle-Ether1.2 l2transport
Router(config-l2vpn-subif)#encapsulation dot1q 2
Router(config-l2vpn-subif)#commit
```

Running Configuration

```
evpn
 evi 100
  advertise-mac
  !
 !
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 36.37.00.00.00.00.00.11.01
   load-balancing-mode single-flow-active
  convergence
      mac-mobility
   !
  !
 !
!
l2vpn
  bridge group 100
   bridge-domain 100
    interface Bundle-Ether1
   !
   evi 100
   !
  !
 !
 interface Bundle-Ether1.2 l2transport
  encapsulation dot1q 2
!
!
```

Verification

Verify the Ethernet Segment Status:

- Verify that the Ethernet Segment Id is the same as that you have configured: In this example, you notice that the ESI on PE1 is 0 36.37.00.00.00.00.00.11.01.
- Verify that the Single-flow-active mode is enabled in the Topology section.

```
Router#show  evpn  ethernet-segment  interface  be 1 detail
Legend:
B   - No Forwarders EVPN-enabled,
C   - MAC missing (Backbone S-MAC PBB-EVPN / Grouping ES-MAC vES),
RT  - ES-Import Route Target missing,
E   - ESI missing,
H   - Interface handle missing,
I   - Name (Interface or Virtual Access) missing,
M   - Interface in Down state,
O   - BGP End of Download missing,
P   - Interface already Access Protected,
Pf  - Interface forced single-homed,
R   - BGP RID not received,
S   - Interface in redundancy standby state,
X   - ESI-extracted MAC Conflict
SHG - No local split-horizon-group label allocated
Hp  - Interface blocked on peering complete during HA event
Rc  - Recovery timer running during peering sequence

Ethernet Segment Id            Interface                          Nexthops
0 36.37.00.00.00.00.00.11.01  BE1                                172.16.0.4
                                                                  172.16.0.5
ES to BGP Gates   : Ready
ES to L2FIB Gates : P
Main port         :
Interface name    : Bundle-Ether1
Interface MAC     : b0a6.51e5.00dd
IfHandle          : 0x2000802c
State             : Up
Redundancy        : Not Defined
ESI type          : 0
Value             : 07.0807.0807.0807.0800
ES Import RT      : 0708.0708.0708 (from ESI)
Source MAC        : 0000.0000.0000 (N/A)
Topology          :
Operational     : MH, Single-flow-active
Configured      : Single-flow-active
Service Carving   : Auto-selection
Multicast         : Disabled
Convergence       : MAC-Mobility
Mobility-Flush    : Debounce 1 sec, Count 0, Skip 0
                  : Last n/a
Peering Details   : 2 Nexthops
172.16.0.4 [MOD:P:00:T]
172.16.0.5 [MOD:P:00:T]
Service Carving Synchronization:
Mode              : NONE
Peer Updates      :
172.16.0.4 [SCT: N/A]
172.16.0.5 [SCT: N/A]
Service Carving Results:
Forwarders    : 1
Elected         : 0
Not Elected   : 0
EVPN-VPWS Service Carving Results:
Primary           : 0
Backup            : 0
Non-DF            : 0
```

```
MAC Flushing mode: STP-TCN
Peering timer    : 3 sec [not running]
Recovery timer   : 30 sec [not running]
Carving timer    : 0 sec [not running]
HRW Reset timer  : 5 sec [not running]
Local SHG label  : 24007
Remote SHG labels: 1
24010            : nexthop 172.16.0.5
Access signal mode: Bundle OOS (Default)
```

```
Router#show l2vpn protection main-interface
Main Interface ID              # of subIntf Protected  Protect Type
Bundle-Ether1                  2            Yes        ERP

Instance : 1
State    : FORWARDING

Sub-Intf # : 2

Flush    # : 6
```

Associated Commands

- **load-balancing-mode**
- **show evpn ethernet-segment**

EVPN Convergence Using NTP Synchronization

Table 22: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Convergence Using NTP Synchronization	Release 7.3.1	This feature leverages the NTP clock synchronization mechanism to handle the transfer of DF role from one edge device to another. In this mechanism, the newly added or recovered PE advertises the Service Carving Timestamp along with the current time to peering PEs. This improves convergence by reducing the time for DF election from three seconds to a few tens of milliseconds. The **show evpn ethernet-segment** command is modified to display the Service-Carving wall clock Timestamp (SCT).

In Ethernet VPN, depending on the load-balancing mode, the Designated Forwarder (DF) is responsible for forwarding Unicast, Broadcast, Unknown Unicast, and Multicast (BUM) traffic to a multihomed Customer Edge (CE) device on a given VLAN on a particular Ethernet Segment (ES).

The DF is selected from the set of multihomed edge devices attached to a given ES. When a new edge router joins the peering group either through failure recovery or booting up of a new device, the DF election process is triggered.

By default, the process of transferring the DF role from one edge device to another takes 3 seconds. The traffic may be lost during this period.

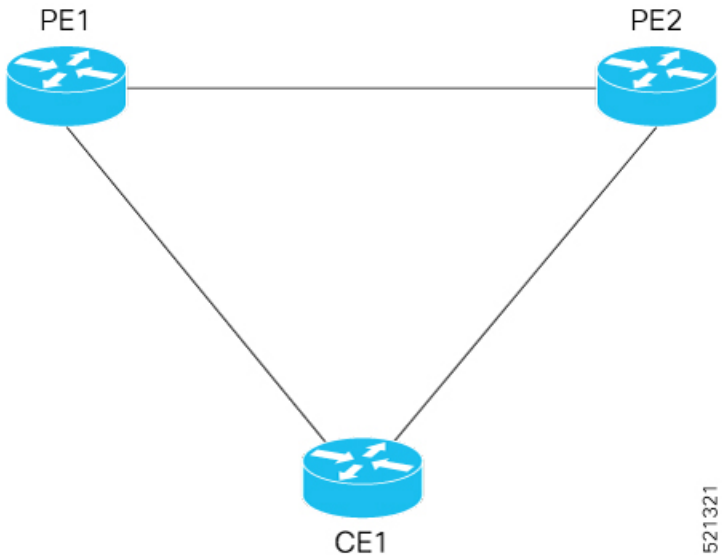
The NTP synchronization mechanism for fast DF election upon recovery leverages the NTP clock synchronization to better align DF events between peering PEs.

If all edge devices attached to a given Ethernet Segment are clock-synchronized with each other using NTP, the default DF election time reduces from 3 seconds to few tens of milliseconds, thereby reducing traffic loss.

Note If the NTP is not synchronized with the NTP server when the EVPN Ethernet Segment interface is coming up, EVPN performs normal DF election.

Let's understand how NTP synchronization works:

Figure 44: EVPN Convergence Using NTP Synchronization



In this topology, CE1 is multihomed to PE1 and PE2.

- PE1 joins the peering group after failure recovery at time (t) = 99 seconds.
- When PE1 joins the peering group, PE1 advertises Route-Type 4 at t = 100 seconds with target Service Carving Time (SCT) value t = 103 seconds to PE2.
- PE2 receives peering Route-Type 4 and learns the DF election time of PE1 to be t =103 seconds.
- If all the peers support NTP, PE2 starts a timer based on the SCT received from PE1 along with a skew value in the Service Carving Time. The skew values are used to eliminate any potential duplicate traffic or loops. Both PE1 and PE2 carves at time t = 103 seconds.

Benefits

- Helps in fast convergence during a primary link recovery
- Supports all the existing load-balancing modes:

- All-active multihoming
- Single-active multihoming
- Port-active multihoming
- Single-Flow-Active multihoming

Limitations

- All devices attached to a given Ethernet Segment must be configured with NTP. If one of the devices doesn't support NTP clock, the mechanism falls back to default timers.

Verification

Use the **show evpn ethernet-segment** command to view the **Service Carving Time** of the edge device.

For example,

```
Router# show evpn ethernet-segment interface Bundle-Ether200 carving detail

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0053.5353.5353.5353.5301 BE200                             10.0.0.1
                                                           172.16.0.1
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether200
     Interface MAC  : 2c62.34fd.2485
     IfHandle       : 0x20004334
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 53.5353.5353.5353.5301
  ES Import RT      : 8888.8888.8888 (Local)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, All-active
     Configured     : All-active (AApF) (default)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       : Reroute
  Peering Details   : 2 Nexthops
     91.0.0.10 [MOD:P:00:T]
     91.0.0.30 [MOD:P:7fff:T]
  Service Carving Synchronization:
     Mode           : NTP_SCT
     Peer Updates   :
               10.0.0.1 [SCT: 2020-10-16 00:28:22:559418]
               10.0.0.3 [SCT: 2020-10-22 17:46:36:587875]
  Service Carving Results:
     Forwarders     : 128
     Elected        : 64

     Not Elected    : 64
```

Associated Commands

- **Show evpn ethernet-segment**

EVPN MPLS Seamless Integration with VPLS

Table 23: Feature History Table

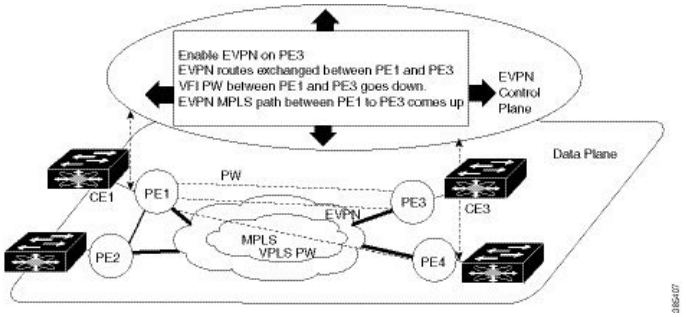
Feature Name	Release Information	Feature Description
EVPN MPLS Seamless Integration with VPLS	Release 7.4.2	This feature enables the co-existence of PE nodes running EVPN and VPLS for the same VPN instance. VPLS or legacy network can be upgraded to the next generation EVPN network without service disruption. You can introduce EVPN service on all the selected VPLS provider edge (PE) nodes simultaneously. However, to avoid traffic disruption, provision EVPN service on existing VPLS-enabled PEs one by one. This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.

Migrate VPLS Network to EVPN Network through Seamless Integration

In EVPN network, VPN instances are identified by EVPN instance ID (EVI-ID). Similar to other L2VPN technologies, EVPN instances are also associated with route-targets and route-distinguisher. EVPN uses control plane for learning and propagating MAC unlike traditional VPLS, where MAC is learnt in the data plane (learns using "flood and learn technique"). In EVPN, MAC routes are carried by MP-BGP protocol. In EVPN enabled PEs, PEs import the MAC route along with the label to their respective EVPN forwarding table only if their route targets (RTs) match. An EVPN PE router is capable of performing VPLS and EVPN L2 bridging in the same VPN instance. When both EVPN and BGP-AD PW are configured in a VPN instance, the EVPN PEs advertise the BGP VPLS auto-discovery (AD) route as well as the BGP EVPN Inclusive Multicast route (type-3) for a given VPN Instance. Route type-3 referred to as ingress replication multicast route, is used to send broadcast, unknown unicast, and multicast (BUM) traffic. Other remote PEs import type-3 routes for the same VPN instance only if the sending PE RTs match with their configured RT. Thus, at the end of these route-exchanges, EVPN capable PEs discover all other PEs in the VPN instance and their associated capabilities. The type-3 routes used by PE to send its BUM traffic to other PEs ensure that PEs with the same RTs receive the BUM traffic. EVPN advertises the customer MAC address using type-2 route.

EVPN MPLS Seamless Integration with VPLS allows you to upgrade the VPLS PE routers to EVPN one by one without any network service disruption. Consider the following topology where PE1, PE2, PE3, and PE4 are interconnected in a full-meshed network using VPLS PW.

Figure 45: EVPN MPLS Seamless Integration with VPLS



The EVPN service can be introduced in the network one PE node at a time. The VPLS to EVPN migration starts on PE1 by enabling EVPN in a VPN instance of VPLS service. As soon as EVPN is enabled, PE1 starts advertising EVPN inclusive multicast route to other PE nodes. Since PE1 does not receive any inclusive multicast routes from other PE nodes, VPLS pseudo wires between PE1 and other PE nodes remain active. PE1 keeps forwarding traffic using VPLS pseudo wires. At the same time, PE1 advertises all MAC address learned from CE1 using EVPN route type-2. In the second step, EVPN is enabled in PE3. PE3 starts advertising inclusive multicast route to other PE nodes. Both PE1 and PE3 discover each other through EVPN routes. As a result, PE1 and PE3 shut down the pseudo wires between them. EVPN service replaces VPLS service between PE1 and PE3. At this stage, PE1 keeps running VPLS service with PE2 and PE4. It starts EVPN service with PE3 in the same VPN instance. This is called EVPN seamless integration with VPLS. The VPLS to EVPN migration then continues to remaining PE nodes. In the end, all four PE nodes are enabled with EVPN service. VPLS service is completely replaced with EVPN service in the network. All VPLS pseudo wires are shut down.

Configure EVPN on the Existing VPLS Network

Perform the following tasks to configure EVPN on the existing VPLS network.

- Configure L2VPN EVPN address-family
- Configure EVI and corresponding BGP route-targets under EVPN configuration mode
- Configure EVI under a bridge-domain

See EVI Configuration Under L2VPN Bridge-Domain, on page 264 section for how to migrate various VPLS-based network to EVPN.

Configure L2 EVPN Address-Family

Perform this task to enable EVPN address family under both BGP and participating neighbor.

Configuration Example

```
Router# configure
Router(config)#router bgp 65530
Router(config-bgp)#nsr
Router(config-bgp)#bgp graceful-restart
Router(config-bgp)#bgp router-id 200.0.1.1
Router(config-bgp)#address-family l2vpn evpn
```

```
Router(config-bgp-af)#exit
Router(config-bgp)#neighbor 200.0.4.1
Router(config-bgp-nbr)#remote-as 65530
Router(config-bgp-nbr)#update-source Loopback0
Router(config-bgp-nbr)#address-family l2vpn evpn
Router(config-bgp-nbr-af)#commit
```

Running Configuration

```
configure
 router bgp 65530
  nsr
  bgp graceful-restart
  bgp router-id 200.0.1.1
  address-family l2vpn evpn
  !
  neighbor 200.0.4.1
   remote-as 65530
   update-source Loopback0
   address-family l2vpn evpn
   !
 !
```

Configure EVI and Corresponding BGP Route Target under EVPN Configuration Mode

Perform this task to configure EVI and define the corresponding BGP route targets. Also, configure advertise-mac, else the MAC routes (type-2) are not advertised.

Configuration Example

```
Router# configure
Router(config)#evpn
Router(config-evpn)#evi 1
Router(config-evpn-evi-bgp)#bgp
Router(config-evpn-evi-bgp)#table-policy spp-basic-6
Router(config-evpn-evi-bgp)#route-target import 100:6005
Router(config-evpn-evi-bgp)#route-target export 100:6005
Router(config-evpn-evi-bgp)#exit
Router(config-evpn-evi)#advertise-mac
Router(config-evpn-evi)#commit
```

Running Configuration

```
configure
 evpn
  evi
   bgp
    table-policy spp-basic-6
    route-target import 100:6005
    route-target export 100:6005
    !
   advertise-mac
   !
  !
 !
```

Configure EVI under a Bridge Domain

Perform this task to configure EVI under the corresponding L2VPN bridge domain.

Configuration Example

```
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface GigabitEthernet
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#evi 1
Router(config-l2vpn-bg-bd-evi)#exit
Router(config-l2vpn-bg-bd)#vfi v1
Router(config-l2vpn-bg-bd-vfi)#neighbor 10.1.1.2 pw-id 1000
Router(config-l2vpn-bg-bd-vfi-pw)#mpls static label local 20001 remote 10001
Router(config-l2vpn-bg-bd-vfi-pw)#commit
```

Running Configuration

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface GigabitEthernet
    !
    evi 1
    !
   vfi v1
    neighbor 10.1.1.2 pw-id 1000
     mpls static label local 20001 remote 10001
     !
    !
    evi 1
 !
```

EVI Configuration Under L2VPN Bridge-Domain

The following examples show EVI configuration under L2VPN bridge-domain for various VPLS-based networks:

Note On reloading the Standby route processor (RP), traffic glitch occurs on the VPLS BUM traffic (< 1 second) in a single direction. Effective from release 7.1.1, this restrcition is not applicable.

MPLS Static Labels Based VPLS

```
l2vpn
 bridge group bg1
  bridge-domain bd-1-1
   interface GigabitEthernet
```

```
   !
   vfi vfi-1-1
    neighbor 200.0.2.1 pw-id 1200001
     mpls static label local 20001 remote 10001
    !
    neighbor 200.0.3.1 pw-id 1300001
     mpls static label local 30001 remote 10001
    !
    neighbor 200.0.4.1 pw-id 1400001
     mpls static label local 40001 remote 10001
    !
  !
  evi 1
!
```

AutoDiscovery BGP and BGP Signalling Based VPLS

```
l2vpn
bridge group bg1
bridge-domain bd-1-2
   interface GigabitEthernet
   !
   vfi vfi-1-2
    vpn-id 2
    autodiscovery bgp
     rd 101:2
     route-target 65530:200
     signaling-protocol bgp
      ve-id 11
      ve-range 16
     !
    !
   evi 2
  !
```

Targeted LDP-Based VPLS

```
bridge-domain bd-1-4
   interface GigabitEthernet
   !
   vfi vfi-1-4
    neighbor 200.0.2.1 pw-id 1200004
    !
    neighbor 200.0.3.1 pw-id 1300004
    !
    neighbor 200.0.4.1 pw-id 1400004
    !
   evi 3
   !
```

Verify EVPN Configuration

Use the following commands to verify EVPN configuration and MAC advertisement. Verify EVPN status, AC status, and VFI status.

- show l2vpn bridge-domain
- show evpn summary

- show bgp rt l2vpn evpn
- show evpn evi
- show l2route evpn mac all

Verify the state of the bridge domain, number of ACs, and VFIs.

```
Router#show l2vpn bridge-domain bd-name bd-1-1
Mon Feb 20 21:03:40.244 EST
Legend: pp = Partially Programmed.
Bridge group: bg1, bridge-domain: bd-1-1, id: 0, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 1 (1 up), VFIs: 1, PWs: 3 (2 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
  List of ACs:
    Gi0/2/0/0.1, state: up, Static MAC addresses: 0, MSTi: 2
  List of Access PWs:
  List of VFIs:
    VFI vfi-1-1 (up)
      Neighbor 200.0.2.1 pw-id 1200001, state: up, Static MAC addresses: 0
      Neighbor 200.0.3.1 pw-id 1300001, state: down, Static MAC addresses: 0
      Neighbor 200.0.4.1 pw-id 1400001, state: up, Static MAC addresses: 0
  List of Access VFIs:
 When PEs are evpn enabled, pseudowires that are associated with that BD will be brought
down. The VPLS BD pseudowires are always up.
```

Verify the number of EVI's configured, local and remote MAC-routes that are advertised.

```
Router#show evpn summary
Mon Feb 20 21:05:16.755 EST
----------------------------
Global Information
----------------------------
Number of EVIs                  : 6
Number of Local EAD Entries     : 0
Number of Remote EAD Entries    : 0
Number of Local MAC Routes      : 4
          MAC                   : 4
          MAC-IPv4              : 0
          MAC-IPv6              : 0
Number of Local ES:Global MAC   : 1
Number of Remote MAC Routes     : 0
          MAC                   : 0
          MAC-IPv4              : 0
          MAC-IPv6              : 0
Number of Remote SOO MAC Routes : 0
Number of Local IMCAST Routes   : 4
Number of Remote IMCAST Routes  : 4
Number of Internal Labels       : 0
Number of ES Entries            : 1
Number of Neighbor Entries      : 4
EVPN Router ID                  : 200.0.1.1
BGP ASN                         : 65530
PBB BSA MAC address             : 0026.982b.c1e5
Global peering timer            :      3 seconds
Global recovery timer           :     30 seconds
```

Verify EVPN route-targets.

```
Router#show bgp rt l2vpn evpn
Mon Feb 20 21:06:18.882 EST
EXTCOMM         IMP/EXP
RT:65530:1              1 / 1
RT:65530:2              1 / 1
RT:65530:3              1 / 1
RT:65530:4              1 / 1
Processed 4 entries
```

Verify EVPN MAC routes pertaining to specific VPN instance.

```
Router#show evpn evi vpn-id 1 mac
Mon Feb 20 21:36:23.574 EST

EVI        MAC address    IP address                   Nexthop
Label
---------- -------------- ---------------------------------------
---------------------------------
1      0033.0000.0001      ::                       200.0.1.1                          45106
```

Verify L2 routing.

```
Router#show l2route evpn mac all
Mon Feb 20 21:39:43.953 EST
Topo ID  Mac Address    Prod   Next Hop(s)
-------- -------------- ------ ---------------------------------------
0        0033.0000.0001 L2VPN  200.0.1.1/45106/ME
1        0033.0000.0002 L2VPN  200.0.1.1/45108/ME
2        0033.0000.0003 L2VPN  200.0.1.1/45110/ME
3        0033.0000.0004 L2VPN  200.0.1.1/45112/ME
```

Verifty EVPN route-type 2 routes.

```
Router#show bgp l2vpn evpn route-type 2
Mon Feb 20 21:43:23.616 EST
BGP router identifier 200.0.3.1, local AS number 65530
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 21
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200.0.1.1:1
*>i[2][0][48][0033.0000.0001][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:2
*>i[2][0][48][0033.0000.0002][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:3
*>i[2][0][48][0033.0000.0003][0]/104
                      200.0.1.1                     100      0 i
```

```
Route Distinguisher: 200.0.1.1:4
*>i[2][0][48][0033.0000.0004][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:1 (default for vrf bd-1-1)
*>i[2][0][48][0033.0000.0001][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:2 (default for vrf bd-1-2)
*>i[2][0][48][0033.0000.0002][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:3 (default for vrf bd-1-3)
*>i[2][0][48][0033.0000.0003][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:4 (default for vrf bd-1-4)
*>i[2][0][48][0033.0000.0004][0]/104
                      200.0.1.1                     100      0 i

Processed 8 prefixes, 8 paths
```

Verify inclusive multicast routes and route-type 3 routes.

```
Router#show bgp l2vpn evpn route-type 3
Mon Feb 20 21:43:33.970 EST
BGP router identifier 200.0.3.1, local AS number 65530
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 21
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200.0.1.1:1
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:2
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:3
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:4
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:1 (default for vrf bd-1-1)
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
*> [3][0][32][200.0.3.1]/80
                      0.0.0.0                                0 i
Route Distinguisher: 200.0.3.1:2 (default for vrf bd-1-2)
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
*> [3][0][32][200.0.3.1]/80
                      0.0.0.0                                0 i
Route Distinguisher: 200.0.3.1:3 (default for vrf bd-1-3)
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
*> [3][0][32][200.0.3.1]/80
                      0.0.0.0                                0 i
Route Distinguisher: 200.0.3.1:4 (default for vrf bd-1-4)
```

```
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
*> [3][0][32][200.0.3.1]/80
                      0.0.0.0                                0 i
```

Clear Forwarding Table

To clear an L2VPN forwarding table at a specified location, you can use the **clear l2vpn forwarding table** command. When BVI is present in the bridge domain, you might experience traffic loss during the command execution. Refer the following work-around to resolve such issues.

When you encounter such issues, delete the BVI and roll back the action. As a result, the traffic on the BVI returns to normal state. The following example shows how to delete the BVI and perform roll back action:

```
Router#clear l2vpn forwarding table location 0/0/CPU0
Fri Mar 24 09:34:02.083 UTC
Router(config)#no int BVI100
Router(config)#commit
Router#roll configuration las 1
Wed Dec 16 18:26:52.869 UTC
Loading Rollback Changes.
Loaded Rollback Changes in 1 sec
Committing
```

Note We can also clear the forwarding table by shutting and unshutting the interface.

Hierarchical EVPN Access Pseudowire

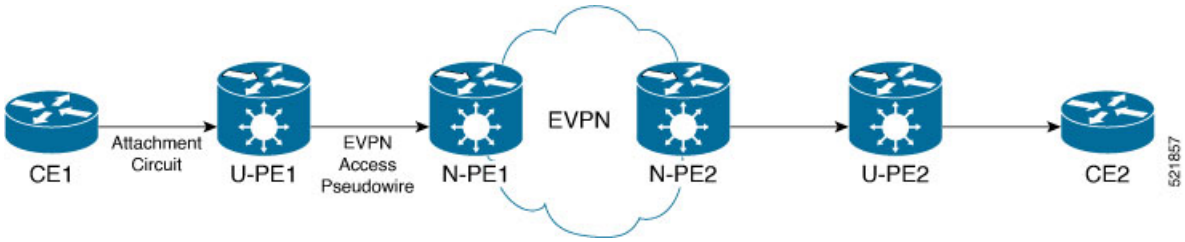
Table 24: Feature History Table

Feature Name	Release Information	Feature Description
Hierarchical EVPN Access Pseudowire	Release 7.6.1	You can configure EVPN VPWS in the access node under the same bridge domain as EVPN in the core to build a PW to the nearest high-end PE that stitches those access circuits using EVPN. This allows the access nodes to leverage the benefits of EVPN. This feature also allows you to reduce the number of pseudowires (PWs) between the network provider edge (N-PE) devices by replacing PE devices with user provider edge (U-PE) and network provider edge (N-PE) devices. This feature prevents signaling overhead and packet replication.

A standard VPN configuration comprises of CE devices and PE devices. With this feature, each PE device is replaced with a user provider edge (U-PE) and network provider edge (N-PE) devices. U-PE devices communicate with the CE devices and N-PE devices on the access side, and N-PE devices communicate with other N-PE devices on the core.

The Hierarchical EVPN Access Pseudowire feature allows you to reduce the number of pseudowires (PWs) between the network provider edge (N-PE) devices. The user provider edge (U-PE) device connects to the N-PE device using EVPN access pseudowire (PW) for each VPN instance. Each CE device is connected to a U-PE device through an attachment circuit.

Hierarchical EVPN Access Pseudowire Topology



In this topology, a user provider edge (U-PE1) device is connected to the CE1 through an attachment circuit. The U-PE1 device transports the CE1 traffic over an EVPN access PW to a network provider edge (N-PE1) device. The N-PE1 is connected with other N-PE2 in an EVPN core. On the N-PE1, the access PW coming from the U-PE1 is much like an AC. The U-PE is not part of the core with the other N-PEs. The N-PE forwards traffic from that access PW to the core PWs that are part of the EVPN core.

Configure Hierarchical EVPN Access Pseudowire

Perform the following task to configure Hierarchical EVPN Access Pseudowire feature on U-PEs and N-PEs.

Configuration Example

```
/* Configure U-PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XG1
Router(config-l2vpn-xc)# p2p P1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/31
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 target 33 source 33
Router(config-l2vpn-xc-p2p-pw)# commit

/* Configure N-PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group evpn
Router(config-l2vpn-bg)# bridge-domain evpn1
Router(config-l2vpn-bg-bd)# neighbor evpn evi 4 target 33
Router(config-l2vpn-bg-bd)# evi 1
Router(config-l2vpn-bg-bd-evi)# commit
```

Running Configuration

This section shows the Hierarchical EVPN Access Pseudowire running configuration.

```
/* U-PE1 Configuration */
l2vpn
 xconnect group XG1
  p2p P1
   interface TenGigE0/0/0/31 l2transport
```

```
   neighbor evpn evi 4 target 33 source 33
   !
!
/* N-PE1 Configuration */
l2vpn
bridge group evpn
  bridge-domain evpn1
   neighbor evpn evi 4 target 33
   evi 1
   !
  !
!
!
```

Verification

Verify the EVPN state, and the list of access PWs. The following is the sample output on N-PE1:

```
Router:N-PE1# show l2vpn bridge-domain bd-name evpn1
Wed Jun 16 09:22:30.328 EDT
Legend: pp = Partially Programmed.
Bridge group: evpn, bridge-domain: evpn1, id: 1, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 0 (0 up), VFIs: 0, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
  List of ACs:
  List of Access PWs:
      EVPN 4,33,192.168.0.4, state: up, Static MAC addresses: 0
  List of VFIs:
  List of Access VFIs:
```

EVPN Seamless Integration with VPWS

Table 25: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Seamless Integration with VPWS	Release 7.4.2	This feature enables you to seamlessly migrate the PE nodes from VPWS to EVPN-VPWS service without disruption in traffic. Such a migration offers your service providers the option to use VPWS or EVPN-VPWS services on PE nodes This feature introduces the **vpws-seamless-integration** command.

Although VPWS is a widely deployed Layer 2 VPN technology, some service providers prefer to migrate to EVPN service in their existing VPWS networks to leverage the benefits of EVPN services.

With EVPN-VPWS Seamless Integration feature, you can migrate the PE nodes from legacy VPWS service to EVPN-VPWS gradually and incrementally without any service disruption.

You can migrate an Attachment Circuit (AC) connected to a legacy VPWS pseudowire (PW) to an EVPN-VPWS PW either by using targeted-LDP signaling or BGP-AD signaling.

Instead of performing network-wide software upgrade at the same time on all PEs, this feature provides the flexibility to migrate one PE at a time. Thus allows the coexistence of legacy VPWS and EVPN-VPWS dual-stack in the core for a given L2 Attachment Circuit (AC) over the same MPLS network. You can enable this feature using the **vpws-seamless-integration** command.

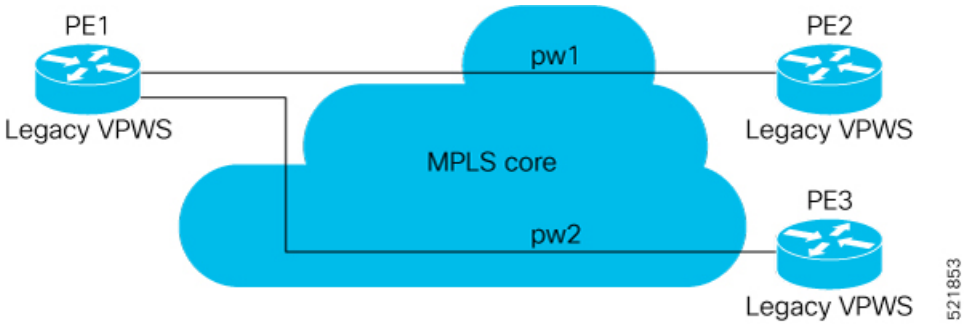
In an EVPN-VPWS network, VPN instances are grouped by EVPN Instance VPN ID (EVI) and identified by an ethernet tag or attachment circuit ID (AC-ID). EVI is also associated with route-targets and route-distinguisher.

During migration, an EVPN-VPWS PE router performs either VPWS or EVPN-VPWS L2 cross-connect for a given AC. When both EVPN-VPWS and BGP-AD PWs are configured for the same AC, the EVPN-VPWS PE during migration advertises the BGP VPWS Auto-Discovery (AD) route as well as the BGP EVPN Auto-Discovery (EVI/EAD) route and gives preference to EVPN-VPWS Pseudowire (PW) over the BGP-AD VPWS PW.

Let's understand how a legacy VPWS network can be migrated seamlessly to EVPN-VPWS with the following scenario:

Consider that a service provider plans to migrate VPWS node to an EVPN node one at a time. The service provider expects the migration to span over multiple years.

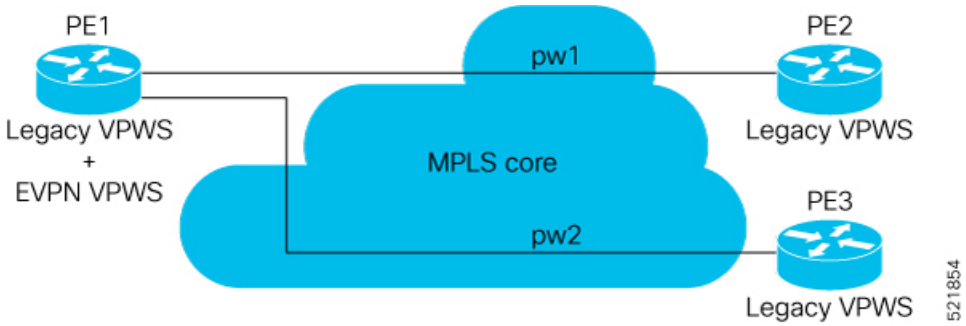
Figure 46:



In this topology, PE1, PE2, PE3 are provider edge devices in the MPLS network and the legacy VPWS cross-connects are up and running between PE1, PE2, and PE3.

- PE1 and PE2 have a legacy PW established between them. (pw1)
- PE1 and PE3 have a legacy PW established between them. (pw2)

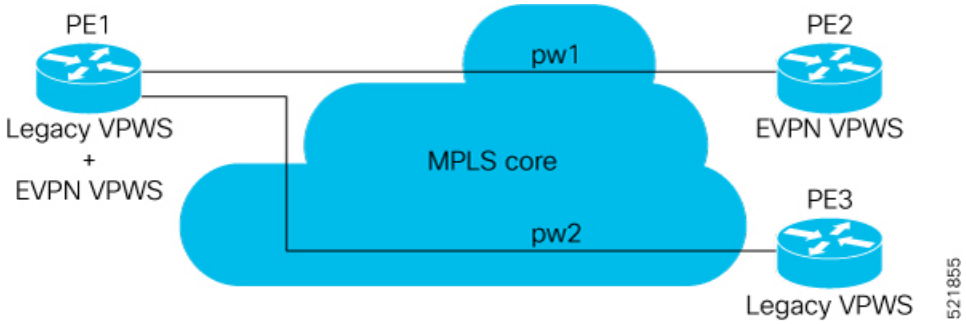
  Service provider wants to replace PE1 with a new hardware. So after replacing the equipment, service provider enables EVPN-VPWS on PE1 first.



Let's understand what happens when only PE1 is migrating to EVPN-VPWS:

- When EVPN-VPWS is enabled, PE1 starts advertising EVPN EVI or Ethernet-AD route to other PE nodes.
- PE1 advertises BGP VPWS Auto-Discovery route and the BGP EVPN Ethernet-AD per EVI route for a given PW.
- As PE2 and PE3 aren't yet migrated, PE1 does not receive any EVI/EAD routes from these PE nodes. Therefore, legacy VPWS runs between PE1, PE2, and PE3.
- PE1 keeps forwarding traffic using legacy VPWS.

After one year, service provider decides to upgrade PE2 and wants to migrate from VPWS to EVPN-VPWS.



- When the upgrade is completed, PE2 starts advertising EVI/EAD route to other PE nodes.
- Both PE1 and PE2 discover each other through EVPN routes.
- As a result, EVPN-VPWS service replaces legacy VPWS service between PE1 and PE2. This is called EVPN-VPWS MPLS Seamless Integration with VPWS.
- EVPN-VPWS service takes high-precedence over legacy VPWS network.
- PE1 and PE2 shuts down the legacy VPWS between them to prevent ongoing duplicate packets from remote CE.

Service provider plans not to migrate PE3 device as of now:

- At this stage, PE1 keeps running legacy VPWS service with PE3.
- The legacy VPWS to EVPN-VPWS migration then continues to remaining PE nodes. The legacy VPWS and EVPN-VPWS dual-stack coexist in the core for a given L2 Attachment Circuit (AC).

After another year, service provider plans to upgrade the PE3 device.

- PE3 is now enabled with EVPN-VPWS service.
- All the PE devices are replaced with EVPN-VPWS services in the network.
- Service provider plans to retain both legacy and an EVPN-VPWS related configuration on PE1 and PE2 nodes.
- During any uncertainties, service provider can roll back the migration. If you rollback the migration to VPWS at node PE2, then PE1 and PE2 will revert to the legacy VPWS between them.

Restriction

- Supported only in single-homing or EVPN port-active multi-homing.
- PWHE is not supported.

Configuration Example

To enable the feature, use the **vpws-seamless-integration** command.

In this example, let's see how to migrate each PE at a time.

When you migrate only PE1, here is the configuration example for PE1, PE2, and PE3:

```
/* Here is the configuration for PE1: */
Router# configure
Router(config)# l2vpn xconnect group 1
Router(config-l2vpn-xc)# mp2mp 2
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3

/* Migrate VPWS to EVPN-VPWS*/
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# root
Router(config)# l2vpn xconnect group 2
Router(config-l2vpn-xc)# p2p 3
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 5
Router(config-l2vpn-xc-p2p-pw)# commit

/* Here is the configuration for PE2: */
Router# configure
Router(config)# l2vpn xconnect group 1
Router(config-l2vpn-xc)# mp2mp 2
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# exit
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 5
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

```
/* Here is the configuration for PE3:*/
Router# configure
Router(config)# l2vpn xconnect group 1
Router(config-l2vpn-xc)# mp2mp 2
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# exit
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 5
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

The following show output indicates that only VPWS is up and EVPN is down:

```
Router# show l2vpn xconnect
Tue Jun  8 12:36:20.253 EDT
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed,
        LU = Local Up, RU = Remote Up, CO = Connected, (SI) = Seamless Inactive

XConnect                   Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description            ST
------------------------   ----------------------------   ------------------------------
service-8  evpn-vpws-8
                      DN   BE1.1                  UP       EVPN 8,8,192.168.0.4   DN
------------------------------------------------------------------------------------------
service-8  mp2mp-8.8:10008
                      UP   BE1.1                  UP       192.168.0.4     534296 UP
------------------------------------------------------------------------------------------
```

When you migrate both PE1 and PE2, here is the configuration example for PE1, PE2, and PE3:

```
/* Here is the configuration for PE1: */
Router# configure
Router(config)# l2vpn xconnect group 1
Router(config-l2vpn-xc)# mp2mp 2
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
/* Migrate VPWS to EVPN-VPWS*\
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# root
Router(config)# l2vpn xconnect group 2
Router(config-l2vpn-xc)# p2p 3
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 5
Router(config-l2vpn-xc-p2p-pw)# commit

/* Here is the configuration for PE2: */
Router# configure
Router(config)# l2vpn xconnect group 1
Router(config-l2vpn-xc)# mp2mp 2
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
/* Migrate VPWS to EVPN-VPWS*/
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

```
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# root
Router(config)# l2vpn xconnect group 2
Router(config-l2vpn-xc)# p2p 3
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 5
Router(config-l2vpn-xc-p2p-pw)# commit

/* Here is the configuration for PE3: */
Router# configure
Router(config)# l2vpn xconnect group 1
Router(config-l2vpn-xc)# mp2mp 2
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# exit
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 5
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

Verification

The following example shows that VPWS is inactive and indicates the status as SB(SI).

```
Router# show l2vpn xconnect
Thu Feb 25 11:57:27.622 EST
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed,
        LU = Local Up, RU = Remote Up, CO = Connected, (SI) = Seamless Inactive

XConnect                    Segment 1                       Segment 2
Group      Name       ST    Description            ST       Description            ST
-------------------------   ----------------------------   ------------------------------
evpn-vpws  test11-1   UP    BE11                   UP       EVPN 11,11,24048       UP
-----------------------------------------------------------------------------------------
legacy-tldp
           test11     DN    BE11                   SB(SI)   192.168.12.110  11     UP
-----------------------------------------------------------------------------------------
```

The following example shows whether EVPN-VPWS or VPWS is used for forwarding the traffic. In this example, `evi: 1` indicates that EVPN is used for forwarding the traffic.

```
Router# show l2vpn forwarding interface gigabitEthernet 0/2/0/8.1 detail location 0/2/CPU0
Wed Apr 28 09:08:37.512 EDT
Local interface: GigabitEthernet0/2/0/8.1, Xconnect id: 0x800001, Status: up
  Segment 1
    AC, GigabitEthernet0/2/0/8.1, status: Bound
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
  Segment 2
    MPLS, Destination address: 192.168.0.4, evi: 1,
ac-id: 1, status: Bound
Pseudowire label: 24004
    Control word enabled
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
```

In the following example, `pw-id: 1` indicates that VPWS is used for forwarding the traffic:

```
Router# show l2vpn forwarding interface gigabitEthernet 0/2/0/8.1 detail location 0/2/CPU0
Wed Apr 28 09:09:45.204 EDT
Local interface: GigabitEthernet0/2/0/8.1, Xconnect id: 0x800001, Status: up
  Segment 1
    AC, GigabitEthernet0/2/0/8.1, status: Bound
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
  Segment 2
    MPLS, Destination address: 192.168.0.4, pw-id: 1, status: Bound
Pseudowire label: 24000
    Control word disabled
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
```

Use the **l2vpn logging pseudowire** command to track the migration of AC from one PW to another.

For example,

```
Router(config)# l2vpn logging pseudowire
RP/0/0/CPU0:Jan 18 15:35:15.607 EST:
l2vpn_mgr[1234]: %L2-EVPN-5-VPWS_SEAMLESS_INTEGRATION_STATE_CHANGE :
GigabitEthernet0/2/0/8.1 - Active XC is now service-1:evpn-vpws-1, standby XC is
service-1:tldp-1
```

TLDP PW to EVPN-VPWS Migration

Similar to migrating VPWS to EVPN, we can migrate TLDP PW to EVPN-VPWS on all the PE routers incrementally.

You can perform this task on all the PE router incrementaly. The following configuration example shows the TLDP PW to EVPN-VPWS migration on PE1:

```
/*Here is an example using TLDP*/
Router# configure
Router(config)# l2vpn xconnect group 1
Router(config-l2vpn-xc)# p2p p1
Router(config-l2vpn-xc-p2p)# interface BE1.1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p)# vpws-seamless-integration
```

Network Convergence using Core Isolation Protection

The Network Convergence using Core Isolation Protection feature allows the router to converge fast when remote links and local interfaces fail. This feature reduces the duration of traffic drop by rapidly rerouting traffic to alternate paths. This feature uses Object Tracking (OT) to detect remote link failure and failure of connected interfaces.

Tracking interfaces can only detect failure of connected interfaces and not failure of a remote router interfaces that provides connectivity to the core. Tracking one or more BGP neighbor sessions along with one or more of the neighbor’s address-families enables you to detect remote link failure.

Object Tracking

Object tracking (OT) is a mechanism for tracking an object to take any client action on another object as configured by the client. The object on which the client action is performed may not have any relationship to the tracked objects. The client actions are performed based on changes to the properties of the object being tracked.

You can identify each tracked object by a unique name that is specified by the track command in the configuration mode.

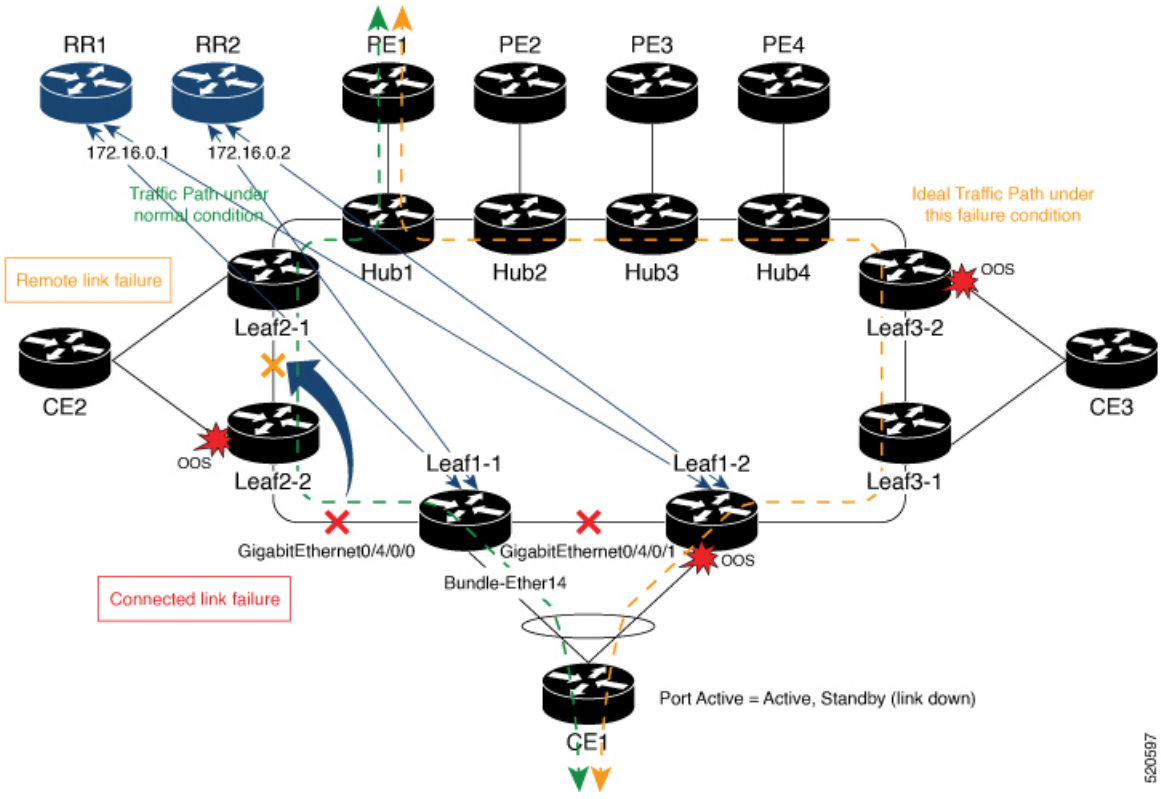
The tracking process receives the notification when the tracked object changes its state. The state of the tracked objects can be up or down.

You can also track multiple objects by a list. You can use a flexible method for combining objects with Boolean logic. This functionality includes:

- Boolean AND function—When a tracked list has been assigned a Boolean AND function, each object defined within a subset must be in an up state, so that the tracked object can also be in the up state.
- Boolean OR function—When the tracked list has been assigned a Boolean OR function, it means that at least one object defined within a subset must also be in an up state, so that the tracked object can also be in the up state.

For more information on OT, see the *Configuring Object Tracking* chapter in the *System Management Configuration Guide for Cisco NCS 560 Series Routers*.

Figure 47: EVPN Convergence Using Core Isolation Protection



Consider a traffic flow from CE1 to PE1. The CE1 can send the traffic either from Leaf1-1 or Leaf1-2. When Leaf1-1 loses the connectivity to both the local links and remote link, BGP sessions to both route reflectors

(RRs) are down; the Leaf1-1 brings down the Bundle-Ether14 connected to CE1. The CE1 redirects the traffic from Leaf1-2 to PE1.

You can track the connected interfaces to identify the connected link failures. However, if there is a remote link failure, tracking connected interfaces does not identify the remote link failures. You must track BGP sessions to identify the remote link failure.

Note When you configure the **bgp graceful-restart** command, unconfiguring a neighbor is considered as a non-gr event. This generates a BGP notification to the neighbor before the neighbor is unconfigured.

On the remote router, if the track is configured for this neighbor, the track state is brought down immediately.

However, certain configurations are treated as graceful reset reason and when unconfigured they supress the BGP notification to the neighbor. The route-reflector-client configuration under the neighbor or neighbor address-family is one of the examples.

On the remote router, if the track is configured for this neighbor, the track state is not brought down immediately because a notification is not received.

To overcome this situation, shutdown the neighbor before unconfiguring the neighbor. This generates a BGP notification to the neighbor, and any track configured for the neighbor is brought down immediately.

Configure EVPN Convergence using Core Isolation Protection

A tracked list contains one or more objects. The Boolean expression enables tracking objects using either AND or OR operators. For example, when tracking two interfaces, using the AND operator, up means that *both* interfaces are up, and down means that *either* interface is down.

Note An object must exist before it can be added to a tracked list.

The NOT operator is specified for one or more objects and negates the state of the object.

After configuring the tracked object, you must associate the neighbor or interface whose state must be tracked.

Perform the following tasks to configure EVPN convergence using core isolation protection:

- Configure BGP
- Track the Line Protocol State of an Interface
- Track neighbor adress-family state
- Track objects for both interfaces and neighbors

Configuration Example

In this example, Leaf1-1 brings the down the AC connected to CE1 when:

Both local interfaces GigabitEthernet0/4/0/0 and GigabitEthernet0/4/0/1of Leaf1-1 are down.

OR

Leaf1-1 BGP sessions to both RRs are down.

CE1 re-directs the traffic it was sending to Leaf1-1 to Leaf1-2.

Perform the following tasks on Leaf1-1:

```
/* Configure BGP */
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# commit

/* Track the Line Protocol State of an Interface */
Router# configure
Router(config)# track interface-1
Router(config-track)# type line-protocol state
Router(config-track-line-prot)# interface GigabitEthernet0/4/0/0
Router(config-track-line-prot)#exit
Router(config-track)#exit
Router(config)# track interface-2
Router(config-track)# type line-protocol state
Router(config-track-line-prot)# interface GigabitEthernet0/4/0/1
Router(config-track-line-prot)#exit
Router(config-track)#exit
Router(config)# track interface-group-1
Router(config-track)# type list boolean or
Router(config-track-list-boolean)# object interface-1
Router(config-track-list-boolean)# object interface-2
Router(config-track-list-boolean)# commit

/* Track neighbor address-family state */
Router# configure
Router(config)# track neighbor-A
Router(config-track)# type bgp neighbor address-family state
Router(config-track-bgp-nbr-af)# address-family l2vpn evpn
Router(config-track-bgp-neighbor)# neighbor 172.16.0.1
Router(config-track-bgp-neighbor)# exit
Router(config-track-bgp-nbr-af)# exit
Router(config-track)# exit
Router(config)# track neighbor-B
Router(config-track)# type bgp neighbor address-family state
Router(config-track-bgp-nbr-af)# address-family l2vpn evpn
Router(config-track-bgp-neighbor)# neighbor 172.16.0.2
Router(config-track-bgp-neighbor)# exit
Router(config-track-bgp-nbr-af)# exit
Router(config-track)# exit
Router(config)# track neighbor-group-1
Router(config-track)# type list boolean or
Router(config-track-list-boolean)# object neighbor-A
Router(config-track-list-boolean)# object neighbor-B
Router(config-track-list-boolean)# commit

/* Track objects for both interfaces and neighbors */
Router# configure
Router(config)# track core-group-1
Router(config-track)# type list boolean and
Router(config-track-list-boolean)# object neighbor-group-1
```

```
Router(config-track-list-boolean)# object interface-group-1
Router(config-track-list-boolean)# action
Router(config-track-action)# track-down error-disable interface Bundle-Ether14 auto-recover
Router(config-track-action)# commit
```

Running Configuration

This section shows EVPN convergence using core isolation protection running configuration.

```
router bgp 100
 address-family l2vpn evpn
 !
 neighbor 172.16.0.1
  remote-as 100
  address-family l2vpn evpn
  !
 !
 neighbor 172.16.0.2
  remote-as 100
  address-family l2vpn evpn
  !
 !
!

track interface-1
 type line-protocol state
  interface GigabitEthernet0/4/0/0
 !
!
track interface-2
 type line-protocol state
  interface GigabitEthernet0/4/0/1
 !
!
track interface-group-1
 type list boolean or
  object interface-1
  object interface-2
 !
!

track neighbor-A
 type bgp neighbor address-family state
  address-family l2vpn evpn
   neighbor 172.16.0.1
  !
 !
!
track neighbor-B
 type bgp neighbor address-family state
  address-family l2vpn evpn
   neighbor 172.16.0.1
  !
 !
!

track neighbor-group-1
 type list boolean or
  object neighbor-A
  object neighbor-B
  !
 !
```

```
!
track core-group-1
 type list boolean and
  object neighbor-group-1
  object interface-group-1
 !
 action
  track-down error-disable interface Bundle-Ether14 auto-recover
  !
 !
```

Verification

Verify that you have configured the EVPN convergence using core isolation protection feature successfully.

```
Router# show track
Wed May 27 04:42:11.995 UTC

Track neighbor-A
        BGP Neighbor AF L2VPN EVPN NBR 172.16.0.1 vrf default
        Reachability is UP
                Neighbor Address Reachablity is Up
                BGP Neighbor Address-family state is Up
        4 changes, last change UTC Tue May 26 2020 20:14:33.171

Track neighbor-B
        BGP Neighbor AF L2VPN EVPN NBR 172.16.0.2 vrf default
        Reachability is UP
                Neighbor Address Reachablity is Up
                BGP Neighbor Address-family state is Up
        4 changes, last change UTC Tue May 26 2020 20:14:27.527

Track core-group-1
        List boolean and is UP
        2 changes, last change 20:14:27 UTC Tue May 26 2020
                object interface-group-1 UP
                object neighbor-group-1 UP

Track interface-1
        Interface GigabitEthernet0/4/0/0 line-protocol
        Line protocol is UP
        2 changes, last change 20:13:32 UTC Tue May 26 2020

Track interface-2
        Interface GigabitEthernet0/4/0/1 line-protocol
        Line protocol is UP
        2 changes, last change 20:13:28 UTC Tue May 26 2020

Track interface-group-1
        List boolean or is UP
        2 changes, last change 20:13:28 UTC Tue May 26 2020
                object interface-2 UP
                object interface-1 UP

Track neighbor-group-1
        List boolean or is UP
        2 changes, last change 20:14:27 UTC Tue May 26 2020
                object neighbor-A UP
                object neighbor-B UP


Router# show track brief
Wed May 27 04:39:19.740 UTC
```

```
Track                          Object                                    Parameter
   Value
-------------------------------------------------------------------------------------------------------
neighbor-A                     bgp nbr L2VPN EVPN 172.16.0.1 vrf defau reachability
Up
neighbor-B                     bgp nbr L2VPN EVPN 172.16.0.1 vrf defau reachability
Up
core-group-1                   list                                      boolean and
  Up
interface-1                    interface GigabitEthernet0/4/0/0          line protocol
  Up
interface-2                    interface GigabitEthernet0/4/0/1          line protocol
  Up
interface-group-1              list                                      boolean or
  Up
neighbor-group-1               list                                      boolean or
  Up
-------------------------------------------------------------------------------------------------------
```

```
Router# show bgp track
Wed May 27 05:05:51.285 UTC

VRF                   Address-family       Neighbor         Status     Flags

default               L2VPN EVPN           172.16.0.1     UP         0x01
default               L2VPN EVPN           172.16.0.2     UP         0x01

Processed 2 entries
```

Configurable Recovery Time for EVPN Core Isolation Group

Table 26: Feature History Table

Feature Name	Release Information	Feature Description
Configurable Recovery Time for EVPN Core Isolation Group	Release 7.6.1	You can now configure the recovery time for the EVPN core isolation group after the core interfaces recover from a network failure. This functionality is important because post-failure recovery, you can provide sufficient time for the EVPN PE nodes to relearn the MAC addresses and BGP routes received from the remote PEs. There's also time to handle delays in exchanging EVPN routes after recovery. This feature introduces the **core-de-isolation** command under the EVPN Timers configuration mode.

When the core link failure is detected on the PE device, the PE device is isolated from the network and brings down the access interfaces connected to this PE till the core interfaces recover. When the core links recover, the default recovery delay timer begins. The access interfaces become active after the default recover delay timer of 60 seconds expire. The core isolation group recovery delay timer was not user-configurable.

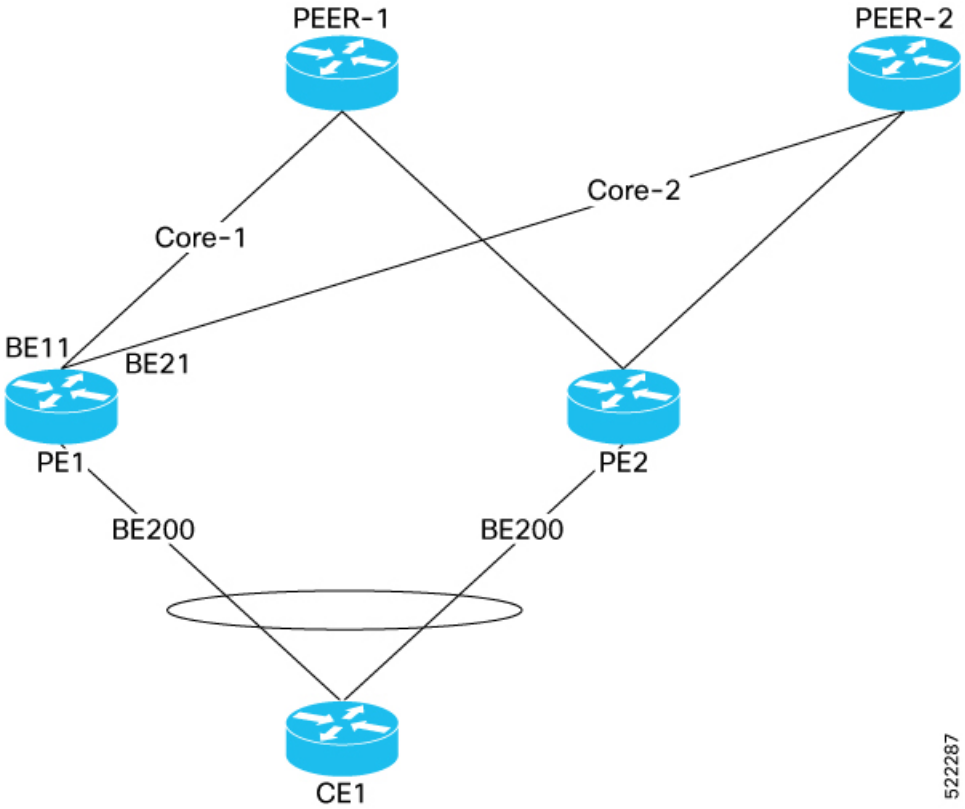
Under scale situations where a network has high MAC addresses, it is observed that the 60 seconds is too short to bring up the access bundle interface as there can be multiple reasons which can delay the exchange of EVPN routes even after the core interfaces have come up.

This feature allows you to configure the core isolation group recovery time to handle delays coming from the core and provides enough time for the EVPN PE nodes to relearn the MAC addresses. You can configure the core isolation group recovery time using the **core-de-isolation** command.

Topology

Consider a topology where CE1 is connected to PE1 and PE2. PE1 and PE2 are running EVPN over the MPLS core network. The core interfaces on PE1 are configured with BE11 and BE22. When the core links of PE1 go down, the EVPN detects the link failure and isolates the PE1 node from the core network, and brings down the access interfaces connected to PE1. This prevents CE1 from sending any traffic to PE1.

When all the core interfaces and BGP sessions come up, PE1 advertises Ethernet A-D Ethernet Segment (ES-EAD) routes again, triggers the service carving, and becomes part of the core network. The access interfaces connected to PE1 from CE1 also come up after the *core-de-isolation* timer value expires.



Configurable Recovery Time for EVPN Core Isolation Group

To enable this feature, configure core interfaces under the EVPN group and associate that group to the Ethernet Segment which is an attachment circuit (AC) attached to the CE.

Perform the following tasks to configure recovery time for EVPN core isolation group:

- Configure EVPN core interfaces on PE1
- Configure *core-de-isolation* timer on PE1
- Configure attachment circuits on CE1

Configuration Example

Configure EVPN core interfaces on PE1.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# group 100
Router(config-evpn-group)# core interface BE11
Router(config-evpn-group)# core interface BE21
Router(config-evpn-group)# commit
```

Configure core-de-isolation timer on PE1.

```
Router# configure
Router(config)# evpn timers
Router(config-evpn-timers)# core-de-isolation 120
Router(config-evpn-timers)# commit
```

Configure attachment circuits on CE1.

```
/* Configure interface Bundle-Ether200 and associate it to core isolation group 100 */
Router # configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether200
Router(config-evpn-ac)# ethernet-segment identifier type 0 11.11.11.11.11.11.11.11.11
Router(config-evpn-ac-es)# bgp route-target 1111.1111.1111
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# core-isolation-group 100

/* Configure interface Bundle-Ether201 and associate it to core isolation group 100 */
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether201
Router(config-evpn-ac)# ethernet-segment identifier type 0 11.22.22.22.22.22.22.22.22
Router(config-evpn-ac-es)# bgp route-target 1111.2222.2222
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# core-isolation-group 100
```

Running Configuration

This section shows the EVPN core isolation group recovery delay timer running configuration.

```
/* Configure EVPN core interfaces on PE1 */
evpn
 group 100
  core interface Bundle-Ether11
  core interface Bundle-Ether21
!
!
/* Configure core-de-isolation timer on PE1 */
evpn timers
  core-de-isolation 120
!
!
/* Configure attachment circuits on CE1 */
evpn
 interface Bundle-Ether200
  ethernet-segment
   identifier type 0 11.11.11.11.11.11.11.11.11
   bgp route-target 1111.1111.1111
```

```
  !
  core-isolation-group 100
!
!
evpn
 interface Bundle-Ether201
  ethernet-segment
   identifier type 0 11.22.22.22.22.22.22.22.22
   bgp route-target 1111.2222.2222
  !
  core-isolation-group 100
!
!
```

Verification

The following output shows that all core interfaces and access interfaces are UP. The *core de-isolation* timer value is configured as 120 seconds, but not running as the core interfaces are UP.

```
Router# show evpn group
EVPN Group: 100

  state: Ready

  Core Interfaces:
     Bundle-Ether11: up
     Bundle-Ether21: up

  Access Interfaces:
     Bundle-Ether200: up
     Bundle-Ether201: up

Router# show evpn summary
-----------------------------
Global Information
-----------------------------
Number of EVIs                    : 141
Number of TEPs                    : 2
Number of Local EAD Entries       : 178
Number of Remote EAD Entries      : 534
Number of Local MAC Routes        : 89
          MAC                     : 89
          MAC-IPv4                : 0
          MAC-IPv6                : 0
Number of Local ES:Global MAC     : 1
Number of Remote MAC Routes       : 0
          MAC                     : 0
          MAC-IPv4                : 0
          MAC-IPv6                : 0
Number of Remote SYNC MAC Routes  : 0
Number of Local IMCAST Routes     : 89
Number of Remote IMCAST Routes    : 178
Number of Internal Labels         : 178
Number of single-home Internal IDs : 0
Number of multi-home Internal IDs  : 0
Number of ES Entries              : 3
Number of Neighbor Entries        : 178
EVPN Router ID                    : 192.168.10.1
BGP ASN                           : 64600
PBB BSA MAC address               : d46a.3599.50d8
Global peering timer              :      3 seconds
Global recovery timer             :     30 seconds
```

```
Global carving timer               :      0 seconds
Global MAC postpone timer          :    300 seconds [not running]
Global core de-isolation timer     :    120 seconds [not running]
EVPN services costed out on node   : No
          Startup-cost-in timer    : Not configured
          EVPN manual cost-out     : No
          EVPN Bundle Convergence  : No
```

Failure Scenario

The following example shows the failure scenario and how the *core de-isolation* timer works.

Let's bring down the core interfaces:

```
Router# configure
Router(config)# interface Bundle-Ether11
Router(config-if)# shutdown
Router(config-if)# exit
Router(config)# interface Bundle-Ether21
Router(config-if)# shutdown
Router(config-if)# commit
```

This example shows when the core interfaces are shutdown even the access interfaces are down and the core is isolated.

```
Router# show evpn group

EVPN Group: 100

  state: Isolated

  Core Interfaces:
     Bundle-Ether11: shutdown
     Bundle-Ether21: shutdown

  Access Interfaces:
     Bundle-Ether200: down
     Bundle-Ether201: down
```

This example shows that the *core de-isolation timer* is not yet running because the core interfaces are still down.

```
Router# show evpn summary
-----------------------------
Global Information
-----------------------------
Number of EVIs                     : 141
Number of TEPs                     : 0
Number of Local EAD Entries        : 178
Number of Remote EAD Entries       : 0
Number of Local MAC Routes         : 89
          MAC                      : 89
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Local ES:Global MAC      : 1
Number of Remote MAC Routes        : 0
          MAC                      : 0
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Remote SYNC MAC Routes   : 0
Number of Local IMCAST Routes      : 89
```

```
Number of Remote IMCAST Routes     : 0
Number of Internal Labels          : 0
Number of single-home Internal IDs : 0
Number of multi-home Internal IDs  : 0
Number of ES Entries               : 3
Number of Neighbor Entries         : 0
EVPN Router ID                     : 192.168.10.1
BGP ASN                            : 64600
PBB BSA MAC address                : d46a.3599.50d8
Global peering timer               :      3 seconds
Global recovery timer              :     30 seconds
Global carving timer               :      0 seconds
Global MAC postpone timer          :    300 seconds [not running]
Global core de-isolation timer     :    120 seconds [not running]
EVPN services costed out on node   : No
          Startup-cost-in timer    : Not configured
          EVPN manual cost-out     : No
          EVPN Bundle Convergence  : No
```

Let's bring up the core interfaces and see how the *core de-isolation* timer starts.

```
Router# rollback configuration last 1

Loading Rollback Changes.
Loaded Rollback Changes in 1 sec
Committing.
6 items committed in 1 sec (5)items/sec
Updating.
Updated Commit database in 1 sec
Configuration successfully rolled back 1 commits.
```

In this example, you can see that the *core de-isolation* timer starts running after the core interfaces come up. When the core interfaces are UP, the state of core changes to Deisolating. In the following output you can see the state as Deisolating and core interfaces are up and the *core de-isolation* timer has started.

The access interfaces come up only after the *core de-isolation* timer value expires. In the following output you can see the access interfaces are still down.

```
Router# show evpn group

EVPN Group: 100

  state: Deisolating

  Core Interfaces:
     Bundle-Ether11: up
     Bundle-Ether21: up

  Access Interfaces:
     Bundle-Ether200: down
     Bundle-Ether201: down

Router# show evpn summary
------------------------------
Global Information
------------------------------
Number of EVIs                 : 141
Number of TEPs                 : 2
Number of Local EAD Entries    : 178
Number of Remote EAD Entries   : 534
Number of Local MAC Routes     : 89
          MAC                  : 89
```

```
          MAC-IPv4                : 0
          MAC-IPv6                : 0
Number of Local ES:Global MAC     : 1
Number of Remote MAC Routes       : 0
          MAC                     : 0
          MAC-IPv4                : 0
          MAC-IPv6                : 0
Number of Remote SYNC MAC Routes  : 0
Number of Local IMCAST Routes     : 89
Number of Remote IMCAST Routes    : 178
Number of Internal Labels         : 178
Number of single-home Internal IDs : 0
Number of multi-home Internal IDs  : 0
Number of ES Entries              : 3
Number of Neighbor Entries        : 178
EVPN Router ID                    : 192.168.10.1
BGP ASN                           : 64600
PBB BSA MAC address               : d46a.3599.50d8
Global peering timer              :      3 seconds
Global recovery timer             :     30 seconds
Global carving timer              :      0 seconds
Global MAC postpone timer         :    300 seconds [not running]
Global core de-isolation timer    :    120 seconds [running, 14.6 sec left]
EVPN services costed out on node  : No
          Startup-cost-in timer   : Not configured
          EVPN manual cost-out    : No
          EVPN Bundle Convergence : No
```

The following output shows that the *core de-isolation* timer has expired.

```
Router# show evpn summary
----------------------------
Global Information
----------------------------
Number of EVIs                    : 141
Number of TEPs                    : 2
Number of Local EAD Entries       : 178
Number of Remote EAD Entries      : 534
Number of Local MAC Routes        : 89
          MAC                     : 89
          MAC-IPv4                : 0
          MAC-IPv6                : 0
Number of Local ES:Global MAC     : 1
Number of Remote MAC Routes       : 0
          MAC                     : 0
          MAC-IPv4                : 0
          MAC-IPv6                : 0
Number of Remote SYNC MAC Routes  : 0
Number of Local IMCAST Routes     : 89
Number of Remote IMCAST Routes    : 178
Number of Internal Labels         : 178
Number of single-home Internal IDs : 0
Number of multi-home Internal IDs  : 0
Number of ES Entries              : 3
Number of Neighbor Entries        : 178
EVPN Router ID                    : 192.168.10.1
BGP ASN                           : 64600
PBB BSA MAC address               : d46a.3599.50d8
Global peering timer              :      3 seconds
Global recovery timer             :     30 seconds
Global carving timer              :      0 seconds
Global MAC postpone timer         :    300 seconds [not running]
Global core de-isolation timer    :    120 seconds [not running]
EVPN services costed out on node  : No
```

```
          Startup-cost-in timer    : Not configured
          EVPN manual cost-out     : No
          EVPN Bundle Convergence  : No
```

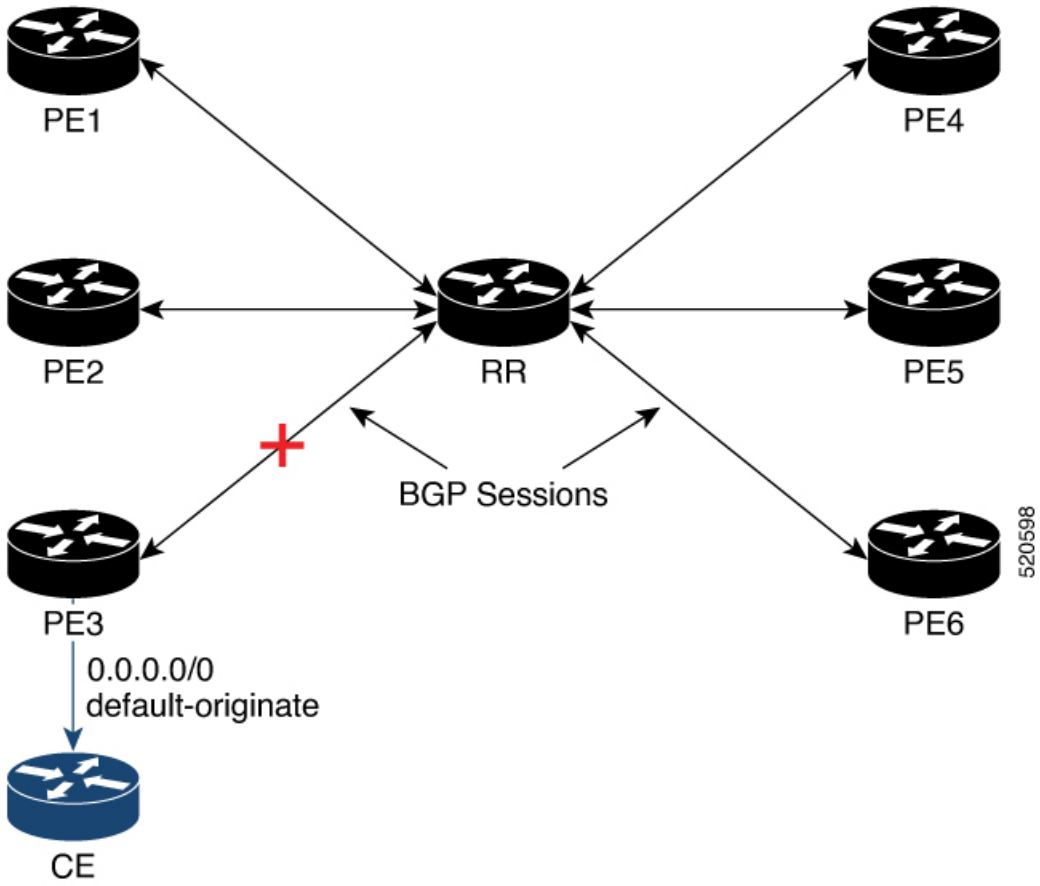
After the *core de-isolation* timer expires, you can see that the state is Ready, and both core and access interfaces are UP.

```
Router# show evpn group

EVPN Group: 100

  state: Ready

  Core Interfaces:
     Bundle-Ether11: up
     Bundle-Ether21: up

  Access Interfaces:
     Bundle-Ether200: up
     Bundle-Ether201: up
```

Conditional Advertisement of Default-Originate

The router advertises the default-originate (0.0.0.0/0) towards the network fabric only upon receiving all the core routes. The router withdraws the advertisement of default-originate when the core is isolated. To avoid traffic drop, install the routes in the hardware. To accommodate an additional delay for the routes to be installed in the hardware, you can configure a timeout for the installed routes.

Figure 48: Advertisement of default-originate



In this topology, PE3 advertises the default-originate to CE only when the PE3 session to RR is established and all the routes are received from the RR.

Configure Conditional Advertisement of Default-Originate

Perform the following tasks to configure conditional advertisement of default-originate.

- Configure BGP
- Configure RPL
- Track BGP neighbor address-family state

Configuration Example

Perform the following task on PE3:

```
/* Configure BGP */
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 192.0.2.1
Router(config-bgp)# address-family vpnv4 unicast
```

```
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 172.16.0.5
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# address-family vpnv4 unicast
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# vrf cust1
Router(config-bgp-vrf)# rd auto
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# redistribute connected
Router(config-bgp-vrf-af)# redistribute static
Router(config-bgp-vrf-af)# exit
Router(config-bgp-vrf)# neighbor 172.16.0.5
Router(config-bgp-vrf-nbr)# remote-as 200
Router(config-bgp-vrf-nbr)# address-family ipv4 unicast
Router(config-bgp-vrf-nbr-af)# default-originate route-policy track-bgp-core-policy
Router(config-bgp-vrf-nbr-af)# route-policy pass in
Router(config-bgp-vrf-nbr-af)# route-policy pass out
Router(config-bgp-vrf-nbr-af) commit


/* Configure RPL */
Router# configure
Router(config)# route-policy track-bgp-core-policy
Router(config-rpl)# if track core-group-1 is up then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
Router(config)# commit


/* Track BGP neighbor address-family state */
Router# configure
Router(config)# track core-group-1
Router(config-track)# type bgp neighbor address-family state
Router(config-track-bgp-nbr-af)# address-family vpnv4 unicast
Router(config-track-bgp-neighbor)# neighbor 172.16.0.5
Router(config-track-bgp-neighbor)# commit
```

Running Configuration

This section shows conditional advertisement of default-originate running configuration.

```
configure
 router bgp 100
  bgp router-id 192.0.2.1
  address-family vpnv4 unicast
!
 neighbor 172.16.0.5
  remote-as 200
  address-family vpnv4 unicast
!

 vrf cust1
  rd auto
  address-family ipv4 unicast
   redistribute connected
   redistribute static
!
 neighbor 172.16.0.5
  remote-as 200
  address-family ipv4 unicast
   default-originate route-policy track-bgp-core-policy
```

```
   route-policy pass in
   route-policy pass out
!

route-policy track-bgp-core-policy
 if track core-group-1 is up then
  pass
  endif
 end-policy
!
track network-core
 type bgp neighbor address-family state
  address-family vpnv4 unicast
   neighbor 172.16.0.5
!
```

Verification

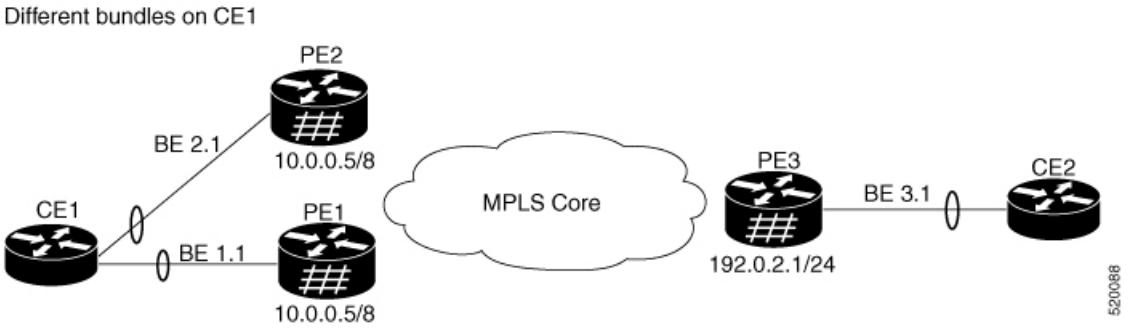
Verify conditional advertisement of default-originate.

```
Router# show rpl active route-policy
Wed May 27 06:54:31.902 UTC

ACTIVE -- Referenced by at least one policy which is attached
INACTIVE -- Only referenced by policies which are not attached
UNUSED -- Not attached (directly or indirectly) and not referenced

The following policies are (ACTIVE)
-----------------------------------------
    track-bgp-core
-----------------------------------------
Router# show rpl route-policy track-bgp-core-policy
Wed May 27 06:54:38.090 UTC
route-policy track-bgp-core-policy
  if track core-group-1 is up then
    pass
  endif
end-policy
!

Router# show bgp policy route-policy track-bgp-core-policy summary
Wed May 27 06:54:42.823 UTC
Network            Next Hop        From            Advertised to
0.0.0.0/0          0.0.0.0         Local           172.16.0.5


Router# show bgp neighbor 172.16.0.5
Wed May 27 06:55:39.535 UTC

BGP neighbor is 172.16.0.5
 Remote AS 9730, local AS 9730, internal link
 Remote router ID 172.16.0.5
  BGP state = Established, up for 10:41:12
[snip]
 For Address Family: IPv4 Unicast
  BGP neighbor version 2
  Update group: 0.4 Filter-group: 0.1  No Refresh request being processed
  Default information originate: default route-policy track-bgp-core-policy, default sent
  AF-dependent capabilities:
[snip]
  Track Enabled, Status UP, Nbr GR state Not Enabled, EOR tmr Not Running
  Advertise routes with local-label via Unicast SAFI
```

EVPN Single-Active Multihoming for Anycast Gateway IRB

The EVPN Single-Active Multihoming for Anycast Gateway IRB feature supports single-active redundancy mode. In this mode, the provider edge (PE) nodes locally connected to an Ethernet Segment load balance traffic to and from the Ethernet Segment based on EVPN service instance (EVI). Within an EVPN service instance, only one PE forwards traffic to and from the Ethernet Segment (ES). This feature supports intersubnet scenario only.

Figure 49: EVPN: Single-Active Multihoming for Anycast Gateway IRB



Consider a topology where CE1 is multihomed to PE1 and PE2. Bundle Ethernet interfaces BE 1.1, BE 2.1, and the ingress interface must belong to the same switching domain on CE1. Enable host routing and configure anycast gateway IP address on both these peering PEs. PE1 and PE2 are connected to PE3 through MPLS core. PE3 has reachability of subnet 10.0.0.5/8 to both peering PEs. Peering PEs has reachability to PE3 subnet 192.0.2.1/24. CE2 is connected to PE3 through an Ethernet interface bundle. PE1 and PE2 advertise Type 4 routes, and then performs designated forwarder (DF) election. The non-DF blocks the traffic in both the directions in single-active mode.

Consider a traffic flow from CE1 to CE2. CE1 sends an address resolution protocol (ARP) broadcast request to both PE1 and PE2. Peering PEs performs designated forwarder (DF) election for shared ESI. If PE1 is the designated forwarder for the EVI, PE1 replies to the ARP request from CE1. PE2 drops the traffic from CE1. Thereafter, all the unicast traffic is sent through PE1. PE2 is set to stand-by or blocked state and traffic is not sent over this path. PE1 advertises MAC to PE3. PE3 always sends and receives traffic through PE1. PE3 sends the traffic to CE2 over Ethernet interface bundle. If BE1 fails, PE2 becomes the DF and traffic flows through PE2.

Configure EVPN Single-Active Multihoming

Perform the following tasks on PE1 and PE2 to configure EVPN Single-Active Multihoming feature:

- Configure EVPN IRB with host routing
- Configure EVPN Ethernet Segment
- Configure Layer 2 Interface
- Configure a Bridge Domain
- Configure VRF

Configure EVPN Ethernet Segment

Perform this task to configure the EVPN Ethernet segment.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# bgp route-target 4000.0000.0001
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
configure
evpn
 interface Bundle-Ether1
   ethernet-segment
    identifier type 0 40.00.00.00.00.00.00.00.01
    load-balancing-mode single-active
    bgp route-target 4000.0000.0001
    !
!
!
```

Configure EVPN Service Instance (EVI) Parameters

Perform this task to define EVPN service instance (EVI) parameters.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 6005
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# rd 200:50
Router(config-evpn-evi-bgp)# route-target import 100:6005
Router(config-evpn-evi-bgp)# route-target export 100:6005
Router(config-evpn-evi-bgp)# commit
```

Running Configuration

```
configure
 evpn
  evi 6005
   bgp
    rd 200:50
    route-target import 100:6005
    route-target export 100:6005
!
!
```

Configure Layer 2 Interface

Perform this task to define Layer 2 interface.

```
Router# configure
Router(config)# interface bundle-ether2.1 l2transport
```

```
Router(config-subif-l2)# no shutdown
Router(config-subif-l2)# encapsulation dot1q 1
Router(config-subif-l2)# rewrite ingress tag pop 1 symmetric
Router(config-subif-l2)#commit
Router(config-subif-l2)#exit
```

Running Configuration

This section shows the Layer 2 interface running configuration.

```
configure
 interface bundle-ether2.1 l2transport
  no shutdown
  encapsulation dot1q 1
  rewrite ingress tag pop 1 symmetric
!
```

Configure a Bridge Domain

Perform the following steps to configure the bridge domain on PE1 and PE2.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 6005
Router(config-l2vpn-bg)# bridge-domain 6005
Router(config-l2vpn-bg-bd)# interface Bundle-Ether2.1
Router(config-l2vpn-bg-bd-ac)# evi 6005
Router(config-l2vpnbg-bd-evi)# commit
Router(config-l2vpnbg-bd-evi)# exit
```

Running Configuration

This section shows the bridge domain running configuration.

```
configure
l2vpn
 bridge group 6005
  bridge-domain 6005
   interface Bundle-Ether2.1
    evi 6005
!
```

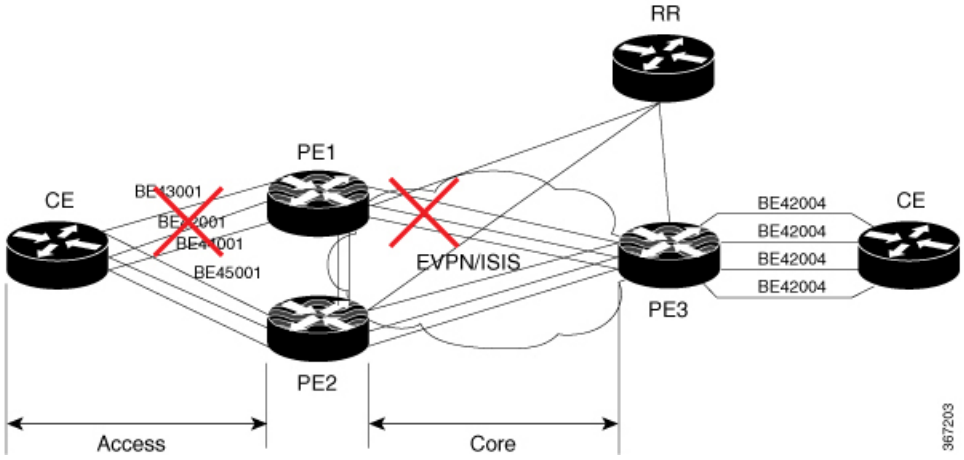
EVPN Core Isolation Protection

The EVPN Core Isolation Protection feature enables you to monitor and detect the link failure in the core. When a core link failure is detected in the provider edge (PE) device, EVPN brings down the PE's Ethernet Segment (ES), which is associated with access interface attached to the customer edge (CE) device.

EVPN replaces ICCP in detecting the core isolation. This new feature eliminates the use of ICCP in the EVPN environment.

Consider a topology where CE is connected to PE1 and PE2. PE1, PE2, and PE3 are running EVPN over the MPLS core network. The core interfaces can be Gigabit Ethernet or bundle interface.

Figure 50: EVPN Core Isolation Protection



When the core links of PE1 go down, the EVPN detects the link failure and isolates PE1 node from the core network by bringing down the access network. This prevents CE from sending any traffic to PE1. Since BGP session also goes down, the BGP invalidates all the routes that were advertised by the failed PE. This causes the remote PE2 and PE3 to update their next-hop path-list and the MAC routes in the L2FIB. PE2 becomes the forwarder for all the traffic, thus isolating PE1 from the core network.

When all the core interfaces and BGP sessions come up, PE1 advertises Ethernet A-D Ethernet Segment (ES-EAD) routes again, triggers the service carving and becomes part of the core network.

Configure EVPN Core Isolation Protection

Configure core interfaces under EVPN group and associate that group to the Ethernet Segment which is an attachment circuit (AC) attached to the CE. When all the core interfaces go down, EVPN brings down the associated access interfaces which prevents the CE device from using those links within their bundles. All interfaces that are part of a group go down, EVPN brings down the bundle and withdraws the ES-EAD route.

Restrictions

- A maximum of 24 groups can be created under the EVPN.
- A maximum of 12 core interfaces can be added under the group.
- The core interfaces can be reused among the groups. The core interface can be a bundle interface.
- EVPN group must only contain core interfaces, do not add access interfaces under the EVPN group.
- The access interface can only be a bundle interface.
- EVPN core facing interfaces must be physical or bundle main interfaces only. Sub-interfaces are not supported.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# group 42001
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/1
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/3
Router(config-evpn-group)#exit
```

```
!
Router(config-evpn)# group 43001
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/2
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/4
Router(config-evpn-group)#exit
!
Router# configure
Router(config)# evpn
Router(config-evpn)# interface bundle-Ether 42001
Router(config-evpn-ac)# core-isolation-group 42001
Router(config-evpn-ac)# exit
!
Router(config-evpn)# interface bundle-Ether 43001
Router(config-evpn-ac)# core-isolation-group 43001
Router(config-evpn-ac)# commit
```

Running Configuration

```
configure
 evpn
  group 42001
   core interface GigabitEthernet0/2/0/1
   core interface GigabitEthernet0/2/0/3
   !
  group 43001
   core interface GigabitEthernet0/2/0/2
   core interface GigabitEthernet0/2/0/4
   !
 !
configure
 evpn
  interface bundle-Ether 42001
   core-isolation-group 42001
   !
  interface bundle-Ether 43001
   core-isolation-group 43001
   !
  !
```

Verification

The **show evpn group** command displays the complete list of evpn groups, their associated core interfaces and access interfaces. The status, up or down, of each interface is displayed. For the access interface to be up, at least one of the core interfaces must be up.

```
Router# show evpn group /* Lists specific group with core-interfaces and access interface
status */
EVPN Group: 42001
  State: Ready
  Core Interfaces:
     Bundle-Ethernet110: down
     Bundle-Ethernet111: down
     GigabethEthernet0/2/0/1: up
     GigabethEthernet0/2/0/3: up
     GigabethEthernet0/4/0/8: up
     GigabethEthernet0/4/0/9: up
     GigabethEthernet0/4/0/10: up
  Access Interfaces:
     Bundle-Ether42001: up
```

```
EVPN Group: 43001
  State: Ready
  Core Interfaces:
     Bundle-Ethernet110: down
     GigabethEthernet0/2/0/2: up
     GigabethEthernet0/2/0/4: up
     GigabethEthernet0/4/0/9: up

  Access Interfaces:
     Bundle-Ether43001: up
```

EVPN Routing Policy

The EVPN Routing Policy feature provides the route policy support for address-family L2VPN EVPN. This feature adds EVPN route filtering capabilities to the routing policy language (RPL). The filtering is based on various EVPN attributes.

A routing policy instructs the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another.

This feature enables you to configure route-policies using EVPN network layer reachability information (NLRI) attributes of EVPN route type 1 to 5 in the route-policy match criteria, which provides more granular definition of route-policy. For example, you can specify a route-policy to be applied to only certain EVPN route-types or any combination of EVPN NLRI attributes. This feature provides flexibility in configuring and deploying solutions by enabling route-policy to filter on EVPN NLRI attributes.

To implement this feature, you need to understand the following concepts:

- Routing Policy Language
- Routing Policy Language Structure
- Routing Policy Language Components
- Routing Policy Language Usage
- Policy Definitions
- Parameterization
- Semantics of Policy Application
- Policy Statements
- Attach Points

For information on these concepts, see Implementing Routing Policy.

Currently, this feature is supported only on BGP neighbor "in" and "out" attach points. The route policy can be applied only on inbound or outbound on a BGP neighbor.

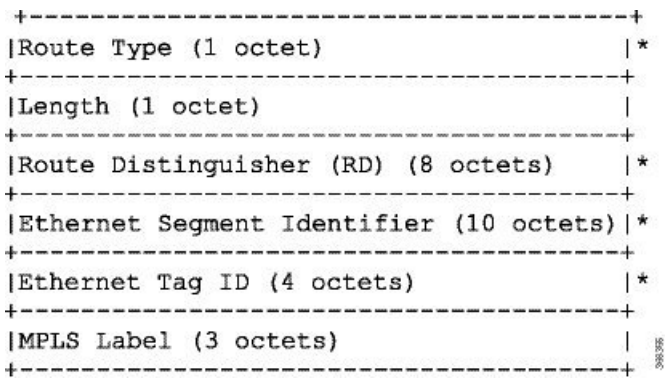
EVPN Route Types

The EVPN NLRI has the following different route types:

Route Type 1: Ethernet Auto-Discovery (AD) Route

The Ethernet (AD) routes are advertised on per EVI and per Ethernet Segment Identifier (ESI) basis. These routes are sent per Ethernet segment (ES). They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed.

An Ethernet A-D route type specific EVPN NLRI consists of the following fields:



NLRI Format: Route-type 1:

```
[Type][Len][RD][ESI][ETag][MPLS Label]
```

Net attributes: `[Type][RD][ESI][ETag]`

Path attributes: `[MPLS Label]`

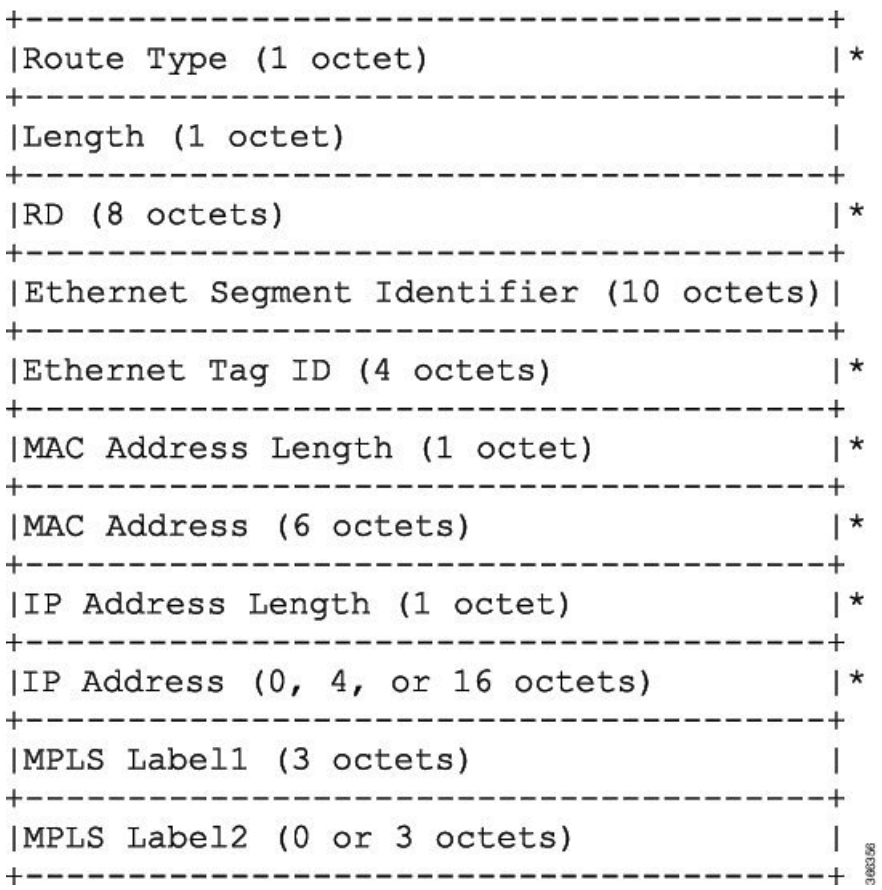
Example

```
route-policy evpn-policy
  if rd in (10.0.0.1:0) [and/or evpn-route-type is 1] [and/or esi in
(0a1.a2a3.a4a5.a6a7.a8a9)] [and/or etag is 4294967295] then
    set ..
  endif
end-policy
!
route-policy evpn-policy
  if rd  in (1.0.0.2:0) [and/or evpn-route-type is 1] [and/or esi in
(00a1.a2a3.a4a5.a6a7.a8a9)] [and/or etag is 4294967295] then
    set ..
  endif
end-policy
```

Route Type 2: MAC/IP Advertisement Route

The host's IP and MAC addresses are advertised to the peers within NLRI. The control plane learning of MAC addresses reduces unknown unicast flooding.

A MAC/IP Advertisement Route type specific EVPN NLRI consists of the following fields:



NLRI Format: Route-type 2:

```
[Type][Len][RD][ESI][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr][MPLS Label1][MPLS
Label2]
```

Net attributes: `[Type][RD][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr]`

Path attributes: `[ESI], [MPLS Label1], [MPLS Label2]`

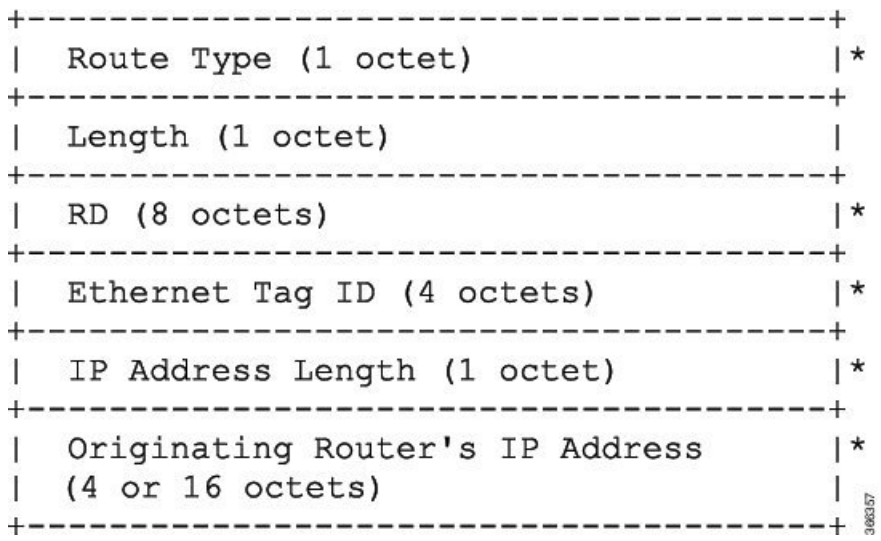
Example

```
route-policy evpn-policy
  if rd in (10.0.0.2:0) [and/or evpn-route-type is 2] [and/or esi in
(0000.0000.0000.0000.0000)] [and/or etag is 0] [and/or macaddress in (0013.aabb.ccdd)]
[and/or destination in (1.2.3.4/32)] then
    set ..
  endif
end-policy
```

Route Type 3: Inclusive Multicast Ethernet Tag Route

This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.

An Inclusive Multicast Ethernet Tag route type specific EVPN NLRI consists of the following fields:



NLRI Format: Route-type 3:

```
[Type][Len][RD][ETag][IP Addr Len][Originating Router's IP Addr]
```

Net attributes: `[Type][RD][ETag][IP Addr Len][Originating Router's IP Addr]`

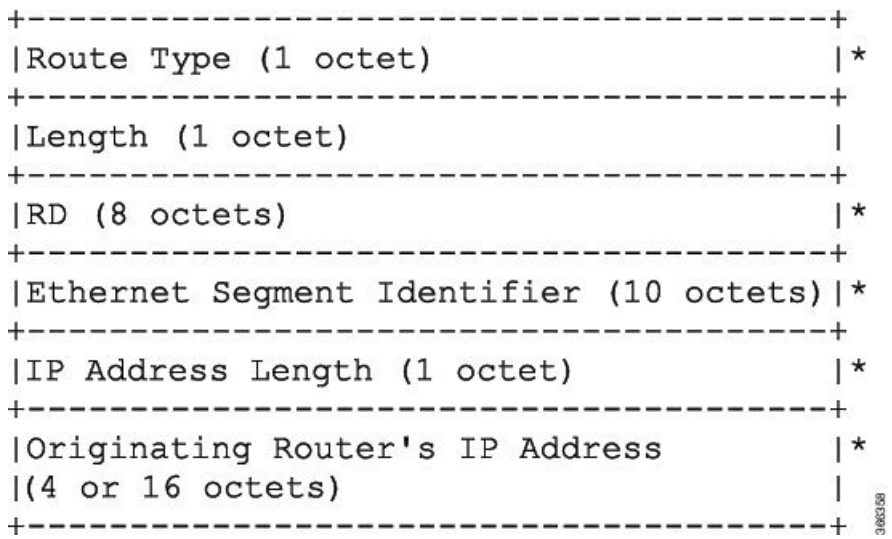
Example

```
route-policy evpn-policy
  if rd  in (10.0.0.1:300) [and/or evpn-route-type is 3] [and/or etag is 0] [and/or
evpn-originator in (10.0.0.1)] then
    set ..
  endif
end-policy
```

Route Type 4: Ethernet Segment Route

Ethernet segment routes enable to connect a CE device to two or PE devices. ES route enables the discovery of connected PE devices that are connected to the same Ethernet segment.

An Ethernet Segment route type specific EVPN NLRI consists of the following fields:



NLRI Format: Route-type 4:

```
[Type][Len][RD][ESI][IP Addr Len][Originating Router's IP Addr]
```

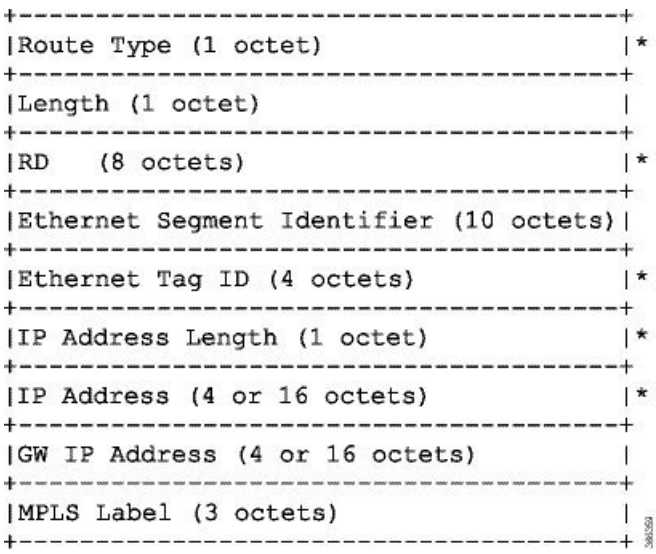
Net attributes: `[Type][RD][ESI][IP Addr Len][Originating Router's IP Addr]`

Example

```
route-policy evpn-policy
  if rd  in (10.0.0.1:0) [and/or evpn-route-type is 4] [and/or esi in
(00a1.a2a3.a4a5.a6a7.a8a9)] [and/or evpn-originator in (10.0.0.1)] then
    set ..
  endif
end-policy
```

Route Type 5: IP Prefix Route

An IP Prefix Route type specific EVPN NLRI consists of the following fields:



NLRI Format: Route-type 5:

```
[Type][Len][RD][ESI][ETag][IP Addr Len][IP Addr][GW IP Addr][Label]
```

Net attributes: `[Type][RD][ETag][IP Addr Len][IP Addr]`

Path attributes: `[ESI], [GW IP Addr], [Label]`

Example

```
route-policy evpn-policy
  if rd in (30.30.30.30:1) [and/or evpn-route-type is 5] [and/or esi in
(0000.0000.0000.0000.0000)] [and/or etag is 0] [and/or destination in (12.2.0.0/16)] [and/or
 evpn-gateway in (0.0.0.0)] then
    set ..
  endif
end-policy
```

EVPN RPL Attribute

Route Distinguisher

A Route Distinguisher (rd) attribute consists of eight octets. An rd can be specified for each of the EVPN route types. This attribute is not mandatory in route-policy.

Example

```
rd in (1.2.3.4:0)
```

EVPN Route Type

EVPN route type attribute consists of one octet. This specifies the EVPN route type. The EVPN route type attribute is used to identify a specific EVPN NLRI prefix format. It is a net attribute in all EVPN route types.

Example

```
evpn-route-type is 3


The following are the various EVPN route types that can be used:
1 - ethernet-ad
2 - mac-advertisement
3 - inclusive-multicast
4 - ethernet-segment
5 - ip-advertisement
```

IP Prefix

An IP prefix attribute holds IPv4 or IPv6 prefix match specification, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. When IP prefix is specified in EVPN route type 2, it represents either a IPv4 or IPv6 host IP Address (/32 or /128). When IP prefix is specified in EVPN route type 5, it represents either IPv4 or IPv6 subnet. It is a net attribute in EVPN route type 2 and 5.

Example

```
destination in (128.47.10.2/32)
destination in (128.47.0.0/16)
destination in (128:47::1/128)
destination in (128:47::0/112)
```

esi

An Ethernet Segment Identifier (ESI) attribute consists of 10 octets. It is a net attribute in EVPN route type 1 and 4, and a path attribute in EVPN route type 2 and 5.

Example

```
esi in (ffff.ffff.ffff.ffff.fff0)
```

etag

An Ethernet tag attribute consists of four octets. An Ethernet tag identifies a particular broadcast domain, for example, a VLAN. An EVPN instance consists of one or more broadcast domains. It is a net attribute in EVPN route type 1, 2, 3 and 5.

Example

```
etag in (10000)
```

mac

The mac attribute consists of six octets. This attribute is a net attribute in EVPN route type 2.

Example

```
mac in (0206.acb1.e806)
```

evpn-originator

The evpn-originator attribute specifies the originating router's IP address (4 or 16 octets). This is a net attribute in EVPN route type 3 and 4.

Example

```
evpn-originator in (1.2.3.4)
```

evpn-gateway

The evpn-gateway attribute specifies the gateway IP address. The gateway IP address is a 32-bit or 128-bit field (IPv4 or IPv6), and encodes an overlay next-hop for the IP prefixes. The gateway IP address field can be zero if it is not used as an overlay next-hop. This is a path attribute in EVPN route type 5.

Example

```
evpn-gateway in (1.2.3.4)
```

EVPN RPL Attribute Set

In this context, the term set is used in its mathematical sense to mean an unordered collection of unique elements. The policy language provides sets as a container for groups of values for matching purposes. Sets are used in conditional expressions. The elements of the set are separated by commas. Null (empty) sets are allowed.

prefix-set

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The prefix-set specifies one or more IP prefixes.

Example

```
prefix-set ip_prefix_set
14.2.0.0/16,
54.0.0.0/16,
12.12.12.0/24,
50:50::1:0/112
end-set
```

mac-set

The mac-set specifies one or more MAC addresses.

Example

```
mac-set mac_address_set
1234.2345.6789,
2345.3456.7890
end-set
```

esi-set

The esi-set specifies one or more ESI's.

Example

```
esi-set evpn_esi_set
1234.2345.3456.4567.5678,
1234.2345.3456.4567.5670
end-set
```

etag-set

The etag-set specifies one or more Ethernet tags.

Example

```
etag-set evpn_etag_set
10000,
20000
end-set
```

Configure EVPN RPL Feature

The following section describe how to configure mac-set, esi-set, evpn-gateway, and evpn-originator.

```
/* Configuring a mac-set and refering it in a route-policy (Attach point - neighbor-in) */
Router# configure
Router(config)# mac-set demo_mac_set
Router(config-mac)# 1234.ffff.aaa3,
Router(config-mac)# 2323.4444.ffff
Router(config-mac)# end-set
Router(config)# !
Router(config)# route-policy policy_use_pass_mac_set
Router(config-rpl)# if mac in demo_mac_set then
Router(config-rpl-if)# set med 200
Router(config-rpl-if)# else
Router(config-rpl-else)# set med 1000
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit
```

```
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# !
Router(config-bgp-af)# neighbor 10.0.0.10
Router(config-bgp-nbr)# remote-as 8
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy policy_use_pass_mac_set in
Router(config-bgp-nbr-af)# commit

/* Configuring a esi-set and refering it in a route-policy (Attach point - neighbor-in) */
Router# configure
Router(config)# esi-set demo_esi
Router(config-esi)# ad34.1233.1222.ffff.44ff,
Router(config-esi)# ad34.1233.1222.ffff.6666
Router(config-esi)# end-set
Router(config)# !
Router(config)# route-policy use_esi
Router(config-rpl)# if esi in demo_esi then
Router(config-rpl-if)# set local-preference 100
Router(config-rpl-if)# else
Router(config-rpl-else)# set local-preference 300
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit

/* Configuring evpn-gateway/evpn-originator in a route-policy (Attach point - neighbor-in
and out) */
Router# configure
Router(config)# route-policy gateway_demo
Router(config-rpl)# if evpn-gateway in (10.0.0.0/32) then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
Router(config)# commit
Router(config)# route-policy originator_demo
Router(config-rpl)# if evpn-originator in (10.0.0.1/32) then
Router(config-rpl-if)# set local-preference 100
Router(config-rpl-if)# else
Router(config-rpl-else)# set med 200
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# !
Router(config-bgp-af)# neighbor 10.0.0.10
Router(config-bgp-nbr)# remote-as 8
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy gateway_demo in
Router(config-bgp-nbr-af)# route-policy originator_demo out
Router(config-bgp-nbr-af)# commit
```

Running Configuration

```
/* Configuring a mac-set and refering it in a route-policy (Attach point - neighbor-in) */
mac-set demo_mac_set
  1234.ffff.aaa3,
  2323.4444.ffff
end-set
!
route-policy policy_use_pass_mac_set
```

```
  if mac in demo_mac_set then
    set med 200
  else
    set med 1000
  endif
end-policy
!
router bgp 100
 address-family l2vpn evpn
 !
 neighbor 10.0.0.10
  remote-as 8
  address-family l2vpn evpn
  route-policy policy_use_pass_mac_set in
  !
 !
!
end

/* Configuring a esi-set and refering it in a route-policy (Attach point - neighbor-in) */
Wed Oct 26 11:52:23.720 IST
esi-set demo_esi
  ad34.1233.1222.ffff.44ff,
  ad34.1233.1222.ffff.6666
end-set
!
route-policy use_esi
  if esi in demo_esi then
    set local-preference 100
  else
    set local-preference 300
  endif
end-policy
```

EVPN Route Policy Examples

```
route-policy ex_2
  if rd in (2.2.18.2:1004) and evpn-route-type is 1 then
    drop
  elseif rd in (2.2.18.2:1009) and evpn-route-type is 1 then
    drop
  else
    pass
  endif
end-policy
!
route-policy ex_3
  if evpn-route-type is 5 then
    set extcommunity bandwidth (100:9999)
  else
    pass
  endif
end-policy
!
route-policy samp
end-policy
!
route-policy samp1
  if rd in (30.0.101.2:0) then
    pass
  endif
end-policy
```

```
!
route-policy samp2
  if rd in (30.0.101.2:0, 1:1) then
    pass
  endif
end-policy
!
route-policy samp3
  if rd in (*:*) then
    pass
  endif
end-policy
!
route-policy samp4
  if rd in (30.0.101.2:*) then
    pass
  endif
end-policy
!
route-policy samp5
  if evpn-route-type is 1 then
    pass
  endif
end-policy
!
route-policy samp6
  if evpn-route-type is 2 or evpn-route-type is 5 then
    pass
  endif
end-policy
!
route-policy samp7
  if evpn-route-type is 4 or evpn-route-type is 3 then
    pass
  endif
end-policy
!
route-policy samp8
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 then
    pass
  endif
end-policy
!
route-policy samp9
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 or evpn-route-type
 is 4 then
    pass
  endif
end-policy
!
route-policy test1
  if evpn-route-type is 2 then
    set next-hop 10.2.3.4
  else
    pass
  endif
end-policy
!
route-policy test2
  if evpn-route-type is 2 then
    set next-hop 10.10.10.10
  else
    drop
  endif
```

```
end-policy
!
route-policy test3
  if evpn-route-type is 1 then
    set tag 9988
  else
    pass
  endif
end-policy
!
route-policy samp21
  if mac in (6000.6000.6000) then
    pass
  endif
end-policy
!
route-policy samp22
  if extcommunity rt matches-any (100:1001) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp23
  if evpn-route-type is 1 and esi in (aaaa.bbbb.cccc.dddd.eeee) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp24
  if evpn-route-type is 5 and extcommunity rt matches-any (100:1001) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp25
  if evpn-route-type is 2 and esi in (1234.1234.1234.1234.1236) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp26
  if etag in (20000) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp27
  if destination in (99.99.99.1) and etag in (20000) then
    pass
  else
    drop
  endif
end-policy
!
```

```
route-policy samp31
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 or evpn-route-type
 is 4 or evpn-route-type is 5 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp33
  if esi in evpn_esi_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp34
  if destination in (90:1:1::9/128) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp35
  if destination in evpn_prefix_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp36
  if evpn-route-type is 3 and evpn-originator in (80:1:1::3) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp37
  if evpn-gateway in (10:10::10) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp38
  if mac in evpn_mac_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp39
  if mac in (6000.6000.6002) then
    pass
  else
    drop
  endif
end-policy
```

```
!
route-policy samp41
  if evpn-gateway in (10.10.10.10, 10:10::10) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp42
  if evpn-originator in (24.162.160.1/32, 70:1:1::1/128) then
    pass
  else
    drop
  endif
end-policy
!
route-policy example
  if rd in (62300:1903) and evpn-route-type is 1 then
    drop
  elseif rd in (62300:19032) and evpn-route-type is 1 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp100
  if evpn-route-type is 4 or evpn-route-type is 5 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp101
  if evpn-route-type is 4 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp102
  if evpn-route-type is 4 then
    drop
  elseif evpn-route-type is 5 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp103
  if evpn-route-type is 2 and destination in evpn_prefix_set1 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp104
  if evpn-route-type is 1 and etag in evpn_etag_set1 then
    drop
```

```
  elseif evpn-route-type is 2 and mac in evpn_mac_set1 then
    drop
  elseif evpn-route-type is 5 and esi in evpn_esi_set1 then
    drop
  else
    pass
  endif
end-policy
!
```

CFM on EVPN ELAN

Table 27: Feature History Table

Feature Name	Release Information	Feature Description
CFM on EVPN ELAN	Release 7.4.1	This feature allows you to effectively manage a network with EVPN services running EVPN ELAN and helps you to monitor the ELAN services, thereby providing high-speed Layer 2 and Layer 3 services with high resiliency. This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native mode only. The following offload types are supported: • Hardware (HW) Offload • Non-Offload • Software (SW) Offload

Table 28: Feature History Table

Feature Name	Release Information	Feature Description
CFM on EVPN ELAN	Release 7.6.1	Connectivity fault management (CFM) enables monitoring an Ethernet network with multiple service instances. With CFM now supporting single-homed EVPN Emulated Local Area Network (ELAN) services, you can monitor users' services against their contractual service level agreements. This removes the operational complexity of managing different market segments and subscribers on your network.

Connectivity fault management (CFM) is a service-level Operations and Maintenance (OAM) protocol that provides tools for monitoring and troubleshooting end-to-end Ethernet services for each VLAN. This includes proactive connectivity monitoring, fault verification, and fault isolation.

Supported Offload Types and Timer Values

The following are supported offload types:

- Hardware (HW) Offload type: The check message (CCM) timers for a CFM session on a physical interface is less than one second.

Note The Hardware (HW) Offload type is supported only in Cisco NC 57 line cards.

- Non-Offload type: The CCM timers for a CFM session on a physical interface is greater than one second.
- Software (SW) Offload type: The CFM session on a bundle interface.

The following are the supported timer values:

- 10s: Interval of 10 seconds
- 1m: Interval of 1 minute
- 10m: Interval of 10 minutes
- Non-Offload type supports 10s, 1m on the physical interface
- SW Offload type supports 10s,1m, LAG 3.33
- CCM interval of 10m is not supported on NCS57 line cards operating in native mode.
- A maximum of 8K CFM UP MEP sessions are supported because EVPN ELAN supports 8K bridge domains on NCS57 line cards operating in native mode.
- You can configure both UP and DOWN MEPs on the same L2VPN with EVPN ELAN in the Cisco NC 57 line cards.
- Supports 3.3ms,10ms,100ms,1s CCM timers for HW-Offload UP MEP with EVPN ELAN on NCS57 line cards operating in native mode.

Restrictions for CFM on EVPN ELAN

CFM on EVPN ELAN is subjected to these restrictions:

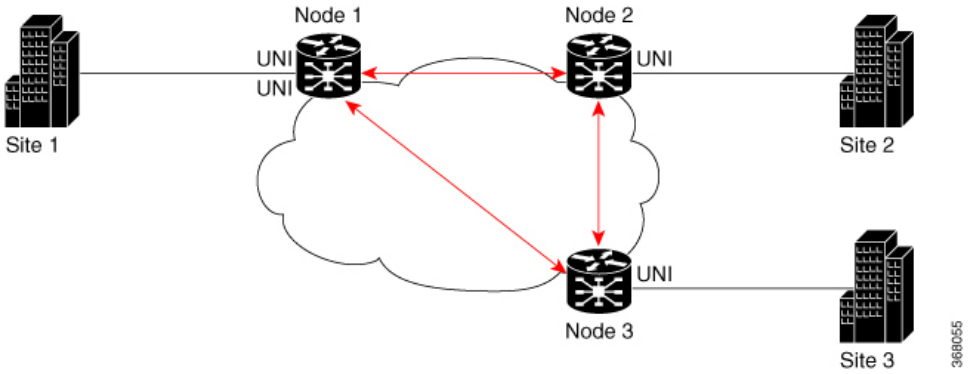
- Supports only single-homed EVPN ELAN.
- Supports single homing with one AC per PW.
- DOWN MEP on AC interface of EVPN-BD is not supported.
- Does not support loss measurement.
- Does not support Y1731.
- CFM over EVPN ELAN with MEPs along with multiple AC scenarios supports CCM and does not support LBM or LBR.

CFM on EVPN ELAN does not support the following configurations:

- UP MEP of different domain and same level on same EVPN-BD
- UP MEP of different level on different AC part of same BD as all AC interfaces are part of same service provider domain (EVPN-BD) in PE.

Configure CFM on EVPN ELAN

Figure 51: CFM on EVPN ELAN: Full Mesh Topology



Node 1, 2 and 3 in this topology can be Cisco routers.

Configuring CFM on EVPN ELAN involves these main tasks:

- Enabling CFM service continuity check
- Configuring MEP cross-check
- Enabling CFM for the interface

Configuration Example for CFM on EVPN ELAN: Full Mesh Topology

```
/* Enabling CFM continuity check */
Router# ethernet cfm
Router(config-cfm# domain bd-domain level 1 id null
Router(config-cfm-dmn)# service bd-domain bridge group bg-elan bridge-domain bd-elan id
icc-based MC MCMC
Router(config-cfm-dmn-svc)# continuity-check interval 1m
/* Configuring MEP cross-check */
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-dmn-svc)# mep-id 1112
Router(config-cfm-dmn-svc)# mep-id 1113
Router(config-cfm-dmn-svc)# commit
```

Repeat the above configurations for node 2 and node 3, with the respective mep-id values. For node 2, configure MEP cross-check with respective mep-id values of node 1 and node 3 (1111 and 1113 respectively, in this example). For node 3, configure MEP cross-check with respective mep-id values of node 1 and node 2 (1111 and 1112 respectively, in this example).

```
/* Enabling CFM on the interface */
Router(config)# interface gigabitEthernet  l2transport
Router(config-subif)# description bg-elan
Router(config-subif)# encapsulation dot1q 100
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# mtu 9100
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain bd-domain service bd-service mep-id 1111
Router(config-if-cfm-mep)# commit
```

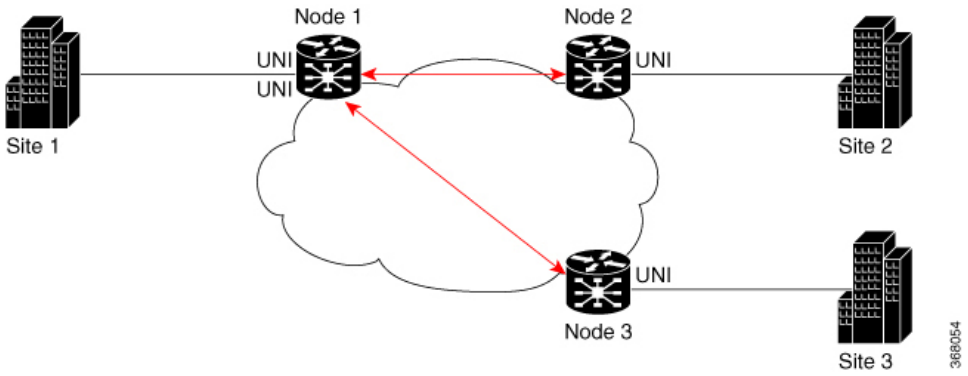
You must repeat the above configurations for node 2 and node 3, with the respective *mep-id* values (that is, 1112 for node 2 and 1113 for node 3, in this example).

Running Configuration for CFM on EVPN ELAN: Full Mesh Topology

This sections shows the running configuration on node 1.

```
ethernet cfm
 domain bd-domain level 1 id null
  service bd-domain bridge group bg-elan bridge-domain bd-elan id icc-based MC MCMC
   continuity-check interval 1m
   mep crosscheck
    mep-id 1112
    mep-id 1113
   !
  !
 !
!

interface GigabitEthernet l2transport
 description bg-elan
 encapsulation dot1q 100
 rewrite ingress tag pop 1 symmetric
 mtu 9100
 ethernet cfm
  mep domain bd-domain service bd-service mep-id 1111
 !
```

Figure 52: CFM on EVPN ELAN: Hub and Spoke Topology



Configuration Example for CFM on EVPN ELAN: Hub and Spoke Topology

The CFM configuration for the hub and spoke topology remains the same as that of full mesh topology mentioned above, except for these additional steps for SLA profile configuration to be done under the interface.

```
/* 1112 and 1113 in this example, are the mep-id values of node 2 and node 3 */
Router(config)#interface gigabitEthernet  l2transport
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain bd-domain service bd-service mep-id 1111
Router(config-if-cfm-mep)# sla operation profile test-profile1 target mep-id 1112
Router(config-if-cfm-mep)# sla operation profile test-profile2 target mep-id 1112
Router(config-if-cfm-mep)# sla operation profile test-profile1 target mep-id 1113
Router(config-if-cfm-mep)# sla operation profile test-profile2 target mep-id 1113
Router(config-if-cfm-mep)# commit
```

Running Configuration for CFM on EVPN ELAN: Hub and Spoke Topology

This sections shows the running configuration on node 1.

```
interface GigabitEthernet l2transport
 description bg-elan
 encapsulation dot1q 100
 rewrite ingress tag pop 1 symmetric
 mtu 9100
 ethernet cfm
  mep domain bd-domain service bd-service mep-id 1111
   sla operation profile test-profile1 target mep-id 1112
   sla operation profile test-profile2 target mep-id 1112
   sla operation profile test-profile1 target mep-id 1113
   sla operation profile test-profile2 target mep-id 1113
 !
```

Example 1: The below configurations details UP MEPs of same domain and level on the following:

- Multiple AC per BD on local
- Single AC per BD on remote

```
Router#sh run ethernet cfm
Tue Sep 14 19:08:32.666 UTC
ethernet cfm
 domain BD-DOMAIN level 4 id null
  service BD-SERVICE bridge group ELAN_FUNC_3 bridge-domain FUNC_3 id number 100
   continuity-check interval 10s
   mep crosscheck
    mep-id 5
    mep-id 1101
    mep-id 1103


Router#sh run l2vpn
Tue Sep 14 19:08:37.979 UTC
l2vpn
 bridge group ELAN_FUNC_3
  bridge-domain FUNC_3
   interface TenGigE0/0/0/0.1
   !
   interface TenGigE0/0/0/1.2
   !
   evi 101

Router#sh run int Te0/0/0/0.1
Tue Sep 14 19:08:42.677 UTC
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain BD-DOMAIN service BD-SERVICE mep-id 1103

Router#sh run int Te0/0/0/1.2
Tue Sep 14 19:08:49.485 UTC
interface TenGigE0/0/0/1.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain BD-DOMAIN service BD-SERVICE mep-id 5
```

Example 2: The below configurations details multiple UP MEPs of same domain and level on AC interfaces that are part of the same BD:

```
Router#sh run ethernet cfm
ethernet cfm
 domain BD-DOMAIN level 4 id null
  service BD-SERVICE bridge group ELAN_FUNC_3 bridge-domain FUNC_3
   continuity-check interval 10s
   mep crosscheck
    mep-id 1
    mep-id 2
    mep-id 21
    mep-id 22
domain BD-DOMAIN1 level 3 id null
  service BD-SERVICE1 bridge group ELAN_FUNC_3 bridge-domain FUNC_3
   continuity-check interval 10s
   mep crosscheck
    mep-id 1001
    mep-id 1021
    mep-id 2001
    mep-id 2021


Router#sh run int Te0/0/0/0
 interface TenGigE0/0/0/0
 ethernet cfm
  mep domain BD-DOMAIN service BD-SERVICE mep-id 21
  mep domain BD-DOMAIN1 service BD-SERVICE1 mep-id 1021
 l2transport


Router#sh run int Te0/0/0/1
interface TenGigE0/0/0/1
 ethernet cfm
  mep domain BD-DOMAIN service BD-SERVICE mep-id 22
  mep domain BD-DOMAIN1 service BD-SERVICE1 mep-id 2021
 l2transport


Router#sh run l2vpn
l2vpn
 bridge group ELAN_FUNC_3
  bridge-domain FUNC_3
   interface TenGigE0/0/0/0
   interface TenGigE0/0/0/1
   Interface TenGigE0/0/0/2
   evi 101
```

Example 3: The below configurations details multiple services for different EVPN-BD on same domain level :

```
Router#sh run ethernet cfm
Tue Sep 14 19:22:01.196 UTC
ethernet cfm
 domain evpn-bd level 4 id null
  service evpn-bd1 bridge group BG1 bridge-domain BD1
   continuity-check interval 10s
   mep crosscheck
    mep-id 5
    mep-id 6
```

```
    mep-id 1101
    mep-id 1103
  service evpn-bd2 bridge group BG2 bridge-domain BD2
   continuity-check interval 10s
   mep crosscheck
    mep-id 11
    mep-id 21
    mep-id 101

Router#sh run l2vpn
l2vpn
 bridge group BG1
  bridge-domain BD1
   interface TenGigE0/0/0/0.1
   interface TenGigE0/0/0/1.2
   evi 101
bridge group BG2
  bridge-domain BD2
   interface TenGigE0/0/0/2.1
   interface TenGigE0/0/0/5.2
   evi 201

Router#sh run int Te0/0/0/0.1
Tue Sep 14 19:22:12.368 UTC
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain evpn-bd service evpn-bd1 mep-id 1103

Router#sh run int Te0/0/0/1.2
Tue Sep 14 19:22:19s.258 UTC
interface TenGigE0/0/0/1.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain evpn-bd service evpn-bd1 mep-id 5

Router#sh run int Te0/0/0/2.1
Tue Sep 14 19:22:23.539 UTC
interface TenGigE0/0/0/2.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain evpn-bd service evpn-bd2 mep-id 101

Router#sh run int Te0/0/0/5.2
Tue Sep 14 19:22:27.954 UTC
interface TenGigE0/0/0/5.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain evpn-bd service evpn-bd2 mep-id 11
```

Example 4: The below configurations details different EVPN-BD on different domain levels:

```
Router#sh run ethernet cfm
Tue Sep 14 19:39:39.522 UTC
ethernet cfm
 domain evpn-bd level 4 id null
  service evpn-bd1 bridge group BG1 bridge-domain BD1
   continuity-check interval 10s
   mep crosscheck
    mep-id 5
    mep-id 6
    mep-id 1101
```

```
    mep-id 1103
   !
  !
 !
 domain evpn-bd2 level 3 id null
  service evpn-bd2 bridge group BG2 bridge-domain BD2
   continuity-check interval 10s
   mep crosscheck
    mep-id 11
    mep-id 21
    mep-id 101
    mep-id 201
   !
  !
 !
!
Router#sh run l2vpn
Tue Sep 14 19:39:44.004 UTC
l2vpn
 bridge group BG1
  bridge-domain BD1
   interface TenGigE0/0/0/0.1
   !
   interface TenGigE0/0/0/1.2
   !
   evi 101
   !
  !
 !
 bridge group BG2
  bridge-domain BD2
   interface TenGigE0/0/0/2.1
   !
   interface TenGigE0/0/0/5.2
   !
   evi 201
   !
  !
 !
!
Router#sh run int Te0/0/0/0.1
Tue Sep 14 19:39:50.042 UTC
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain evpn-bd service evpn-bd1 mep-id 1103
  !
 !
!

Router#sh run int Te0/0/0/1.2
Tue Sep 14 19:39:53.798 UTC
interface TenGigE0/0/0/1.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain evpn-bd service evpn-bd1 mep-id 5
  !
 !
!

Router#sh run int Te0/0/0/2.1
Tue Sep 14 19:39:59.176 UTC
interface TenGigE0/0/0/2.1 l2transport
 encapsulation dot1q 1
```

```
 ethernet cfm
  mep domain evpn-bd2 service evpn-bd2 mep-id 101
  !
 !
!

Router#sh run int Te0/0/0/5.2
Tue Sep 14 19:40:03.689 UTC
interface TenGigE0/0/0/5.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain evpn-bd2 service evpn-bd2 mep-id 11
  !
 !
!
```

Related Topics

CFM on EVPN ELAN, on page 314

Associated Commands

- continuity-check
- ethernet cfm
- mep crosscheck
- mep domain
- sla operation

EVPN Bridging and VPWS Services over BGP-LU Underlay

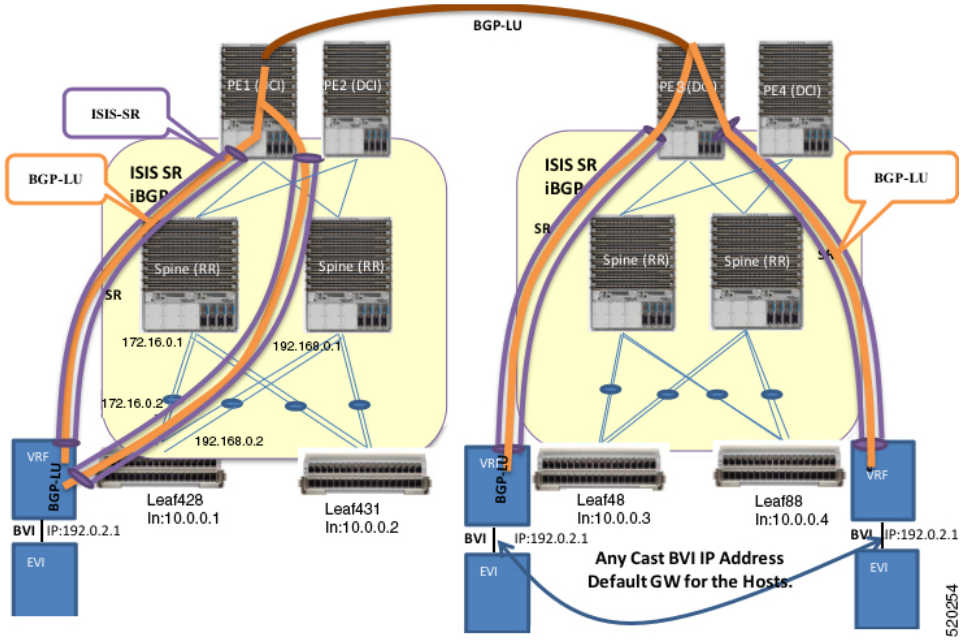
The EVPN Bridging and VPWS Services over BGP-LU Underlay feature allows you to configure end-to-end EVPN services between data centers (DCs). This feature allows you to perform ECMP at three-levels: transport, BGP- LU, and service level.

This feature supports the following services:

- IRB VRF over BGP-LU using IGP (SR or non-SR (LDP or IGP))
- EVPN Aliasing over BGP-LU using IGP (SR or non-SR (LDP or IGP))
- VPWS over BGP-LU using IGP

Note EVPN IRB with default-vrf over BGP-LU over IGP is not supported on the Cisco NCS routers.

Figure 53: EVPN Bridging and VPWS Services over BGP-LU Underlay



This section explains the topology of EVPN Bridging and VPWS Services over BGP-LU Underlay feature:

- Consider two data centers that are connected through DCI. Configure EVPN with bridging and inter-subnet routing on the leaf nodes.
- Configure EVPN instance with BVI attachment circuit to interface with L3-VRF.
- Configure BVI interface with anycast IP address with the same MAC address. This is the default gateway for all the hosts across the same EVPN bridged domain.
- The leaf acts as default gateway for its local hosts.
- Connect hosts to leaf nodes. Leaf nodes are routed across the spines. For DC interconnectivity, the spines are connected through provider edge (PE) device and Data Center Interconnect (DCI).
- IS-IS labelled IGP and I-BGP are enabled internally across the leaf nodes, spine and DCI. The spine acts as a Route Reflector (RR).
- Configure IS-IS SR policy across the leaf node, spine and DCI.
- Configure BGP-LU between the DCs.
- Labelled Unicast BGP routers are learnt across the leaf nodes and tunnelled through IGP labelled paths (IS-IS SR).

  For example, at Leaf428, BGP-LU routes are learnt for remote loopback 10.0.0.3 and 10.0.0.4.
- IRB (BVI) interface routes are learnt across the EVPN instances and programmed as labelled routes tunnelled through BGP-LU.

  For example, at Leaf428, 192.0.2.1 can be reached with two BGP-LU paths 10.0.0.3 and 10.0.0.4.

After establishing the BGP-LU services, you can configure either EVPN instance or EVPN VPWS to support BGP-LU.

Configure EVPN Bridging and VPWS Services over BGP-LU Underlay

Perform these tasks to configure the EVPN Bridging and VPWS Services over BGP-LU Underlay feature.

- Configure IGP
- Configure BGP
- Configure EVPN instance and ESI
- Configure BVI (IRB) Interface
- Configure VRF
- Configure BVI with VRF
- Configure VRF under BGP
- Configure bridge domain and associate with attachment circuits and EVPN instance
- Configure bridge domain and associate with attachment circuits, EVPN instance and BVI
- Configure EVPN VPWS

Configuration Example

```
/* Configure IGP */
IGP configuration is a pre-requisite to configure EVPN. IGP can be OSPF or ISIS.
Router# configure
Router(config)#router ospf 1
Router(config-ospf)#router-id 209.165.201.1
Router(config-ospf)#area 10
Router(config-ospf-ar)#interface loopback0\
Router(config-ospf-ar-if)#exit
Router(config-ospf-ar)#interface TenGigE0/0/0/1\
Router(config-ospf-ar-if)#exit
Router(config-ospf-ar)#interface TenGigE0/0/0/17\
Router(config-ospf-ar-if)#commit

/* Configure BGP enable BGP-LU, and configure iBGP peering */
Router# configure
Router(config)#router bgp 100
Router(config-bgp)#router-id 209.165.201.1
Router(config-bgp)#bgp graceful-restart
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#redistribute connected
Router(config-bgp-af)#network 209.165.200.225/27
Router(config-bgp-af)#allocate-label all
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#allocate-label all
Router(config-bgp-af)#exit
Router(config-bgp)#neighbor-group spines
Router(config-bgp-nbrgrp)#remote-as 100
Router(config-bgp-nbrgrp)#update-source loopback0
Router(config-bgp-nbrgrp)#address-family ipv4 labeled-unicast multipath
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#address-family ipv6 labeled-unicast multipath
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#address-family l2vpn evpn
```

```
Router(config-bgp-nbrgrp-af)#advertise vpnv4 unicast re-originated
Router(config-bgp-nbrgrp-af)#advertise vpnv6 unicast re-originated
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)exit
Router(config-bgp)neighbor 209.165.200.225
Router(config-bgp-nbr)#use neighbor-group spines
Router(config-bgp-nbr)#commit

/* Configure VPN4 address-family to enforce iBGP policy. */
Router(config)#router bgp 100
Router(config-bgp)#router-id 209.165.201.1
Router(config-bgp)#ibgp policy out enforce-modifications
Router(config-bgp)#address-family vpnv4 unicast
Router(config-bgp-af)#commit

The above steps establish BGP-LU. You can configure either EVPN VPLS or EVPN VPWS on the
PE routers to support BGP-LU.
/* Configure EVPN instance and ESI */
Router#configure
Router(config)#evpn
Router(config-evpn)#evi 100
Router(config-evpn-instance)#advertise-mac
Router(config-evpn-instance-mac)#exit
Router(config-evpn-instance)#exit
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment identifier type 0 aa.aa.aa.aa.aa.aa.aa.aa.ac
Router(config-evpn-ac-es)#bgp route-target 0011.0011.0012
Router(config-evpn-ac)#commit

/* Configure BVI (IRB) Interface */
Router#configure
Router(config)#interface BVI200
Router(config-if)#ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)#commit

/*  Configure VRF */
Router# configure
Router(config)# vrf vpn2
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import route-target 81:2
Router(config-vrf-af)# exit
Router(config-vrf)# address-family ipv6 unicast
Router(config-vrf-af)# import route-target 81:2
Router(config-vrf-af)# commit

/* Configure BVI with VRF */
Router(config)# interface BVI200
Router(config-if)# host-routing
Router(config-if)# vrf vpn72
Router(config-if-vrf)# ipv4 address ipv4 address 192.0.2.1 255.255.255.0
Router(config-if-vrf)# mac-address 10.1111.1
Router(config-if)# commit

/* Configure VRF under BGP */
Router(config)# router bgp 100
Router(config-bgp)# vrf vpn2
Router(config-bgp-vrf)# rd 102:2
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# maximum-paths ibgp 8
Router(config-bgp-vrf-af)# redistribute connected
Router(config-bgp-vrf-af)# exit
Router(config-bgp-vrf)# address-family ipv6 unicast
```

```
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# maximum-paths ibgp 8
Router(config-bgp-vrf-af)# redistribute connected
Router(config-bgp-vrf-af)# commit

/* Configure bridge domain and associate with attachment circuits and EVPN instance */
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface BundleEther1.100
Router(config-l2vpn-bg-bd-ac)#evi 100
Router(config-l2vpn-bg-bd-evi)#commit

/* Configure bridge domain and associate with attachment circuits, EVPN instance and BVI
*/
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg2
Router(config-l2vpn-bg)#bridge-domain bd2
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/38.200
Router(config-l2vpn-bg-bd-ac)#routed interface BVI200
Router(config-l2vpn-bg-bd-bvi)#evi 200
Router(config-l2vpn-bg-bd-bvi)#commit
Router(config-l2vpn-bg-bd-bvi)#exit

Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg3
Router(config-l2vpn-bg)#bridge-domain bd3
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/38.202
Router(config-l2vpn-bg-bd-ac)#routed interface BVI202
Router(config-l2vpn-bg-bd-bvi)#evi 202
Router(config-l2vpn-bg-bd-bvi)#commit

/* Configure EVPN VPWS */
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#neighbor-group spines
Router(config-bgp-nbrgrp)#remote-as 100
Router(config-bgp-nbrgrp)#update-source loopback0
Router(config-bgp-nbrgrp)#address-family ipv4 labeled-unicast multipath
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#address-family ipv6 labeled-unicast multipath
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#address-family l2vpn evpn
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)exit
Router(config-bgp)neighbor 209.165.200.225
Router(config-bgp-nbr)#use neighbor-group spines
Router(config-bgp-nbr)#commit
Router(config-bgp-af)#exit
Router(config-bgp)#exit
Router(config)#l2vpn
Router(config-l2vpn)#xconnect group aa-evpn-vpws
Router(config-l2vpn-xc)#p2p vpws_513
Router(config-l2vpn-xc-p2p)#interface Bundle-Ether1.513
Router(config-l2vpn-xc-p2p)#neighbor evpn evi 513 target 513 source 513
Router(config-l2vpn-xc-p2p)# commit
```

Running Configuration

This section shows flooding disable running configuration.

```
/* Configure IGP */
router ospf 1
 router-id 209.165.201.1
 area 10
  interface Loopback0
  !
  interface TenGigE0/0/0/1
  !
  interface TenGigE0/0/0/17
  !
 !
/* Configure BGP, BGP-LU, and iBGP peering */
router bgp 100
 router-id 209.165.201.1
 bgp graceful-restart
 address-family ipv4 unicast
  redistribute connected
  network 209.165.200.225/27
  allocate-label all
address-family ipv6 unicast
  allocate-label all
neighbor-group spines
  remote-as 100
  update-source loopback0
  address-family ipv4 labeled-unicast multipath
  !
  address-family ipv6 labeled-unicast multipath
  !
  address-family l2vpn evpn
   advertise vpnv4 unicast re-originated
   advertise vpnv6 unicast re-originated
 !
 neighbor 209.165.200.225
  use neighbor-group spines
!

/* Configure VPN4 address-family */
router bgp 100
 router-id 209.165.201.1
 ibgp policy out enforce-modifications
 address-family vpnv4 unicast
!

/* Configure EVPN instance and ESI */
evpn
 evi 100
  advertise-mac
 !
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 aa.aa.aa.aa.aa.aa.aa.aa.ac
   bgp route-target 0011.0011.0012
  !
 !
!

/* Configuring BVI (IRB) Interface */
configure
 interface BVI200
  ipv4 address 192.0.2.1 255.255.255.0

/* Configure VRF */
vrf vpn2
```

```
 address-family ipv4 unicast
  import route-target 81:2
  !
  !
!
 address-family ipv6 unicast
  import route-target 81:2
  !
  !
!

/* Configure BVI with VRF */
interface BVI200
 host-routing
 vrf vpn72
  ipv4 address ipv4 address ipv4 address 192.0.2.1 255.255.255.0
   mac-address 10.1111.1
!

/* Configure VRF under BGP */
router bgp 100
 vrf vpn2
  rd 102:2
  address-family ipv4 unicast
   label mode per-vrf
   maximum-paths ibgp 8
   redistribute connected
  !
  address-family ipv6 unicast
   label mode per-vrf
   maximum-paths ibgp 8
   redistribute connected
  !
!

/* Configure bridge domain and associate with attachment circuits and EVPN instance */
l2vpn
 bridge group bg1
  bridge-domain b1
   interface Bundle-Ether1.100
   !
   evi 100

/*
bridge group bg2
  bridge-domain bd2
   interface TenGigE0/0/0/38.200
   !
   routed interface BVI200
   !
   evi 200
   !
  !

/* Configurige bridge domain and associate with attachment circuits, EVPN instance and BVI
 */
bridge group bg3
  bridge-domain bd3
   interface TenGigE0/0/0/38.202
   !
   routed interface BVI202
   !
   evi 202
   !
```

```
  !
 !

/* Configure EVPN VPWS */
configure
 router bgp 100
  neighbor-group spines
  remote-as 100
  update-source Loopback0
  address-family ipv4 labeled-unicast multipath
  !
  address-family ipv6 labeled-unicast multipath
  !
  address-family l2vpn evpn

neighbor 209.165.200.225
   use neighbor-group spines
 !
!
 l2vpn
  xconnect group aa-evpn-vpws
   p2p vpws_513
    interface Bundle-Ether1.513
    neighbor evpn evi 513 target 513 source 513
```

Verification

Verify that you have configured EVPN Bridging and VPWS Services over BGP-LU Underlay feature successfully.

Note Load Balancing is not supported for EVPN Bridging over BGP-LU with Multipaths.

```
Router#show cef vrf AIM9 10.0.0.1
Tue Jan 20 22:00:56.233 UTC
10.0.0.1/8, version 4, internal 0x5000001 0x0 (ptr 0x97d34b44) [1], 0x0 (0x0), 0x208
(0x98bef0f0)
 Updated Mar 18 06:01:46.175
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
   via 10.0.0.3/8, 7 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0x972c6f08 0x0]
    recursion-via-/32
    next hop VRF - 'default', table - 0xe0000000
    next hop 10.0.0.3/8 via 16448/0/21
     next hop 192.0.2.1/24 BE128        labels imposed {16111 64013 80002}
   via 100.0.0.88/32, 7 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0x972c6d68 0x0]
    recursion-via-/32
    next hop VRF - 'default', table - 0xe0000000
    next hop 10.0.0.4/8 via 16488/0/21
     next hop 192.0.2.1/24 BE128        labels imposed {16111 64009 80002}

Router#show l2vpn xconnect group aa-evpn-vpws xc-name vpws_513 detail
Wed Jan 22 13:14:05.878 GMT+4

Group aa-evpn-vpws, XC vpws_513, state is up; Interworking none
  AC: Bundle-Ether1.513, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
```

```
    VLAN ranges: [513, 513]
    MTU 1500; XC ID 0xa00005f7; interworking none
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      drops: illegal VLAN 0, illegal length 0
  EVPN: neighbor 24000, PW ID: evi 513, ac-id 513, state is up ( established )
    XC ID 0xc0000001
    Encapsulation MPLS
    Source address 209.165.200.225
    Encap type Ethernet, control word enabled
    Sequencing not set
    LSP : Up
EVPN          Local      Remote
-----------------------------------
Label         29045      1048577
MTU           1500       1500
Control word  enabled    enabled
AC ID         513        513
EVPN type    Ethernet    Ethernet
-----------------------------------

Router# show evpn internal-label vpn-id 513 detail
Tue Jan 28 13:22:19.110 GMT+4

VPN-ID  Encap Ethernet Segment Id          EtherTag  Label
-----------------------------------------------------------------
513      MPLS 0099.9900.0000.0000.9999        0      None
 Multi-paths resolved: FALSE (Remote all-active)
 Multi-paths Internal label: None
  EAD/ES  10.0.0.5                           0

513 MPLS   0099.9900.0000.0000.9999          513    24000
 Multi-paths resolved: TRUE (Remote all-active)
 Multi-paths Internal label: 24000
 EAD/ES 10.0.0.5                            0
 EAD/EVI (P) 10.0.0.5                      29104
 Summary pathlist:
0xffffffff (P) 10.0.0.5                    29104
-----------------------------------------------------------------

Router# show mpls forwarding labels 24000 hardware egress detail location 0/0/CPU0

Tue Jan 28 13:22:19.110 GMT+4
Label  Label       or ID              Interface       Switched
-----------------------------------------------------------------
24000  29104       EVPN:513            10.0.0.5       N/A
     Updated: Oct 18 13:14:02.193
     Version: 137839, Priority: 3
     Label Stack (Top -> Bottom): { 29104 }
     NHID: 0x0, Encap-ID: 0x140ea00000002, Path idx: 0, Backup path idx: 0, Weight: 0
     MAC/Encaps: 0/4, MTU: 0
     Packets Switched: 0



 LEAF - HAL pd context :
 sub-type : MPLS, ecd_marked:0, has_collapsed_ldi:0
 collapse_bwalk_required:0, ecdv2_marked:0,
HW Walk:
LEAF:
    PI:0x308de88fb8 PD:0x308de89058 rev:5554240 type: MPLS (2)
    LEAF location: LEM
    FEC key: 0x23e0220000d71
```

```
    label action: MPLS_NOP
LWLDI:
       PI:0x309faa82c8 PD:0x309faa8308 rev:5554239 p-rev:5459825 5459825  ldi type:EOS0_EOS1

        FEC key: 0x23e0220000d71 fec index: 0x0(0) num paths:2, bkup paths: 0
        Collpased IMP LDI: ECD_MARKED
        IMP pattern:3
        PI:0x309faa82c8 PD:0x309faa8308 rev:5554239 p-rev:5459825 5459825
        FEC key: 0x257c720000d71 fec index: 0x20000003(3) num paths:2
        Path:0  fec index: 0x20018f14(102164) DSP fec index: 0x200001f8(504),
                MPLS encap key: 0xf1b00000400140ea MPLS encap id: 0x400140ea Remote: 0
                Label Stack: 29104 16012  dpa-rev:55458217
        Path:1  fec index: 0x20018f15(102165) DSP fec index: 0x200001f9(505),
                MPLS encap key: 0xf1b00000400140eb MPLS encap id: 0x400140eb Remote: 0
                Label Stack: 29104 16012  dpa-rev:55458218

 REC-SHLDI HAL PD context :
ecd_marked:10, collapse_bwalk_required:0, load_shared_lb:0

    RSHLDI:
        PI:0x3093d16af8 PD:0x3093d16bc8 rev:5494421 dpa-rev:36033167 flag:0x1
        FEC key: 0x249e440000d71 fec index: 0x2001c169(115049) num paths: 1
        p-rev:5459825
        Path:0 fec index: 0x2001c169(115049) DSP fec index: 0x200001f8(504),

 LEAF - HAL pd context :
 sub-type : MPLS, ecd_marked:1, has_collapsed_ldi:0
 collapse_bwalk_required:0, ecdv2_marked:0,

HW Walk:
LEAF:
    PI:0x308de433b8 PD:0x308de43458 rev:5459864 type: MPLS (2)
    LEAF location: LEM
    FEC key: 0

    LWLDI:
        PI:0x309ffe9798 PD:0x309ffe97d8 rev:5459825 p-rev:4927729 4927729  ldi
type:IMP_EOS0_EOS1
        FEC key: 0x1a1c740000d71 fec index: 0x0(0) num paths:2, bkup paths: 0
        IMP LDI: ECD_MARKED  SERVICE_MARKED
        IMP pattern:3
        PI:0x309ffe9798 PD:0x309ffe97d8 rev:5459825 p-rev:4927729 4927729
        FEC key: 0x23e0220000d71 fec index: 0x20000002(2) num paths:2
        Path:0  fec index: 0x2001f8b4(129204) DSP fec index: 0x200001f8(504),
                MPLS encap key: 0xf1b0000040013ef0 MPLS encap id: 0x40013ef0 Remote: 0
                Label Stack: 16012  dpa-rev:35993054.  <<< LU Label>>>>
        Path:1  fec index: 0x2001f8b5(129205) DSP fec index: 0x200001f9(505),
                MPLS encap key: 0xf1b0000040013ef2 MPLS encap id: 0x40013ef2 Remote: 0
                Label Stack: 16012  dpa-rev:35993055  <<< LU Label>>>>

 REC-SHLDI HAL PD context :
ecd_marked:10, collapse_bwalk_required:0, load_shared_lb:0

    RSHLDI:
        PI:0x308dd32c38 PD:0x308dd32d08 rev:4927729 dpa-rev:35005343 flag:0x3
        FEC key: 0x1a1c740000d71 fec index: 0x20000813(2067) num paths: 2
        p-rev:4926086
        Path:0 fec index: 0x2001eefd(126717) DSP fec index: 0x200001f8(504),
        Path:1 fec index: 0x2001eefe(126718) DSP fec index: 0x200001f9(505),
LEAF - HAL pd context :
 sub-type : MPLS, ecd_marked:1, has_collapsed_ldi:0
 collapse_bwalk_required:0, ecdv2_marked:0,
HW Walk:
LEAF:
```

```
    PI:0x308dde33b8 PD:0x308dde3458 rev:4924403 type: MPLS (2)
    LEAF location: LEM
    FEC key: 0

    LWLDI:
        PI:0x308b04ea58 PD:0x308b04ea98 rev:4924400 p-rev:4924389 4924389 4924389 4924389
 ldi type:IMP_EOS0_EOS1
        FEC key: 0x1a75340000d71 fec index: 0x0(0) num paths:4, bkup paths: 0
        IMP LDI: ECD_MARKED
        IMP pattern:3
        PI:0x308b04ea58 PD:0x308b04ea98 rev:4924400 p-rev:4924389 4924389 4924389 4924389

        FEC key: 0x1a74720000d71 fec index: 0x200001f8(504) num paths:4
        Path:0  fec index: 0x2001ee86(126598) DSP:0x21
                MPLS encap key: 0xf1b0000040015878 MPLS encap id: 0x40015878 Remote: 0
                Label Stack: 16005  dpa-rev:34999715
        Path:1  fec index: 0x2001ee87(126599) DSP:0x22
                MPLS encap key: 0xf1b000004001587a MPLS encap id: 0x4001587a Remote: 0
                Label Stack: 16005  dpa-rev:34999716
        Path:2  fec index: 0x2001ee88(126600) DSP:0xc000002
                MPLS encap key: 0xf1b0000040016980 MPLS encap id: 0x40016980 Remote: 0
                Label Stack: 16005  dpa-rev:34989935
        Path:3  fec index: 0x2001ee89(126601) DSP:0xc000003
                MPLS encap key: 0xf1b00000400157fc MPLS encap id: 0x400157fc Remote: 0
                Label Stack: 16005  dpa-rev:34989936

SHLDI:
            PI:0x30927740c8 PD:0x3092774198 rev:4924389 dpa-rev:34999705 flag:0x0
            FEC key: 0x1a75340000d71 fec index: 0x200001ff(511) num paths: 4 bkup paths: 0

            p-rev:4924311 4924329 8779 4920854
            Path:0 fec index: 0x2001ee8f(126607) DSP:0x21 Dest fec index: 0x0(0)
            Path:1 fec index: 0x2001ee90(126608) DSP:0x22 Dest fec index: 0x0(0)
            Path:2 fec index: 0x2001ee91(126609) DSP:0xc000002 Dest fec index: 0x0(0)
            Path:3 fec index: 0x2001ee92(126610) DSP:0xc000003 Dest fec index: 0x0(0)
TX-NHINFO:
                PI: 0x308dc51298 PD: 0x308dc51318 rev:4924311 dpa-rev:34994174 Encap hdl:
0x3091632e98
                Encap id: 0x40010003 Remote: 0 L3 int: 1670 flags: 0x3
                npu_mask: 0x1 DMAC: 84:78:ac:2d:f8:1f

            TX-NHINFO:
                PI: 0x308dc51c20 PD: 0x308dc51ca0 rev:4924329 dpa-rev:34994264 Encap hdl:
0x30916332c8
                Encap id: 0x40010001 Remote: 0 L3 int: 1679 flags: 0x3
                npu_mask: 0x1 DMAC: d4:6d:50:7c:f9:4d

            TX-NHINFO:
                PI: 0x308dc51ff0 PD: 0x308dc52070 rev:8779 dpa-rev:61964 Encap hdl:
0x308e9f4980
                Encap id: 0x40010007 Remote: 0 L3 int: 1728 flags: 0x807
                npu_mask: 0x1 DMAC: 84:78:ac:2d:f8:22

            TX-NHINFO:
                PI: 0x308dc51480 PD: 0x308dc51500 rev:4920854 dpa-rev:34989846 Encap hdl:
0x308e9f4db0
                Encap id: 0x40010005 Remote: 0 L3 int: 1727 flags: 0x807
                npu_mask: 0x1 DMAC: 40:55:39:11:37:39

 LEAF - HAL pd context :
 sub-type : MPLS, ecd_marked:1, has_collapsed_ldi:0
 collapse_bwalk_required:0, ecdv2_marked:0,
HW Walk:
LEAF:
```

```
    PI:0x308dde35b8 PD:0x308dde3658 rev:4926089 type: MPLS (2)
    LEAF location: LEM
    FEC key: 0

    LWLDI:
        PI:0x308b04eb48 PD:0x308b04eb88 rev:4926086 p-rev:4924389 4924389 4924389 4924389
 ldi type:IMP_EOS0_EOS1
        FEC key: 0x1a75340000d71 fec index: 0x0(0) num paths:4, bkup paths: 0
        IMP LDI: ECD_MARKED
        IMP pattern:3
        PI:0x308b04eb48 PD:0x308b04eb88 rev:4926086 p-rev:4924389 4924389 4924389 4924389

        FEC key: 0x1a74820000d71 fec index: 0x200001f9(505) num paths:4
        Path:0  fec index: 0x2001ee81(126593) DSP:0x21
                MPLS encap key: 0xf1b000004001587c MPLS encap id: 0x4001587c Remote: 0
                Label Stack: 16006  dpa-rev:35002526
        Path:1  fec index: 0x2001ee82(126594) DSP:0x22
                MPLS encap key: 0xf1b000004001588a MPLS encap id: 0x4001588a Remote: 0
                Label Stack: 16006  dpa-rev:35002527
        Path:2  fec index: 0x2001ee83(126595) DSP:0xc000002
                MPLS encap key: 0xf1b0000040016964 MPLS encap id: 0x40016964 Remote: 0
                Label Stack: 16006  dpa-rev:34991843
        Path:3  fec index: 0x2001ee84(126596) DSP:0xc000003
                MPLS encap key: 0xf1b00000400157fe MPLS encap id: 0x400157fe Remote: 0
                Label Stack: 16006  dpa-rev:34991844
SHLDI:
            PI:0x30927740c8 PD:0x3092774198 rev:4924389 dpa-rev:34999705 flag:0x0
            FEC key: 0x1a75340000d71 fec index: 0x200001ff(511) num paths: 4 bkup paths: 0

            p-rev:4924311 4924329 8779 4920854
            Path:0 fec index: 0x2001ee8f(126607) DSP:0x21 Dest fec index: 0x0(0)
            Path:1 fec index: 0x2001ee90(126608) DSP:0x22 Dest fec index: 0x0(0)
            Path:2 fec index: 0x2001ee91(126609) DSP:0xc000002 Dest fec index: 0x0(0)
            Path:3 fec index: 0x2001ee92(126610) DSP:0xc000003 Dest fec index: 0x0(0)

            TX-NHINFO:
                PI: 0x308dc51298 PD: 0x308dc51318 rev:4924311 dpa-rev:34994174 Encap hdl:
0x3091632e98
                Encap id: 0x40010003 Remote: 0 L3 int: 1670 flags: 0x3
                npu_mask: 0x1 DMAC: 84:78:ac:2d:f8:1f

            TX-NHINFO:
                PI: 0x308dc51c20 PD: 0x308dc51ca0 rev:4924329 dpa-rev:34994264 Encap hdl:
0x30916332c8
                Encap id: 0x40010001 Remote: 0 L3 int: 1679 flags: 0x3
                npu_mask: 0x1 DMAC: d4:6d:50:7c:f9:4d

            TX-NHINFO:
                PI: 0x308dc51ff0 PD: 0x308dc52070 rev:8779 dpa-rev:61964 Encap hdl:
0x308e9f4980
                Encap id: 0x40010007 Remote: 0 L3 int: 1728 flags: 0x807
                npu_mask: 0x1 DMAC: 84:78:ac:2d:f8:22

            TX-NHINFO:
                PI: 0x308dc51480 PD: 0x308dc51500 rev:4920854 dpa-rev:34989846 Encap hdl:
0x308e9f4db0
                Encap id: 0x40010005 Remote: 0 L3 int: 1727 flags: 0x807
                npu_mask: 0x1 DMAC: 40:55:39:11:37:39
```

Support for DHCPv4 and DHCPv6 Client over BVI

The Support for DHCPv4 and DHCPv6 Client over the BVI feature allows you to configure DHCPv4 and DHCPv6 client on the Bridged Virtual Interface (BVI). You can configure a BVI, and request DHCP IPv4 or IPv6 address on the BVI. This allows your customer's device to have initial connectivity to your network without user intervention in the field. After the device is connected to your network, the customer devices can push a node-specific configuration with static IP addresses on a different BVI for customer deployment.

Configure DHCPv4 and DHCPv6 Client over BVI

Perform the following tasks to configure DHCPv4 and DHCPv6 client over BVI:

- Configure AC interface
- Configure L2VPN
- Configure BVI

Configuration Example

```
/* Configure AC interface */
Router# configure
Router(config)# interface tenGigE 0/5/0/1/1
Router(config-if)# bundle id 1 mode on
Router(config-if)# exit
Router(config)# interface Bundle-Ether1
Router(config-if)# no shut
Router(config-if)# exit
Router(config)# interface bundle-ether 1.100 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 100
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure L2VPN */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BVI
Router(config-l2vpn-bg)# bridge-domain bvi
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1.100
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)# routed interface BVI1
Router(config-l2vpn-bg-bd-bvi)# commit

/* Configure BVI */
Router# configure
Router(config)# interface BVI1
Router(config-if)# ipv4 address dhcp
Router(config-if)# ipv6 address dhcp
Router(config-if)# commit
```

Running Configuration

This section shows the DHCPv4 and DHCPv6 client over BVI running configuration.

```
interface TenGigE0/5/0/1/1
```

```
bundle id 1 mode on
!
interface Bundle-Ether1
!
interface Bundle-Ether1.100 l2transport
encapsulation dot1q 100
rewrite ingress tag pop 1 symmetric
!
l2vpn
bridge group BVI
  bridge-domain bvi
   interface Bundle-Ether1.100
   !
   routed interface BVI1
   !
  !
!
interface BVI1
ipv4 address dhcp
ipv6 address dhcp
!
```

Verification

The show output given in the following section display the details of DHCPv4 and DHCPv6 client over BVI configuration.

```
Router# show l2vpn bridge-domain
Legend: pp = Partially Programmed.
Bridge group: BVI, bridge-domain: bvi, id: 0, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 64000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 2 (2 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    BV1, state: up, BVI MAC addresses: 1
    BE1.100, state: up, Static MAC addresses: 0
  List of Access PWs:
  List of VFIs:
  List of Access VFIs:

Router# show dhcp ipv4 client

    Interface name         IP Address      Binding State         Lease Time Rem
----------------------    ------------    ---------------    ----------------------
BVI1                      172.16.0.2      BOUND             3598 secs (00:59:58)
-------------------------------------------------------------------------------------
Router# show dhcp ipv6 client

    Interface name         IPv6 Address         State              Lease Time Rem
----------------------    --------------    ---------------     --------------------
BVI1                      2000::1           BOUND               2591982
-------------------------------------------------------------------------------------

Router# show dhcp ipv4 client bvi1 detail

-----------------------------------------------------
Client Interface name         : BVI1
Client Interface handle       : 0x8804054
Client ChAddr                 : 008a.9628.ac8a
Client ID                     : BVI1.00:8a:96:28:ac:8a
Client State                  : BOUND
Client IPv4 Address (Dhcp)    : 172.16.0.2
Client IPv4 Address Mask      : 255.240.0.0
```

```
Client Lease Time Allocated   : 3600 secs (01:00:00)
Client Lease Time Remaining   : 3571 secs (00:59:31)
Client Selected Server Address: 172.16.0.1
Client Next Hop Address       : 0.0.0.0
-------------------------------------------------

Router# show dhcp ipv4 client BVI1 statistics

Client Interface name           : BVI1
-------------------------------------------------
     CLIENT COUNTER(s)          |        VALUE
-------------------------------------------------
Num discovers sent              :         44
Num requests sent               :          1
Num offers received             :          1
Num acks received               :          1
-------------------------------------------------

Router# show dhcp ipv6 client

    Interface name         IPv6 Address         State              Lease Time Rem
----------------------    --------------    ---------------     --------------------
BVI1                       2000::1          BOUND               2591685
-------------------------------------------------------------------------------------

Router# show dhcp ipv6 client statistics-all

Interface name          : BVI1
Interface handle        : 0x8804054
VRF                     : 0x60000000

     TYPE          |    TRANSMIT    |      RECEIVE   |     DROP       |
-----------------------------------------------------------------------
SOLICIT            |            17  |             0  |             0  |
ADVERTISE          |             0  |             1  |             0  |
REQUEST            |             1  |             0  |             0  |
REPLY              |             0  |             2  |             0  |
CONFIRM            |             0  |             0  |             0  |
RENEW              |             1  |             0  |             0  |
REBIND             |             0  |             0  |             0  |
RELEASE            |             0  |             0  |             0  |
RECONFIG           |             0  |             0  |             0  |
INFORM             |             0  |             0  |             0  |


     TIMER          |    STARTED     |    STOPPED     |     EXPIRED    |
-----------------------------------------------------------------------
     INIT           |             1  |             0  |             1  |
     VBIND          |             0  |             0  |             0  |
     RENEW          |             2  |             1  |             0  |
     REBIND         |             2  |             1  |             0  |
     RETRANS        |            19  |             3  |            16  |
     VALID          |             2  |             1  |             0  |
```

Configure DHCPv6 Client Options

You can configure different DHCPv6 client options to differentiate between clients as required. Configure different DHCPv6 client options to differentiate how a DHCPv6 client communicates with a DHCPv6 server. The different DHCPv6 client options that you can configure are:

- **DUID:** If the DUID DHCPv6 client option is configured on an interface, DHCPv6 client communicates with the DHCPv6 server through the link layer address.

- **Rapid Commit:** If the Rapid Commit DHCPv6 client option is configured on an interface, DHCPv6 client can obtain configuration parameters from the DHCPv6 server through a rapid two-step exchange (solicit and reply) instead of the default four-step exchange (solicit, advertise, request, and reply).
- **DHCP Options:** The various other DHCPv6 options that can be configured on a DHCPv6 client are:
  - **Option 15:** Option 15 is also known as the User Class option and it is used by a DHCPv6 client to identify the type or category of users or applications it represents.
  - **Option 16:** Option 16 is also known as the Vendor ID option and it is used by a DHCPv6 a client to identify the vendor that manufactured the hardware on which the client is running.
  - **Option 23:** Option 23 is also known as the Domain name Server (DNS) option provides a list of one or more IPv6 addresses of DNS recursive name servers to which a client's DNS resolver can send DNS queries.
  - **Option 24:** Option 24 is also known as the Domain List option and it specifies the domain search list that the client uses to resolve hostnames with the DNS.
- **DHCP Timers:** This option is used to set different timer value for DHCP client configurations. The various DHCP timer options are:
  - **Release-timeout:** It is used to set retransmission timeout value for the initial release message.
  - **Req-max-rt:** It is used to set the maximum retransmission timeout value for the request message.
  - **Req-timeout:** It is used to set the initial request timeout value of the request message.
  - **Sol-max-delay:** It is used to set the maximum delay time of the first solicit message.
  - **Sol-max-rt:** It is used to set the maximum solicit retransmission time.
  - **Sol-time-out:** It is used to set the intial timeout value of the solicit message.

Configuration Example

Perform this task to configure DHCPv6 client options on a BVI interface.

```
Router# configure
Router(config)# interface BVI 10
Router(config-if)# ipv6 address dhcp-client-options
Router(config-dhcpv6-client)# duid linked-layer-address
Router(config-dhcpv6-client)# rapid-commit
Router(config-dhcpv6-client)# timers release-timeout 3
Router(config-dhcpv6-client)# timers sol-max-delay 1
Router(config-dhcpv6-client)# timers sol-time-out 1
Router(config-dhcpv6-client)# timers sol-max-rt 120
Router(config-dhcpv6-client)# timers req-max-rt 30
Router(config-dhcpv6-client)# timers req-timeout 1
Router(config-dhcpv6-client)# commit
```

Verification

To verify the DHCPv6 client options, use the **show dhcp ipv6 client BVI10 detail** command.

```
Router# show dhcp ipv6 client BVI10 detail
Wed Jun  10 16:19:21.272 IST

-------------------------------------------------------
```

```
Client Interface name : MgmtEth0/0/CPU0/1
Client Interface handle : 0x4040
Client MACAddr : 02f0.2b39.44be
Client State : BOUND
Client Link Local Address : fe80::f0:2bff:fe39:44be
Client IPv6 Address (Dhcp) : 600:1::12
Lease Remaining (in secs) : 74
DUID : 0003000102f02b3944be

Client Configuration
Timers
SOL_MAX_DELAY : 1 secs (00:00:01)
SOL_TIMEOUT : 1 secs (00:00:01)
SOL_MAX_RT : 120 secs (00:02:00)
REQ_TIMEOUT : 1 secs (00:00:01)
REQ_MAX_RT : 30 secs (00:00:30)
REL_TIMEOUT : 3 secs (00:00:01)

Options
RAPID-COMMIT : True
USER-CLASS : ciscoupnnp
VENDOR-CLASS : vendor
DNS-SERVERS : True
DOMAIN-LIST : True

DUID Type : DUID_LL

Server Information
Server Address : fe80::d2:a1ff:feb2:3b9f
Preference : 0
DUID : 000300010206826e2e00
Status : SUCCESS
IA-NA
Status : SUCCESS
IAID : 0x40400001
T1 : 60 secs (00:01:00)
T2 : 96 secs (00:01:36)
IA-ADDR
IA NA Address : 600:1::12
Preferred Time : 120 secs (00:02:00)
Valid Time : 120 secs (00:02:00)
Flags : 0x0
```

Related Topics

- Support for DHCPv4 and DHCPv6 Client over BVI, on page 334

Associated Commands

- show l2vpn bridge-domain
- show dhcp ipv4 client
- show dhcp ipv6 client
- show dhcp ipv4 client bvi

Layer 2 fast reroute

Table 29: Feature History Table

Feature Name	Release Information	Feature Description
Layer 2 Fast Reroute	Release 7.3.1	In the event of a link failure, this feature enables the router to switch traffic quickly to a precomputed loop-free alternative (LFA) path by allocating a label to the incoming traffic. This minimizes the traffic loss ensuring fast convergence. This feature introduces the **convergence reroute** command.

The L2 FRR feature safeguards against traffic loss when a PE-CE link fails before the remote PE receives the mass withdraw message indicating the link is down. On the PE router with the failed link to the CE, traffic is redirected through a backup path to its EVPN Multi-home (MH) peer router, which then forwards the traffic to the CE.

In the control plane, the MAC address remains associated with the local bridge port ESI, but in the data-path L2FRR activates the backup path for the MAC address which has been pre-populated on the AC segment.

As a consequence, **show** commands keep displaying the MAC address - bridge port association even after the AC is down.

Through this permanent association of hosts (or MAC addresses) to an AC or bridge port, the L2 MAC-IP routes are retained on PE1 even on failure. In addition to displaying the retained MAC address - bridge port association, the **show** commands on PE1 continues to display the retained ARP entries and L2 MAC-IP routes. The AC service state displays the **Down** state.

Managing MAC on layer 2 FRR-enabled attachment circuits (AC)

MAC addresses remain intact when a layer 2 FRR-enabled attachment circuit goes down.

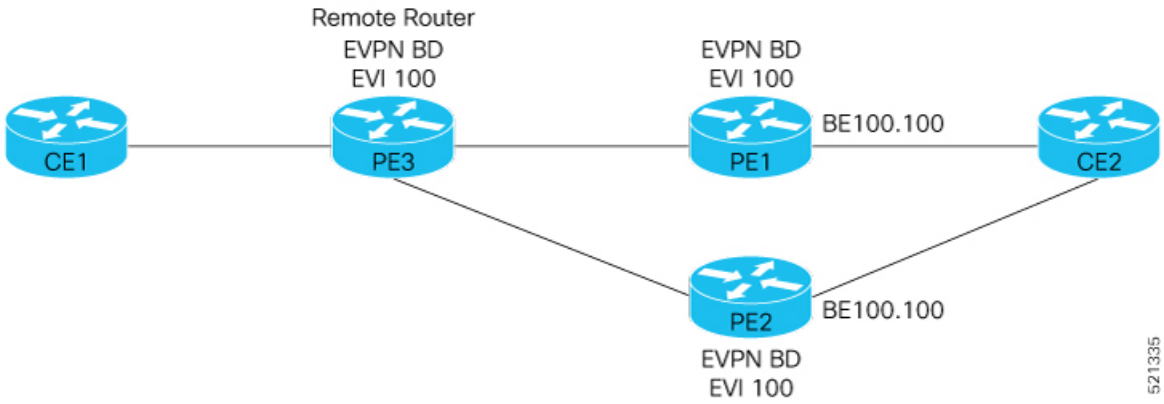
In ELAN configurations, retaining locally learned MAC addresses on layer 2 FRR-enabled circuits during downtime is crucial. This allows for quick identification and redirection of unicast traffic to the customer edge (CE) through MAC lookup.

Benefits

This feature provides fast and predictable convergence:

- Fast and predictable convergence.
- Manual configuration for predictable failover behavior.
- No need to change the topology.

Figure 54: Layer 2 Fast Reroute



In this topology:

- CE2 is multi-homed to PE1 and PE2.
- PE1 and PE2 are in EVPN active-active or single-active mode. They are connected to a remote router PE3 over the MPLS core network.
- CE1 is connected to PE3.
- Both PE1 and PE2 are enabled with L2 FRR. An FRR label is added per EVI for the backup path.

Consider a traffic flow from CE1 to CE2 in a regular scenario:

- The traffic is sent from CE1 to PE3.
- PE3 distributes the traffic over PE1 and PE2.
- PE1 and PE2 send the traffic to CE2.

When the PE3, CE2 traffic over PE1 takes the PE2 path which is PE1-PE2 backup path the traffic is forwarded to CE2. This traffic redirect lasts until PE3 updates the routes and sends to PE2.

- When L2FRR is enabled on PE1 and PE2, both PE1 and PE2 have the FRR labels. The logical backup path is pre-programmed in the hardware. When PE1 detects a failure on the access side (CE2), instead of sending traffic to CE2, PE1 sends the traffic over the pre-programmed backup path to PE2.
- All incoming traffic to PE1 is redirected to PE2 using this FRR label.
- PE1 encapsulates all the traffic with the FRR label of PE2 and forwards the traffic to PE2.
- Upon receiving the FRR label, PE2 forwards the traffic to CE2 through the attachment circuit (AC), even if the AC is in a blocking state.

Associated Commands

- **convergence reroute**
- **show evpn ethernet-segment**
- **show evpn evi**
- **show evpn evi ead private**

Restrictions for layer 2 fast reroute

This feature is supported on:

- BGP MPLS-based EVPN E-LAN.
- PE devices in EVPN active-active or single-active mode.
- Unicast traffic.
- This feature is not supported on BUM traffic.

Configure layer 2 fast reroute for E-LAN service

Configure L2 FRR on a PE router in the E-LAN EVPN multi-homing network.

Procedure

Step 1 Associate the Ethernet segment with the bundle interface and enable L2 FRR using the **convergence reroute** command. Configure this command on both PE1 and PE2 with in the specified topology map to enable L2FRR.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface Bundle-Ether4.1
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)# evi 1
Router(config-l2vpn-bg-bd-evi)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# bridge group bg2
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface Bundle-Ether4.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evi 2
Router(config-l2vpn-bg-bd-evi)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)#
```

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# exit
Router(config-evpn-instance)# exit
Router(config-evpn)# evi 2
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# exit
Router(config-evpn-instance)# exit
Router(config-evpn)# interface Bundle-Ether4
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# convergence
Router(config-evpn-ac-es-conv)# reroute
Router(config-evpn-ac-es)# exit
```

```
Router(config-evpn-ac# exit
Router(config-evpn)# exit
Router(config)# exit
```

Step 2 Verify the L2 FRR configuration.

```
Router# show evpn ethernet-segment carving detail
...Ethernet Segment Id      Interface                          Nexthops
----------------------- ---------------------------------- --------------------
0040.0000.0000.0000.0001 BE4                                4.5.6.7
                                                            5.6.7.8
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether4
     Interface MAC  : 00c9.c654.9a04
     IfHandle       : 0x7800008c
     State          : Up
     Redundancy     : Not Defined
  ESI ID            : 1
  ESI type          : 0
     Value          : 0040.0000.0000.0000.0001
  ES Import RT      : 4000.0000.0000 (from ESI)
  Topology          :
     Operational    : MH, Single-active
     Configured     : Single-active (AApS)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence    : Reroute
  Peering Details   : 2 Nexthops
     4.5.6.7 [MOD:P:00:T]
     5.6.7.8 [MOD:P:00:T]
  Service Carving Synchronization:
     Mode           : NTP_SCT
     Peer Updates   :
                 4.5.6.7 [SCT: 2025-01-22 17:01:01.1737583]
                 5.6.7.8 [SCT: 2025-01-22 17:00:36.1737583]
  Service Carving Results:
     Forwarders     : 2
     Elected        : 1
            EVI E   :        2
     Not Elected    : 1
            EVI NE  :        1
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flush msg     : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Revert timer      : 0 sec [not running]
  HRW Reset timer   : 5 sec [not running]
  Local SHG label   : 24008
     IPv6_Filtering_ID : 1:16
  Remote SHG labels : 1
              24007 : nexthop 5.6.7.8
  Access signal mode: Bundle OOS
  ...

Router# show l2vpn forwarding interface BE4.1 private location 0/RP0/CPU0
Wed Jan 22 17:02:01.387 EST

Xconnect ID 0xc0000002
```

```
  Xconnect info:
    xcon_status=Up, xcon_bound=TRUE, switching_type=0, data_type=12
    xcon_name=

    Object: XCON
    Base info: version=0xaabbcc13, flags=0x3110, type=2, object_id=UNSPECIFIED, reserved=0


  AC info:
    xcon_id=0xc0000002, ifh=0x7800008c, subifh=0x78000096, ac_id=0, ac_type=21, status=Bound

    ac_mtu=1500, iw_mode=1, adj=150.0.0.120+Bundle-Ether4,
    r_aps_channel=FALSE, prot_exclusion=FALSE
    rg_id=0, ro_id=0x0000000000000000
    evpn internal label = None
    E-Tree = Root
    FXC local-switch AC xcid = 0x0 (Invalid)
    FXC local-switch PW xcid = 0xffffffff (Invalid)
    EVPN MP route flags = 0x0
    Statistics:
      packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 0
      bytes: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 0
      MAC move: 0
      packets dropped: PLU 0, tail 0
      bytes dropped: PLU 0, tail 0

    Object: AC
    Base info: version=0xaabbcc11, flags=0x0, type=3, object_id=0x10001000000002d8|v9,
reserved=0

    AC Backup info:
      VC label: 24004
      Local VC label: 0
      Backup Pseudowire XC ID: 0x0
      Statistics:
        packets: received 0, sent 0
        bytes: received 0, sent 0
        packets dropped: PLU 0, tail 0, out of order 0
        bytes dropped: PLU 0, tail 0, out of order 0

      Object: AC_BACKUP
      Base info: version=0xaabbcc39, flags=0x0, type=43, object_id=0x1000100000000300|v1,
reserved=0


      Nexthop info:
        nh_addr=5.6.7.8,
        ecd_plat_data_valid=TRUE, ecd_plat_data_len=104, plat_data_size=232
        child_count=0, child_evpn_ole_count=2, child_mac_count=0, child_pwhe_mp_count=0,
child_ac_backup_count=2,
        child_vni_count=0, child_ifl_count=0, child_sg_count=0

        Object: NHOP
       Base info: version=0xaabbcc14, flags=0x4010, type=7, object_id=0x10001000000002f4|v5,
 reserved=0


  Bridge port info:
    xcon_id=0xc0000002, bridge_id=0, shg_id=0, mac_limit=disabled, 262144, mac_limit_action=0

    aging_timeout=300, mvrp_seq_number=0, learn_key=0
    bp_segl_type=0x3, mtu=1500
    is_flooding_disabled=FALSE, is_mac_learning_disabled=FALSE,
is_mac_port_down_flush_disabled=FALSE,
```

```
   EVPN ESI ID: 0
   EVPN SHG Local Label: None
   EVPN SHG Remote Labels: 0
    MAC learning: enabled
    Software MAC learning: enabled
    MAC port down flush: enabled
    Flooding:
      Broadcast & Multicast: enabled
      Unknown unicast: enabled
    MAC aging time: 300 s, Type: inactivity
    MAC limit: none
    MAC Secure: disabled, Logging: disabled, Accept-Shutdown: enabled
    DHCPv4 snooping: profile not known on this node, disabled
    Dynamic ARP Inspection: disabled, Logging: disabled
    IP Source Guard: disabled, Logging: disabled
    IGMP snooping profile: profile not known on this node
    MLD snooping profile: profile not known on this node
    Router guard disabled
    vES:disabled
    Etree Leaf:disabled
    STP participating: disabled
    Storm control: disabled
    Main port: Bundle-Ether4, MSTI: 2

    Object: BRIDGE_PORT
    Base info: version=0xaabbcc1a, flags=0x0, type=12, object_id=0x10001000000002d9|v6,
reserved=0
```

EVPN Preferred Nexthop

Table 30: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Preferred Nexthop	Release 7.3.1	With this feature, you can set an active and backup path, in a dual-homed mode based on the nexthop IP address, thereby allowing greater control over traffic patterns. If you are unable to use single-active mode due to hardware, topology, or technological limitations, this feature enables you to direct traffic to a specific remote PE. This feature introduces the **preferred nexthop** command.

The EVPN Preferred Nexthop feature allows you to choose a primary nexthop and backup nexthop among the remote PE devices in dual-homed mode. By default, in an all-active dual-homed topology, traffic is load balanced using ECMP across both remote PE devices.

Configure the **preferred-nexthop** command when you want to direct traffic to one specific remote PE, and you are unable to use single-active mode due to hardware, topology, or technological limitations. The router allocates an internal label and will not allocate or consume ECMP FEC. The internal label enables fast switchover to backup PE when the primary link fails.

When remote PEs are operating in EVPN all-active mode, configure the **preferred-nexthop** command per EVI to choose an active and backup path based on the nexthop IP address. You can set the highest IP address as primary, which results in the lower IP address as a backup or vice versa.This feature provides you greater

control over traffic patterns, that is to achieve symmetric traffic flow, and to allow support when a topology cannot support an all-active remote PE. Preferred nexthop is supported for native EVPN, EVPN VPWS, and EVPN PWHE. This feature supports a topology that has only two remote nexthops.

Configure EVPN Preferred Nexthop

Perform the following task to configure EVPN preferred nexthop.

Configuration Example

This example shows the configuration of highest IP address as the preferred nexthop.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-evi)# preferred-nexthop highest-ip
Router(config-evpn-evi)# commit
```

This example shows the configuration of lowest IP address as the preferred nexthop.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-evi)# preferred-nexthop lowest-ip
Router(config-evpn-evi)# commit
```

This example shows the configuration of preferred nexthop using the **modulo** keyword.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-evi)# preferred-nexthop modulo
Router(config-evpn-evi)# commit
```

Running Configuration

This section shows the EVPN preferred nexthop running configuration.

```
/* Configuration of highest IP address as the preferred nexthop */
evpn
 evi 100
  preferred-nexthop highest-ip
!

/* Configuration of lowest IP address as the preferred nexthop */
evpn
 evi 100
  preferred-nexthop lowest-ip
!

/* Configuration of preferred nexthop using the modulo keyword */
evpn
 evi 100
  preferred-nexthop modulo
```

Verification

The output shows that the Highest IP is selected as primary (P) and the lowest IP as backup (B). The path selection is programmed in CEF.

```
Router#show evpn evi vpn-id 100 detail
Mon Oct 26 14:00:51.459 EDT

VPN-ID     Encap      Bridge Domain                 Type
---------- ---------- ----------------------------- -------------------
100        MPLS       bd100                         EVPN
…
   Preferred Nexthop Mode: Highest IP

Router#show evpn internal-label vpn-id  100 detail
Mon Oct 26 14:01:46.665 EDT

VPN-ID     Encap  Ethernet Segment Id         EtherTag     Label
---------- ------ -------------------------- ----------   --------
100        MPLS   0100.0000.acce.5500.0100    0            28120
      Multi-paths resolved: TRUE (Remote all-active) (Preferred NH, Highest IP)
      Multi-paths Internal label: 28120
      EAD/ES      192.168.0.1                               0
                        192.168.0.3                                   0
      EAD/EVI     192.168.0.1                               28099
                       192.168.0.3                                 28099
      Summary pathlist:
     0xffffffff (P) 192.168.0.3                               28099
      0xffffffff (B) 192.168.0.1                               28099

Router#show cef mpls local-label 28120 eOS
Mon Oct 26 14:04:10.851 EDT
Label/EOS 28120/1, version 56, internal 0x1000001 0x30 (ptr 0x4d3ba2a8) [1], 0x0 (0x0),
0x208 (0x4e6502c0)
Updated Oct 26 14:00:31.225
…
   via 192.168.0.3/32, 6 dependencies, recursive [flags 0x0]
    path-idx 0 NHID 0x0 [0x4d3bb58c 0x0], Internal 0x4e7890f8
    recursion-via-/32
    next hop 192.168.0.3/32 via 28103/0/21
     local label 28120
     next hop 27.27.27.3/32 Gi0/2/0/7    labels imposed {ImplNull 28099}
   via 192.168.0.1/32, 6 dependencies, recursive, backup (Local-LFA) [flags 0x300]
    path-idx 1 NHID 0x0 [0x4d3bb454 0x0]
    recursion-via-/32
    next hop 192.168.0.1/32 via 28105/0/21
     local label 28120
     next hop 26.26.26.1/32 Gi0/2/0/6    labels imposed {ImplNull 28099}
```

EVPN Access-Driven DF Election

Table 31: Feature History Table

Feature Name	Release Information	Feature Description

EVPN Access-Driven DF Election	Release 7.3.1	This feature enables the access network to control EVPN PE devices by defining the backup path much before the event of a link failure, thereby reducing the traffic loss. The following keywords are added to the **service-carving** command: • **preference-based** • **access-driven**

This feature includes a preference-based and access-driven DF election mechanism.

In a preference-based DF election mechanism, the weight decides which PE is the DF at any given time. You can use this method for topologies where interface failures are revertive. However, for topologies where an access-PE is directly connected to the core PE, use the access-driven DF election mechanism.

When access PEs are configured in a non-revertive mode, the access-driven DF election mechanism allows the access-PE to choose which PE is the DF.

Consider an interface in an access network that connects PE nodes running Multichassis Link Aggregation Control Protocol (mLACP) and the EVPN PE in the core. When this interface fails, there may be a traffic loss for a longer duration. The delay in convergence is because the backup PE is not chosen before failure occurs.

The EVPN Access-Driven DF Election feature allows the EVPN PE to preprogram a backup PE even before the failure of the interface. In the event of failure, the PE node will be aware of the next PE that will take over. Thereby reducing the convergence time. Use the *preference df weight* option for an Ethernet segment identifier (ESI) to set the backup path. By configuring the weight for a PE, you can control the DF election, thus define the backup path.

Restrictions

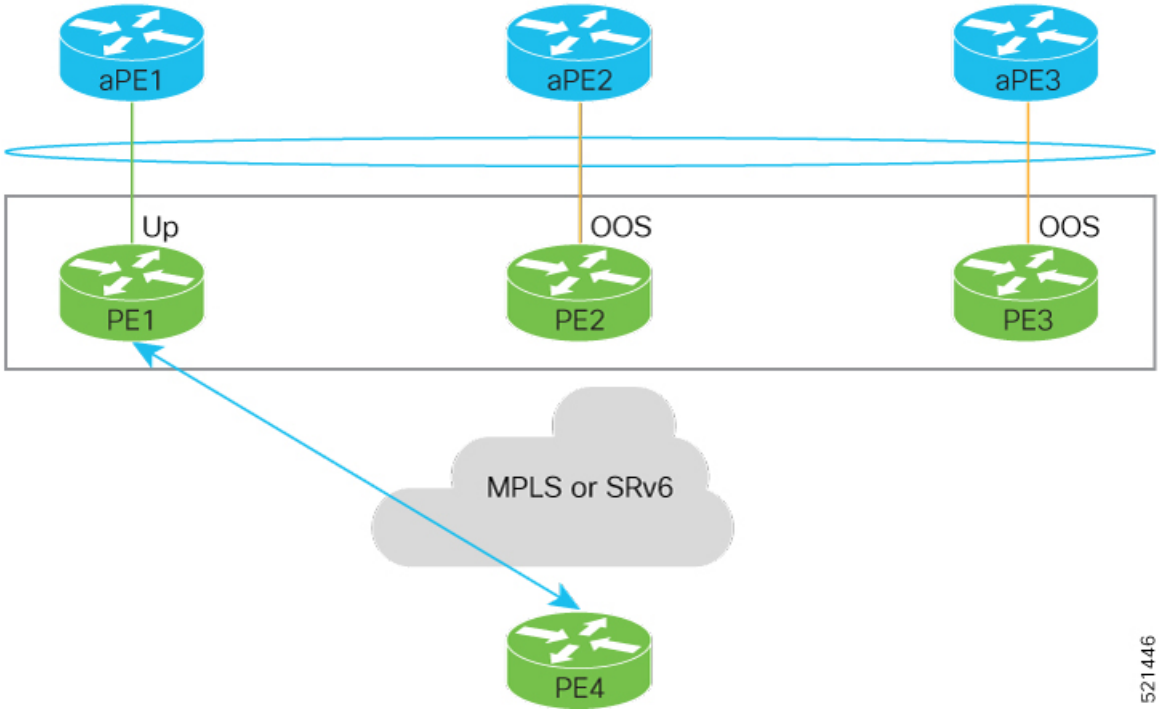
- The feature is supported only in an EVPN-VPWS scenario where EVPN PEs are in the port-active mode.
- The bundle attached to the ethernet segment must be configured with **lacp mode active**.

  LACP mode on is not supported.

Topology

Let’s understand the feature on how the backup path is precomputed with the following topology.

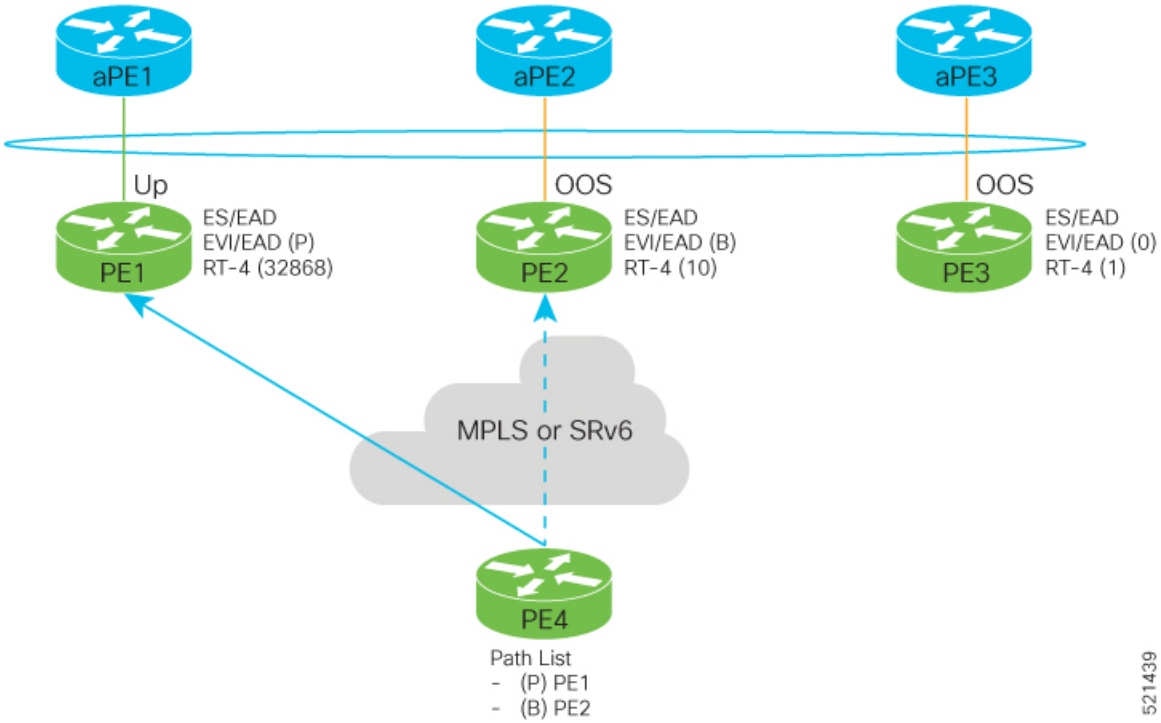
Figure 55: EVPN Access-Driven DF Election



- PE1, PE2, and PE3 are PEs for the EVPN core network.
- aPE1, aPE2, and aPE3 are their access PE counterparts and configured in a multichassis link aggregation group (MCLAG) redundancy group. Only one link among the three is active at any given time. aPE1, aPE2, and aPE3 are in a non-revertive mode.
- PE1 is directly connected to aPE1, PE2 to aPE2, and PE3 to aPE3. EVPN VPWS is configured on the PE devices in the core.
- All PE devices are attached to the same bundle and shares the same ethernet segment identifier.
- PE1, PE2, and PE3 are configured with a weight of 100, 10, and 1 respectively.

Traffic Flow

In this example, consider a traffic flow from a host connected to PE4 to the host connected to the access PE.



- aPE1-PE1 interface state is up. The aPE2-PE2 and aPE3-PE3 remains in OOS state.
- The traffic is sent from PE4 to aPE1 through PE1 as the PE1 is configured with a highest weight of 100.
- The highest weight is modified by adding 32768 to the configured weight. For example, the weight of PE1 is 100, 32768 is added to this weight. Hence, 32868 is advertised to the peer PEs.
- The highest weight is advertised as P-bit, which is primary. The next highest weight is advertised as B-bit, which is secondary. The lowest weight as non-DF (NDF).
- When the EVPN PE devcies are of same weight, the traffic is sent based on the IP address. Lowest IP address takes the precedence.
- Only one PE indicates that the state of the bundle for the Ethernet Segment is up. For all other PEs, the Ethernet Segment is standby and the bundle is in OOS state.
- All PE devices are aware of the associated next hop and weights of their peers.

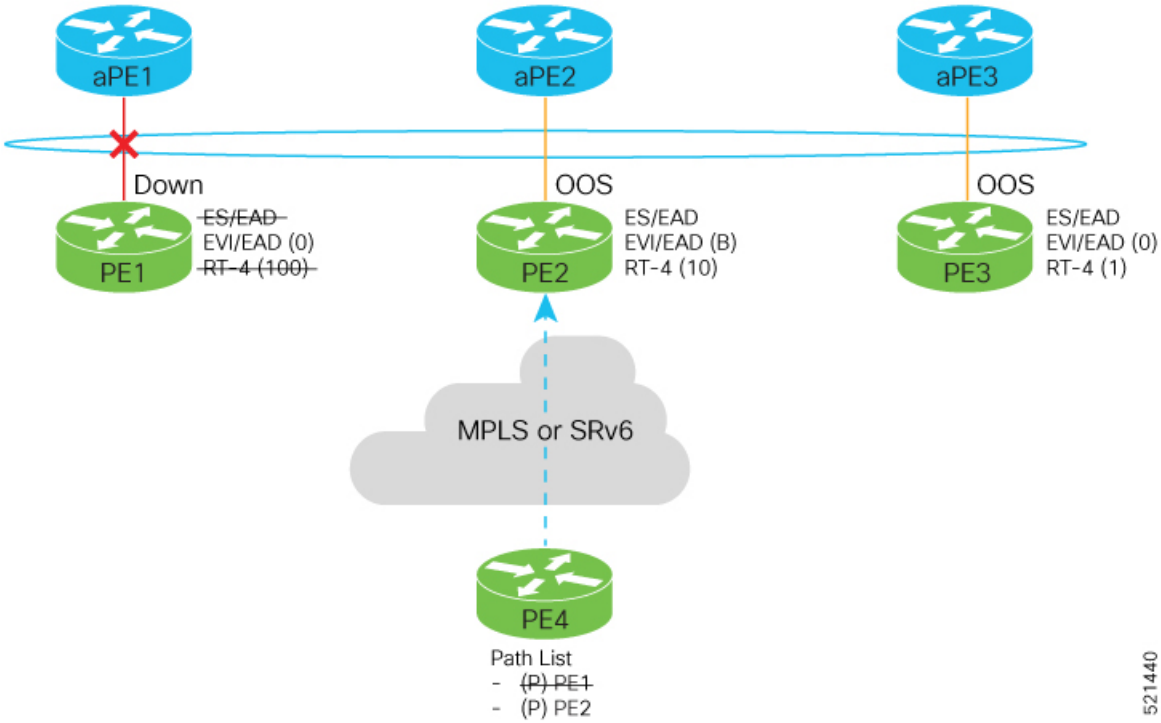
Failure and Recovery Scenarios

The weights configured on the EVPN PE devices cascade in the same order as the protection mechanism on the access side PEs:

- During the network failure, the redundancy ordering for the access PEs is aPE1, aPE2, aPE3.
- The weights of PE1 through PE3 are weight of PE1 > weight of PE2 > weight of PE3.
- If this ordering is not satisfied, the network will eventually converge, but it will not be as efficient as if the weights are ordered correctly.

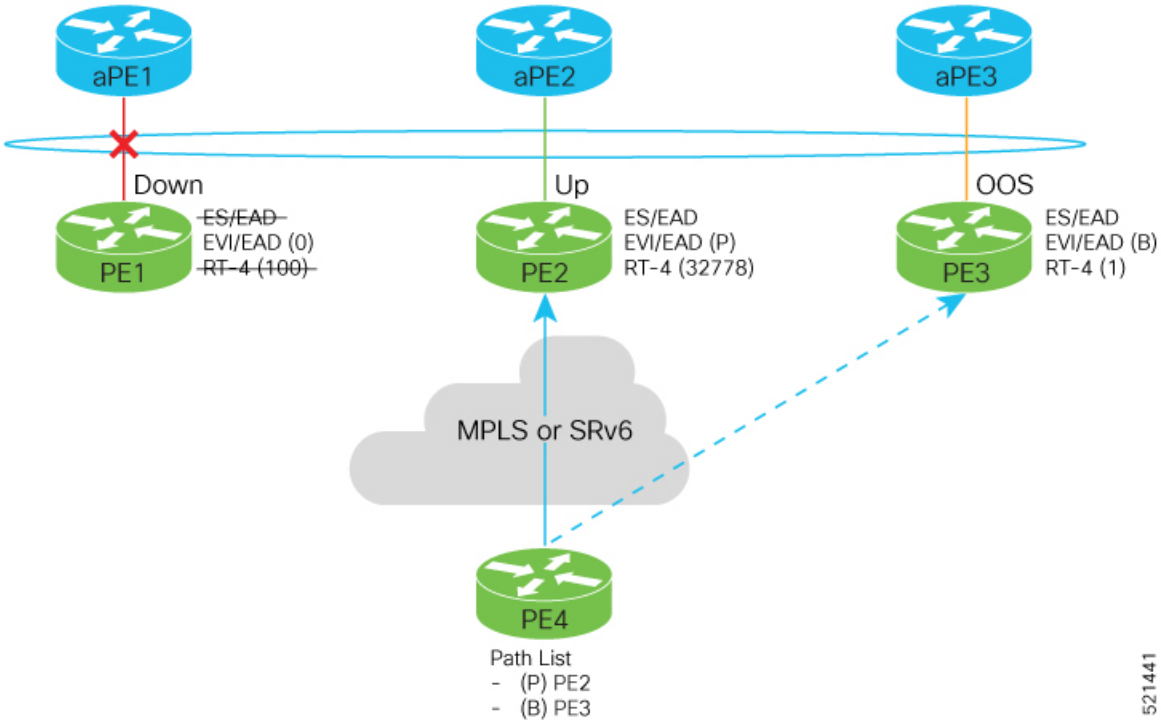
Scenario - 1

Consider a scenario where the aPE1-PE1 interface is down.



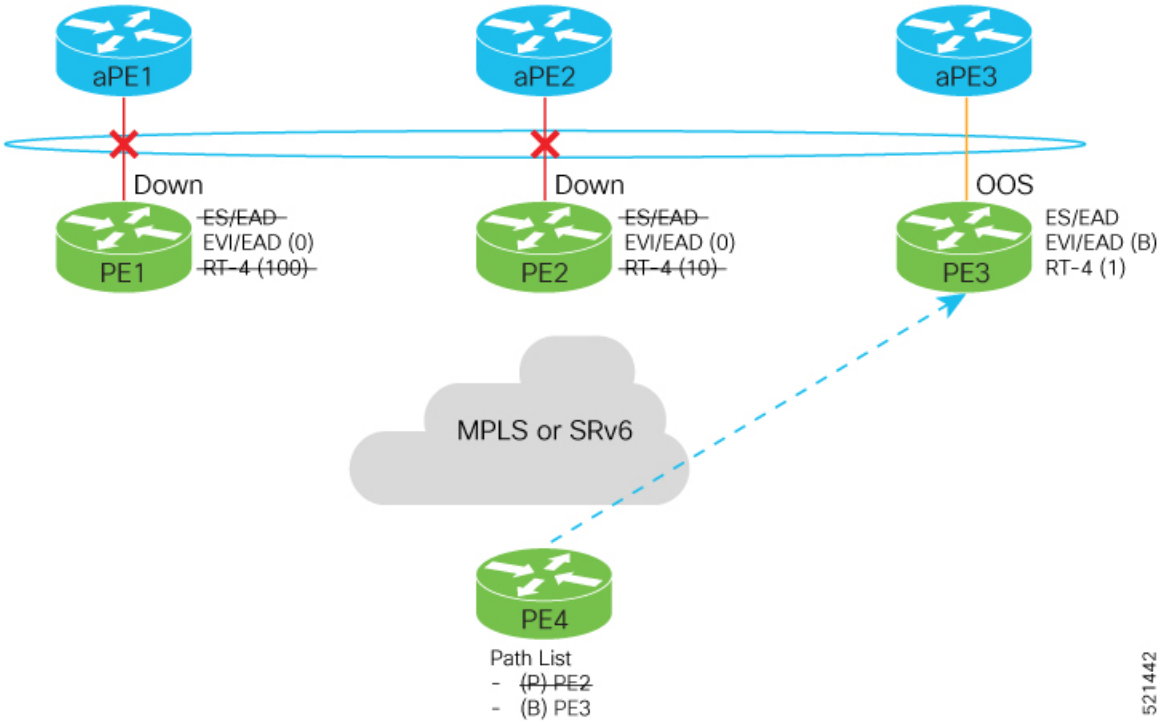
When aPE1-PE1 interface is down, the PE1 withdraws the EAD/ES route, and the traffic is sent through the backup path, which is PE2.

The aPE2-PE2 becomes the primary with a weight of 32778, and aPE3-PE3 becomes the backup. The aPE2-PE2 advertises P-bit to PE4. aPE3-PE3 advertises the B-bit to PE4.

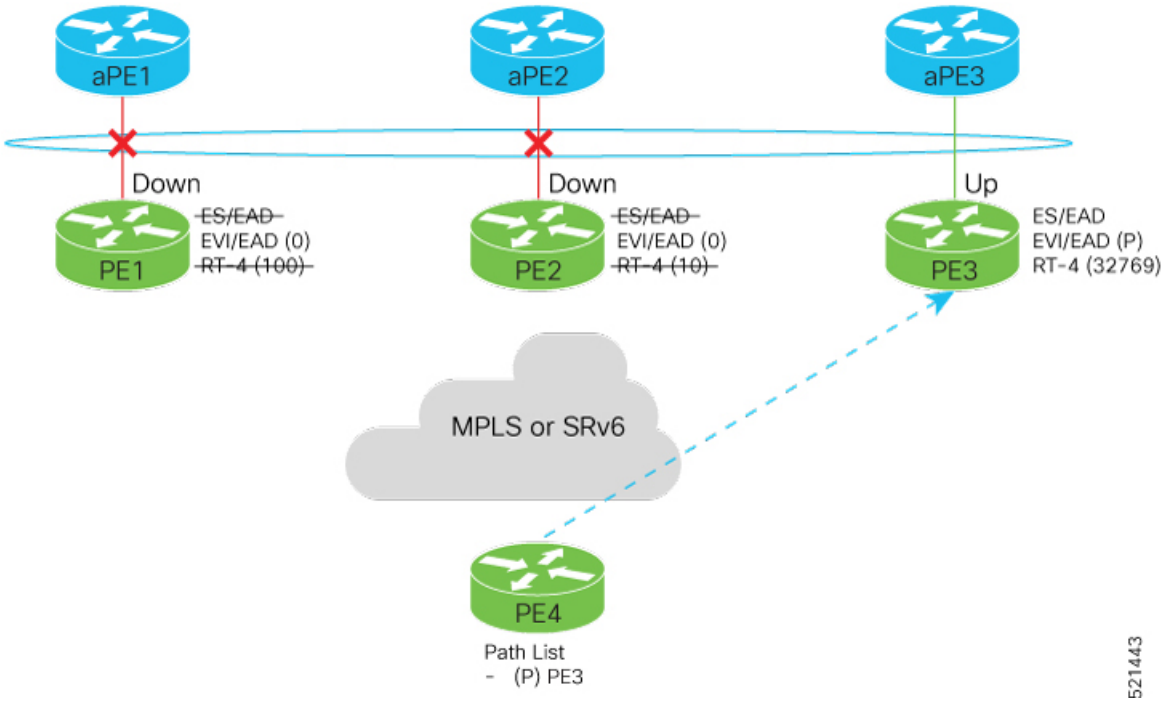


Scenario - 2

Consider a scenario where aPE2-PE2 interface is also down.

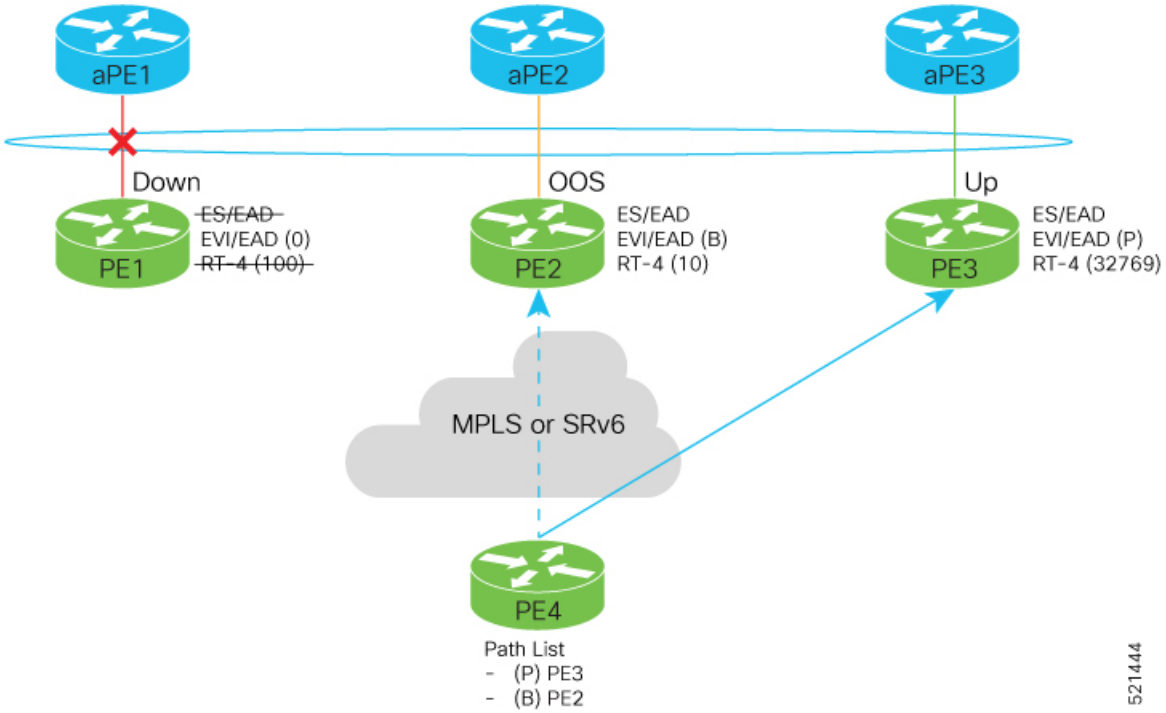


When the aPE2-PE2 interface is also down, the traffic is sent through aPE3-PE3 link. aPE3-PE3 becomes the primary path with a weight of 32769.



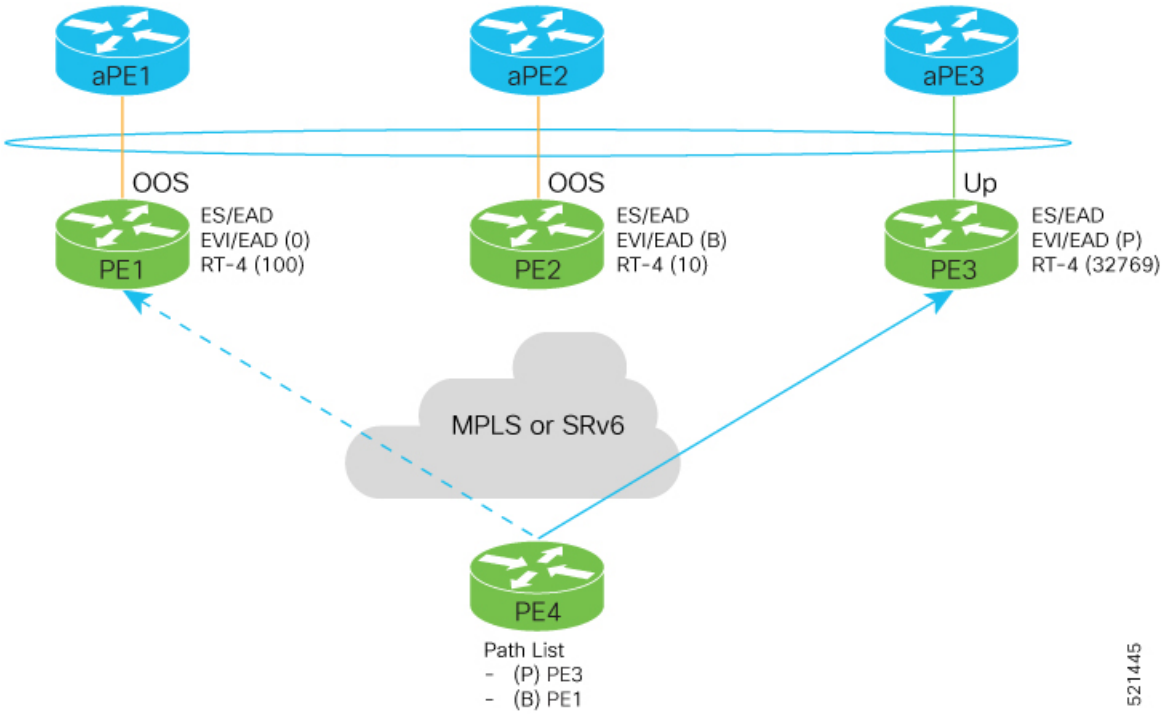
Scenario - 3

When the aPE2-PE2 interface comes up, the aPE3-PE3 link still remains the primary path. aPE2-PE2 interface becomes the backup path with a weight of 10.



Scenario - 4

When the aPE1-PE1 interface comes up, the aPE3-PE3 link remains the primary path with a weight of 32769. aPE1-PE1 interface becomes the backup path with a weight of 100. The aPE2-PE2 interface becomes NDF with a weight of 10.



Configure EVPN Access-Driven DF Election

Perform the following tasks to configure EVPN Access-Driven DF Election feature:

- Configure EVPN access-driven DF election on PE1, PE2, and PE3
- Configure LACP on aPE1, aPE2, and aPE3
- Configure EVPN-VPWS for PE1, PE2, and PE3

  See the *EVPN Virtual Private Wire Service (VPWS)* chapter on how to configure EVPN-VPWS.

Configuration Example

- All PE devices are configured with different weights. PE1, PE2, and PE3 are configured with a weight of 100, 10, and 1 respectively.
- The bundle attached to the ethernet segment is configured with **lacp mode active**.
- EVPN VPWS is configured on the PE devices.

```
/* Configure EVPN access-driven DF election on PE1, PE2, and PE3 */

/*  PE1 Configuration */
Router#configure
Router(config)#evpn
```

```
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 100
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit

/* PE2 Configuration */
Router#configure
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 10
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit

/* PE3 Configuration */
Router#configure
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 1
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit
```

Configure LACP on aPE1, aPE2, and aPE3

```
/* aPE1 Configuration */
Router#configure
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/40
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 10000
Router(config-if)description Connection to PE1
Router(config-if)commit

/* aPE2 Configuration */
Router#configure
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/39
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 20000
Router(config-if)description Connection to PE2
Router(config-if)commit

/* aPE3 Configuration */
Router#configure
```

```
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/38
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 30000
Router(config-if)description Connection to PE3
Router(config-if)commit
```

Running Configuration

This section shows the running configuration of EVPN Access-Driven DF Election feature.

```
/* PE1 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 100
    access-driven
   !
 !

/* PE2 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 10
    access-driven
   !
 !

/* PE3 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 1
    access-driven
   !
 !

/* aPE1 Configuration */

interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/40
 bundle id 10 mode active
 bundle port-priority 10000
 description Connection to PE1
!

/* aPE2 Configuration */
```

```
interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/39
 bundle id 10 mode active
 bundle port-priority 20000
 description Connection to PE2
!

/* aPE3 Configuration */

interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/40
 bundle id 10 mode active
 bundle port-priority 30000
 description Connection to PE3
!
```

Verification

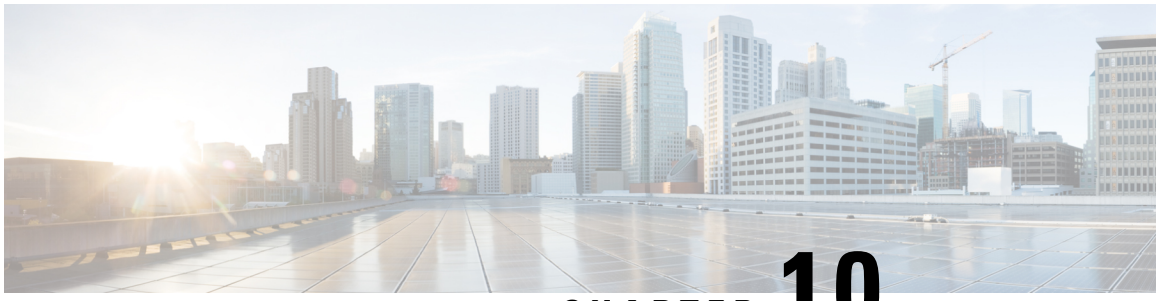
Verify that you have configured the EVPN Access-Driven DF Election feature successfully.

```
Router#show evpn ethernet-segment detail
Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0001.0001.0001.1b01.001b BE1                               192.168.0.1
                                                           192.168.0.3
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether1
     Interface MAC  : 02ef.af8d.8008
     IfHandle       : 0x00004190
     State          : Up
     Redundancy     : Active
  ESI type          : 0
     Value          : 01.0001.0001.1b01.001b
  ES Import RT      : 0100.0100.011b (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : Port-Active
  Service Carving   : Preferential
     Multicast      : Disabled
  Convergence       :
  Peering Details   : 2 Nexthops
     192.168.0.1 [PREF:P:d6ce:T] >> Weight in hexadecimal
     192.168.0.3 [PREF:P:457]
  Service Carving Synchronization:
     Mode           : NONE
     Peer Updates   :
  Service Carving Results:
     Forwarders     : 24
     Elected        : 6
     Not Elected    : 0
  EVPN-VPWS Service Carving Results:
     Primary        : 18
     Backup         : 0
     Non-DF         : 0
  MAC Flushing mode : STP-TCN
```

```
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Local SHG label   : 28384
Remote SHG labels : 0
Access signal mode: Bundle OOS (Default)
```

Associated Commands

- service-carving
- show evpn ethernet-segment



CHAPTER 10

Configure EVPN IRB, Distributed Anycast Gateway and E-tree

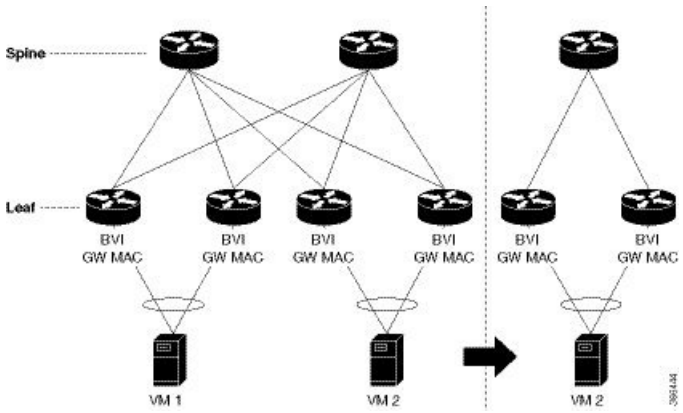
This chapter introduces you to Ethernet VPN (EVPN) Integrated Routing and Bridging (IRB), Distributed Anycast Gateway, and E-Tree features and their description.

- EVPN IRB , on page 359
- EVPN Single-Homing Access Gateway , on page 361
- EVPN Multihoming All-Active, on page 362
- EVPN Single-Active Multihoming for Anycast Gateway IRB, on page 363
- Enable Auto-BGP RT with Manual ESI Configuration, on page 367
- Supported EVPN IRB Scenarios, on page 367
- Distributed Anycast Gateway, on page 367
- BVI-Coupled Mode, on page 370
- VM Mobility Support, on page 372
- Configuring EVPN IRB, on page 374
- Running Configuration for EVPN IRB, on page 375
- Verify EVPN IRB, on page 376
- EVPN IPv6 Hosts with Mobility, on page 386
- Duplicate IP Address Detection, on page 397
- EVPN Automatic Unfreezing of MAC and IP Addresses, on page 399
- EVPN E-Tree, on page 402
- EVPN E-Tree Using RT Constraints, on page 410
- EVPN E-Tree Per-PE (Scenario 1b), on page 425
- DHCPv4 Relay on IRB, on page 431
- DHCPv4 Relay Synchronization for All-Active Multihoming, on page 438
- DHCPv6 Relay IAPD on IRB, on page 439
- DHCPv6 PD Synchronization for All-Active Multihoming using Session Redundancy , on page 442
- IAPD Route Distribution and Withdrawal in DHCPv6 Relay, on page 445

EVPN IRB

EVPN IRB feature enables a Layer 2 VPN and an Layer 3 VPN overlay that allows end hosts across the overlay to communicate with each other within the same subnet and across different subnets within the VPN.

Figure 56: EVPN IRB



The benefit of EVPN IRB is that it allows the hosts in an IP subnet to be provisioned anywhere in the data center. When a virtual machine (VM) in a subnet is provisioned behind a EVPN PE, and another VM is required in the same subnet, it can be provisioned behind another EVPN PE. The VMs do not have to be localized; they need not be directly connected; or be in the same complex. The VM is allowed to move across in the same subnet. Availability of IP MPLS network across all the EVPN PEs enables the provisioning of VM mobility. The EVPN PEs route traffic to each other through MPLS encapsulation.

The EVPN PEs are connected to each other by a spine so they have IP reachability to each other's loopback interfaces. The IP network and MPLS tunnels existing between these EVPN PEs constitute the IP MPLS underlay fabric.

You can configure the MPLS tunnels to tunnel Layer 2 traffic, and to overlay VPN on these tunnels. EVPN control plane distributes both Layer 2 MAC reachability and Layer 3 IP reachability for hosts within the context of the VPN; it overlays a tenant's VPN network on top of the MPLS underlay fabric. Thus you can have tenant's hosts, which are in the same subnet layer 2 domain, but distributed across the fabric, communicate to each other as if they are in a Layer 2 network.

The Layer 2 VLAN and the corresponding IP subnet are not only a network of physically connected hosts on Layer 2 links, but an overlayed network on top of underlayed IP MPLS fabric which is spread across the datacenter.

A routing service, which enables stretching of the subnet across the fabric, is available. It also provides Layer 3 VPN and performs routing between subnets within the context of the Layer 3 VPN. The EVPN PEs provide Layer 2 bridging service between hosts that are spread across the fabric within a Layer 2 domain that is stretched across the fabric, and Layer 3 VPN service or inter-subnet routing service for hosts in different subnets within Layer 3 VPN. For example, as shown in the above topology diagram, the two VM are in the same subnet but they are not connected directly through each other through a Layer 2 link. The Layer 2 link is replaced by MPLS tunnels that are connecting them. The whole fabric acts as a single switch and bridges traffic from one VM to the other. This also enables VM mobility.

Note Egress marking is not supported on L2 interfaces in a bridge domain.

In the above topology diagram, the VMs, VM1 and VM2 are connected each other. When VM2 migrates to a different switch and different server, the VM's current MAC address and IP address are retained. When the subnet is stretched between two EVPN PEs, the same IRB configuration is applied on both the devices.

For stretching within the same subnet, you must configure the AC interface and the EVI; it is not required to configure IRB interface or VRF.

Note Only a single custom MAC address is supported for all BVIs across the system.

Limitations

In case static MAC address is configured on a bundle-ether interface, the following limitations are applied:

- Locally generated packets, such as ICMP, BGP, and so on, going out from the interface have the source MAC address as the statically configured MAC address.
- Transit (forwarded) packets going out of the interface do not have the configured static MAC as source MAC address. In such a scenario, the upper 36-bits come from the system MAC address (or the original/dynamic MAC address) and the lower 12-bits set as zero. To check the dynamic pool of MAC addresses included, use the show ethernet mac-allocation detail command.

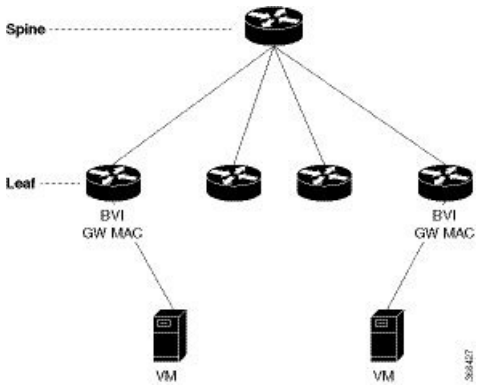
  For example, if the dynamic MAC address is 008A.9624.48D8 and the configured static MAC address is 0011.2222.1111. Then, the source MAC for transit (forwarded) traffic will be 008A.9624.4000.
- To prevent recursive lookup on the egress PE, avoid filtering host routes from EVPN VPNv4 redistribution; instead, inject host routes into the VPNv4 domain.

For more information on limitations, refer *Limitations and Compatible Characteristics of Ethernet Link Bundles* in *Interface and Hardware Component Configuration Guide for Cisco NCS 560 Series Routers*

EVPN Single-Homing Access Gateway

The EVPN provider edge (PE) devices learn the MAC address and IP address from the ARP traffic that they receive from the customer edge (CE) devices. The PEs create the MAC+IP routes. The PEs advertise the MAC+IP routes to MPLS core. They inject the host IP routes to IP-VPN gateway. Subnet routes are also advertised from the access EVPN PEs in addition to host routes. All the PE nodes add the host routes in the IP-VRF table. The EVPN PE nodes add MAC route to the MAC-VRF table. The IP-VPN PE advertise the subnet routes to the provider edge devices which add the subnet routes to IP-VRF table. On the PE devices, IRB gateway IP addresses and MAC addresses are not advertised through BGP. IRB gateway IP addresses or MAC addresses are used to send ARP requests towards the datacenter CEs.

Figure 57: EVPN Single-Homing Access Gateway

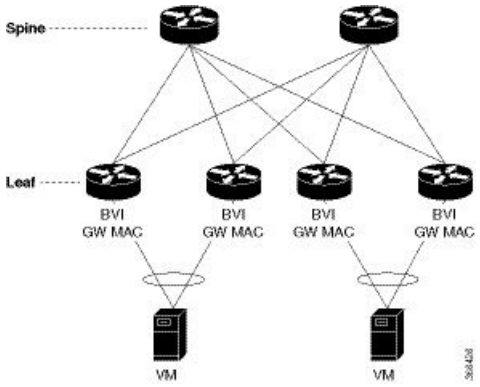


The above topology depicts how EVPN single-homing access gateway enables network connectivity by allowing a CE device to connect to one PE device. The PE device is attached to the Ethernet Segment through bundle or physical interfaces. Null Ethernet Segment Identifier (ESI) is used for single-homing.

EVPN Multihoming All-Active

In EVPN IRB, both EVPN and IP VPN (both VPNv4 and VPNv6) address families are enabled between routers and Data Center Interconnect (DCI) gateways. When Layer 2 (L2) stretch is not available in multiple data centers (DC), routing is established through VPNv4 or VPNv6 routes. When Layer 2 stretch is available, host routing is applied where IP-MAC routes are learnt by ARP and are distributed to EVPN/BGP. In remote peer gateway, these IP-MAC EVPN routes are imported into IP VPN routing table from EVPN route-type 2 routes with secondary label and Layer 3 VRF route-target.

Figure 58: EVPN Multi-Homing All-Active

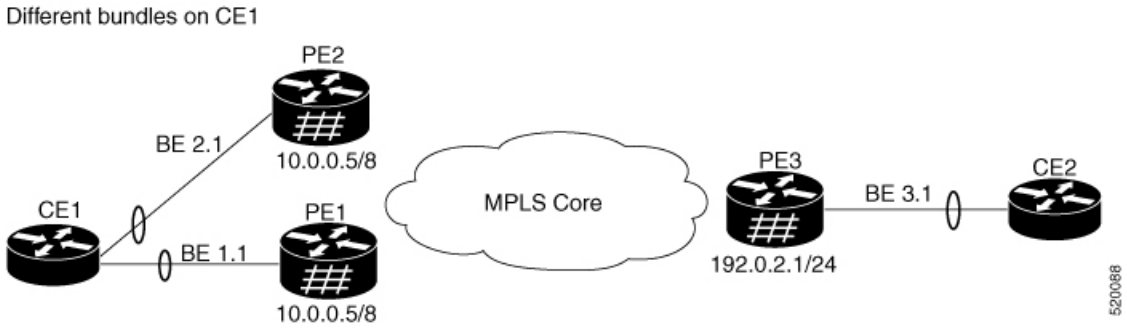


The above topology describes how EVPN Multi-homing access gateway enables redundant network connectivity by allowing a CE device to connect to more than one PE device. Disruptions to the network connectivity are prevented by allowing a CE device to be connected to a PE device or several PE devices through multi-homing. Ethernet segment is the bunch of Ethernet links through which a CE device is connected to more than one PE devices. The All-Active Link Aggregation Group bundle operates as an Ethernet segment. Only MC bundles that operates between two chassis are supported.

EVPN Single-Active Multihoming for Anycast Gateway IRB

The EVPN Single-Active Multihoming for Anycast Gateway IRB feature supports single-active redundancy mode. In this mode, the provider edge (PE) nodes locally connected to an Ethernet Segment load balance traffic to and from the Ethernet Segment based on EVPN service instance (EVI). Within an EVPN service instance, only one PE forwards traffic to and from the Ethernet Segment (ES). This feature supports intersubnet scenario only.

Figure 59: EVPN: Single-Active Multihoming for Anycast Gateway IRB



Consider a topology where CE1 is multihomed to PE1 and PE2. Bundle Ethernet interfaces BE 1.1, BE 2.1, and the ingress interface must belong to the same switching domain on CE1. Enable host routing and configure anycast gateway IP address on both these peering PEs. PE1 and PE2 are connected to PE3 through MPLS core. PE3 has reachability of subnet 10.0.0.5/8 to both peering PEs. Peering PEs has reachability to PE3 subnet 192.0.2.1/24. CE2 is connected to PE3 through an Ethernet interface bundle. PE1 and PE2 advertise Type 4 routes, and then performs designated forwarder (DF) election. The non-DF blocks the traffic in both the directions in single-active mode.

Consider a traffic flow from CE1 to CE2. CE1 sends an address resolution protocol (ARP) broadcast request to both PE1 and PE2. Peering PEs performs designated forwarder (DF) election for shared ESI. If PE1 is the designated forwarder for the EVI, PE1 replies to the ARP request from CE1. PE2 drops the traffic from CE1. Thereafter, all the unicast traffic is sent through PE1. PE2 is set to stand-by or blocked state and traffic is not sent over this path. PE1 advertises MAC to PE3. PE3 always sends and receives traffic through PE1. PE3 sends the traffic to CE2 over Ethernet interface bundle. If BE1 fails, PE2 becomes the DF and traffic flows through PE2.

Configure EVPN Single-Active Multihoming

Perform the following tasks on PE1 and PE2 to configure EVPN Single-Active Multihoming feature:

- Configure EVPN IRB with host routing
- Configure EVPN Ethernet Segment
- Configure Layer 2 Interface
- Configure a Bridge Domain
- Configure VRF

Configure EVPN IRB with Host Routing

Perform this task to configure EVPN IRB with host routing.

Configuration Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 6005
Router(config-l2vpn-bg)# bridge-domain 6005
Router(config-l2vpn-bg-bd)# routed interface BVI50
Router(config-l2vpn-bg-bd-bvi)# exit
Router(config-l2vpn-bg-bd-bvi)# interface Bundle-Ether2.1
Router(config-l2vpn-bg-bd-ac)# evi 6005
Router(config-l2vpnbg-bd-evi)# commit
Router(config-l2vpnbg-bd-evi)# exit
Router(config)# interface BVI50
Router(config-if)# host-routing
Router(config-if)# vrf 30
Router(config-if)# ipv4 address 10.0.0.5 255.0.0.0
Router(config-if)# local-proxy-arp
Router(config-if)# mac-address 1.1.1
Router(config-if)# commit
```

Running Configuration

This section shows EVPN IRB with host routing running configuration.

```
configure
l2vpn
 bridge group 6005
  bridge-domain 6005
   interface Bundle-Ether2.1
    evi 6005
!
   !
 interface BVI34
 host-routing
 vrf 30
 ipv4 address 10.0.0.5 255.0.0.0
 arp learning local
 local-proxy-arp
 mac-address 1.1.1
```

Configure EVPN Ethernet Segment

Perform this task to configure the EVPN Ethernet segment.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# bgp route-target 4000.0000.0001
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
configure
evpn
 interface Bundle-Ether1
   ethernet-segment
    identifier type 0 40.00.00.00.00.00.00.00.01
    load-balancing-mode single-active
    bgp route-target 4000.0000.0001
    !
!
!
```

Configure EVPN Service Instance (EVI) Parameters

Perform this task to define EVPN service instance (EVI) parameters.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 6005
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# rd 200:50
Router(config-evpn-evi-bgp)# route-target import 100:6005
Router(config-evpn-evi-bgp)# route-target export 100:6005
Router(config-evpn-evi-bgp)# commit
```

Running Configuration

```
configure
 evpn
  evi 6005
   bgp
    rd 200:50
    route-target import 100:6005
    route-target export 100:6005
!
!
```

Configure Layer 2 Interface

Perform this task to define Layer 2 interface.

```
Router# configure
Router(config)# interface bundle-ether2.1 l2transport
Router(config-subif-l2)# no shutdown
Router(config-subif-l2)# encapsulation dot1q 1
Router(config-subif-l2)# rewrite ingress tag pop 1 symmetric
Router(config-subif-l2)#commit
Router(config-subif-l2)#exit
```

Running Configuration

This section shows the Layer 2 interface running configuration.

```
configure
 interface bundle-ether2.1 l2transport
  no shutdown
  encapsulation dot1q 1
```

```
  rewrite ingress tag pop 1 symmetric
!
```

Configure a Bridge Domain

Perform the following steps to configure the bridge domain on PE1 and PE2.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 6005
Router(config-l2vpn-bg)# bridge-domain 6005
Router(config-l2vpn-bg-bd)# interface Bundle-Ether2.1
Router(config-l2vpn-bg-bd-ac)# evi 6005
Router(config-l2vpnbg-bd-evi)# commit
Router(config-l2vpnbg-bd-evi)# exit
```

Running Configuration

This section shows the bridge domain running configuration.

```
configure
l2vpn
 bridge group 6005
  bridge-domain 6005
   interface Bundle-Ether2.1
    evi 6005
!
```

Configure VRF

Perform this task to configure VRF.

Configuration Example

```
Router# configure
Router(config)# vrf vrf1
Router(config-vrf)# address-family ipv4 unicast
Router(config-l2vpn-vrf-af)# route-target import 100:6005
Router(config-l2vpn-vrf-af)# route-target export 100:6005
Router(config-l2vpn-vrf-af)# commit
```

Running Configuration

This section shows the VRF running configuration.

```
configure
 vrf vrf1
  address-family ipv4 unicast
   route-target import 100:6005
   route-target export 100:6005

!
```

Enable Auto-BGP RT with Manual ESI Configuration

Configuring an ES-Import RT was previously mandatory for Type 0 ESI. The ES-Import RT is auto-extracted by default, and the configuration serves to override the default value. This feature is based on RFC 7432 but applied specifically to ESI Type 0. For more information, see Section 5 of RFC 7432.

Supported EVPN IRB Scenarios

EVPN IRB supports the following scenarios:

Dual-homing supports the following methods:

- Only all-active mode is supported
- Only two PE gateways in a redundancy group

Single-homing supports the following methods:

- Physical
- VLAN
- Bundle-ethernet
- QinQ access
- Only IPv4 is supported.
- Subnet-stretch feature with EVPN IRB is supported in VRF as well as in Global Routing Table (GRT). In GRT, **bgp implicit-import** under the BGP address-family l2vpn evpn must be configured.

Distributed Anycast Gateway

EVPN IRB for the given subnet is configured on all the EVPN PEs that are hosted on this subnet. To facilitate optimal routing while supporting transparent virtual machine mobility, hosts are configured with a single default gateway address for their local subnet. That single (anycast) gateway address is configured with a single (anycast) MAC address on all EVPN PE nodes locally supporting that subnet. This process is repeated for each locally defined subnet requires Anycast Gateway support.

The host-to-host Layer 3 traffic, similar to Layer 3 VPN PE-PE forwarding, is routed on the source EVPN PE to the destination EVPN PE next-hop over an IP or MPLS tunnel, where it is routed again to the directly connected host. Such forwarding is also known as Symmetric IRB because the Layer 3 flows are routed at both the source and destination EVPN PEs.

The following are the solutions that are part of the Distributed Anycast Gateway feature:

EVPN IRB with All-Active Multi-Homing without Subnet Stretch or Host-Routing across the Fabric

For those subnets that are local to a set of multi-homing EVPN PEs, EVPN IRB Distributed Anycast Gateway is established through subnet routes that are advertised using EVPN Route Type 5 to VRF-hosting remote leafs. Though there is no need for the /32 routes within the subnet to be advertised, host MAC and ARP entries have to synced across the EVPN PE to which the servers are multi-homed.

Note The Subnet Stretch feature with EVPN IRB is exclusively available for use within VRF instances and is not applicable to the global VRF.

This type of multi-homing has the following characteristics:

- All-active EV LAG on access
- Layer 3 ECMP for the fabric for dual-homed hosts based on subnet routes
- Absence of Layer 2 subnet stretch over the fabric
- Layer 2 stretch within redundancy group of leafs with orphan ports

Prefix-routing solution for a non-stretched subnet is summarized as below:

Across multi-homing EVPN PEs:

- Local ARP cache and MAC addresses are synchronized for dual-homed hosts through EVPN MAC+IP host route advertisements. They are imported as local, and are based on the local ESI match, for optimal forwarding to the access gateway.
- Orphan MAC addresses and host IP addresses are installed as remote addresses over the fabric.
- ES/EAD routes are exchanges for the designated forwarder (DF) election and split-horizon label.

Across remote EVPN PEs:

- Dual-homed MAC+IP EVPN Route Type 2 is exchanged with the ESI, EVI Label, Layer 2-Route Type. It is not imported across the fabric, if there is no subnet stretch or host-routing.
- The subnet IP EVPN Route Type 5 is exchanged with VRF label and Layer 3-Route Type.
- Layer 3 Route Type for the VRFs is imported that are present locally.
- Layer 2 Route Type for locally present BDs is imported. It is only imported from the leaf in the same redundancy group, if BD is not stretched.

EVPN IRB with All-Active Multihoming with Subnet Stretch or Host-Routing across the Fabric

For a bridge domain or subnet that is stretched across remote EVPN PEs, both /32 host routes and MAC routes are distributed in a EVPN overlay control plane to enable Layer 2 and Layer 3 traffic to the end points in a stretched subnet.

This type of multihoming has the following characteristics:

- All-active EV-LAG on the access gateway
- Layer 2 or Layer 3 ECMP for the fabric for dual-homed hosts based on Route Type 1 and Route Type 2
- Layer 3 unipath over the fabric for single-homed hosts based on Route Type 2
- Layer 2 subnet stretch over the fabric
- Layer 2 stretch within redundancy group of leafs with orphan ports

MAC and host routing solution for a stretched subnet is summarized as follows:

Across multihoming EVPN PEs:

- The Local ARP cache and MAC addresses are synchronized for dual-homed hosts through EVPN MAC+IP host route advertisements. They are imported as local, based on the local ESI match, for optimal forwarding to the access gateway.
- Synchronized MAC+IP are re-originated for inter-subnet Layer 3 ECMP.
- Orphan MAC address and host IP address are installed as remote addresses over the fabric.
- ES/EAD route is exchanged for designated forwarder (DF) election and split-horizon label.

Across remote EVPN PEs:

- Dual-homed MAC+IP EVPN Route Type 2 is exchange with ESI, EVI label, Layer 2-Route Type, VRF label, and Layer 3-Route Type.
- Subnet IP EVPN Route Type 5 is exchanged for VRF label, Layer 3-Route Type for silent hosts, and non-stretched subnets.
- Layer 3 Route Type is imported for locally present VRFs.
- Layer 2 Route Type is imported for locally present bridge domains.

MAC and IP Unicast Control Plane

This use case has following types:

Prefix Routing or No Subnet Stretch

IP reachability across the fabric is established using subnet prefix routes that are advertised using EVPN Route Type 5 with the VPN label and VRF RTs. Host ARP and MAC sync are established across multi-homing EVPN PEs using MAC+IP Route Type 2 based on a shared ESI to enable local switching through both the multi-homing EVPN PEs.

Host Routing or Stretched Subnet

When a host is discovered through ARP, the MAC and IP Route Type 2 is advertised with both MAC VRF and IP VRF router targets, and with VPN labels for both MAC-VRF and IP-VRF. Particularly, the VRF route targets and Layer 3 VPN label are associated with Route Type 2 to achieve PE-PE IP routing identical to traditional L3VPNs. A remote EVPN PE installs IP/32 entries directly in Layer 3 VRF table through the advertising EVPN PE next-hop with the Layer 3 VPN label encapsulation, much like a Layer 3 VPN imposition PE. This approach avoids the need to install separate adjacency rewrites for each remote host in a stretched

subnet. Instead, it inherits a key Layer 3 VPN scale benefit of being able to share a common forwarding rewrite or load-balance resource across all IP host entries reachable through a set of EVPN PEs.

ARP and MAC sync

For hosts that are connected through LAG to more that one EVPN PE, the local host ARP and MAC entries are learnt in data plane on either or both of the multihoming EVPN PEs. Local ARP and MAC entries are synced across the two multihoming EVPN PEs using MAC and IP Route Type 2 based on a shared ESI to enable local switching through both the multihoming EVPN PEs. Essentially, a MAC and IP Route Type 2 that is received with a local ESI causes the installation of a synced MAC entry that points to the local AC port, and a synced ARP entry that is installed on the local BVI interface.

MAC and IP Route Re-origination

MAC and IP Route Type 2 received with a local ESI, which is used to sync MAC and ARP entries, is also re-originated from the router that installs a SYNC entry, if the host is not locally learnt and advertised based on local learning. This route re-origination is required to establish overlay IP ECMP paths on remote EVPN PEs, and to minimize traffic hit on local AC link failures, that can result in MAC and IP route withdraw in the overlay.

Note If custom or static MAC address is configured on a BVI interface, the MAC address on the wire may be different than what is configured. This has no operational or functional impact.

Intra-subnet Unicast Data Plane

The Layer 2 traffic is bridged on the source EVPN PE using ECMP paths to remote EVPN PEs, established through MAC+IP RT2, for every ES and for every EVI, ES and EAD Route Type 2 routes that are advertised from the local EVPN PEs.

Inter-subnet Unicast Data Plane

Inter-subnet traffic is routed on the source ToRs through overlay ECMP to the destination ToR next-hops. Data packet are encapsulated with the VPN label advertised from the ToR and tunnel label for the BGP next-hop towards the spine. It is then routed again on the destination ToR using a local ARP adjacency towards the host. IP ECMP on the remote ToRs is established through local and re-originated routes advertised from the local ToRs.

BVI-Coupled Mode

When ACs go down, the BVI also goes down. However, with this mode enabled, the state of the BVI remains Up even though the ACs go down. Hence, the BVI state becomes EVPN-aware.

BVI tracks the Up or Down state of ACs and PWs in a bridge. When the EVPN port is available, there may be an L2 redirect path over EVI to carry the traffic between L3 to L2. However, this depends on the remote or peer EVI-EAD routes received.

Under certain conditions, you can reduce the churns of BVI state adjacency by keeping the BVI state Up. BVI state drives the state of EVPN_SYNC adjacencies being pushed to forwarding entries, thereby reducing

the churns further. Keeping the BVI state Up, the router creates adjacencies in the forwarding table, which indicates that a local adjacency is invalid when an interface is down.

Configure BVI-Coupled Mode

Perform this task to configure BVI-coupled mode.

```
evpn
 evi 101
  bgp
   route-target import 60000:101
   route-target export 60000:101
  !
  bvi-coupled-mode

l2vpn
 bridge group BG-1
  bridge-domain BD-1
   interface Bundle-Ether100.101
   !
   routed interface BVI101
   !
   evi 101
```

Verification

Verify that the BVI-coupled mode is enabled.

```
Router# show evpn evi detail

VPN-ID     Encap      Bridge Domain                 Type
---------- ---------- ----------------------------- --------------------
101        MPLS       BD-1                          EVPN
   Stitching: Regular
   Unicast Label  : 35048
   Multicast Label: 33000
   Reroute Label: 0
   Flow Label: N
   Control-Word: Enabled
   E-Tree: Root
   Forward-class: 0
   Advertise MACs: Yes
   Advertise BVI MACs: No
   Aliasing: Enabled
   UUF: Enabled
   Re-origination: Enabled
   Multicast:
     Source connected   : No
     IGMP-Snooping Proxy: No
     MLD-Snooping Proxy : No
   BGP Implicit Import: Enabled
   VRF Name: cust1
   Preferred Nexthop Mode: Off
BVI Coupled Mode: Yes ------------------------> enabled
   BVI Subnet Withheld: ipv4 No, ipv6 No
   RD Config: none
   RD Auto  : (auto) 201.201.201.1:101
   RT Auto  : 60000:101
   Route Targets in Use           Type
   ------------------------------ -------------
   60000:101                      Import
```

```
60000:101                     Export
--------------------- --------------------
```

VM Mobility Support

VM mobility is the ability of virtual machines to migrate between one server and another while retaining their existing MAC and IP addresses.

The following are the two key components in EVPN Route Type 2 that enable VM Mobility:

- Host MAC advertisement component that is imported into local bridge MAC table, and Layer 2 bridged traffic across the network overlay.
- Host IP advertisement component that is imported into the IP routing table in a symmetric IRB design, enables routed traffic across the network overlay.

The above-mentioned components are advertised together in a single MAC + IP host route advertisement. An additional MAC-only route could also be advertised.

The following behaviors of VM are supported. The VM can:

- retain existing MAC and acquire a new IP address
- retain existing IP address and acquire a new MAC
- retain both existing MAC and IP address

MAC and MAC-IP Sequence Numbers

The IRB gateway device assigns, manages, and advertises sequence numbers that are associated with the locally learnt MAC routes through hardware learning, and the locally learnt MAC-IP routes through ARP.

Synchronized MAC and MAC-IP Sequence Numbers

In a host that is multi-homed to two ToRs, the locally learnt MAC and MAC-IP routes are synchronized across the two multi-homing peers through Route Type 2 learnt routes with a local ESI. So a device could have either MAC and MAC-IP, or both of them, learnt through both synchronized and local learning. Sequence numbers are synchronized across local and synchronized routes, because of which the sequence number that is advertised from the two ToRs for a given route is always the same.In certain situations, remote-sync route with same ESI can have a higher sequence number than a local route. In such a case, the local route sequence number is bumped up to match remote-sync route sequence number.

Local Sequence Number Updates

Host mobility is triggered when a local route is learnt while a remote route already exists. When mobility occurs, the local route is assigned a sequence number that is one higher than the existing remote route. This new local route is then advertised to the rest of the network.

Best Route Selection after Host Movement

When a host moves, the EVPN-PE at the new location of the host generates and advertises a higher sequence route to the network. When a higher sequence number route is received, as per RFC 7432, it is considered as the new best route and it is used for forwarding traffic. Best route selection is done for both MAC and MAC-IP routes.

Stale Route Deletion after a Host Movement

After a host moves from local to remote ESI, if a remote route from a different ESI is received and if a local route for the same host with a lower sequence number exists, then the local route is deleted and is withdrawn from the network.

The new higher sequence number remote MAC route is now considered best and is used to forward traffic. An ARP probe is sent to the host at the old local location. Because the host is at new remote location, probe will not succeed, resulting in clearing old local MAC-IP route.

Host Movement Detection through GARP

If a host sends a Gratuitous ARP (GARP) at its new location after a movement, the local MAC and local MAC-IP learning independently trigger mobility for both routes.

Host Move Detection with Silent Host

If a host does not send a GARP or a data packet at its new location following a move, the aging of the local MAC at the old location triggers mobility for both routes.

Host Move Detection without GARP with Data Packet

If the host does not send a GARP following a move, a data packet from the host triggers a proactive ARP probe to discover host MAC-IP and trigger mobility for this host across the overlay.

Duplicate MAC Detection

Duplicate MAC detection and freezing is supported as per RFC 7432.

Detection: Duplicate detection and recovery parameters are configurable. The default configuration is five times in 180 seconds and route freezing after three duplicate cycles. With the default configuration, when a host moves five times in 180 seconds, it is marked as duplicate for 30 seconds. Route advertisement for hosts in Duplicate state is suppressed. Host is taken out of duplicate state after 30 seconds. After a host is detected as duplicate for 3 times, on the fourth duplicate cycle, the host is permanently frozen. All route advertisements are suppressed for the frozen hosts.

In multi-homed hosts, a MAC is not necessarily learnt locally but is learnt through synchronization. Duplicate detection is supported for both local and remote-sync hosts. Remote-sync routes are differentiated from remote routes.

MAC-IP Handling: If the MAC route is in duplicate or frozen state, the corresponding local MAC-IP is updated, except that the route deletes are not withheld.

Duplicate State Handling: When a host is in duplicate state, route advertisements are suppressed. However, local routes are programmed in hardware so that traffic on local EVPN-PE is forwarded to the local host.

Recovery: It is possible to unfreeze permanently frozen hosts. The following is the recommended procedure to clear frozen hosts:

- Shutdown the host which is causing duplicate traffic.
- Use the **clear l2route evpn frozen-mac frozen-flag** command to clear the frozen hosts.

Configuring EVPN IRB

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 3
RP/0/RSP0/CPU0:router(config-if)# lacp system mac 1.1.1
RP/0/RSP0/CPU0:router(config-if)# exit


/* Configure EVPN L3VRF per DC tenant. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# vrf irb1
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 1000:1
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target 1000:1
RP/0/RSP0/CPU0:router(config-vrf-af)# exit


/* Configure Layer 2 attachment circuit (AC) from multichassis (MC) bundle interface, and
bridge-group virtual interface (BVI) per bridge domain. */
/* Note: When a VM migrates from one subnet to another (subnet stretching), apply the
following IRB configuration to both the EVPN PEs. *\

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface bvi 1001
RP/0/RSP0/CPU0:router(config-if)# host-routing
RP/0/RSP0/CPU0:router(config-if)# vrf irb1
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.10.0.4 255.255.255.0
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 172.16.0.1 secondary
RP/0/RSP0/CPU0:router(config-if)# mac-address 00aa.1001.00aa

/* Configure EVPN Layer 2 bridging service. Note: This configuration is performed in Layer
 2 gateway or bridging scenario. */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 1
Router(config-l2vpn-bg)# bridge-domain 1-1
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/1.1
Router(config-l2vpn-bg-bd-ac)# evi 1
Router(config-l2vpn-bg-bd-ac-evi)# commit
Router(config-l2vpnbg-bd-ac-evi)# exit


/* Configure BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 3107
RP/0/RSP0/CPU0:router(config-bgp)# vrf irb1
RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd auto
```

```
RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# redistribute connected
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# redistribute static
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# exit


/* Configure EVPN, and configure main bundle ethernet segment parameters in EVPN. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# bgp
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target import 1000:1
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target export 1000:1
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# exit
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
RP/0/RSP0/CPU0:router(config-evpn-evi)# unknown-unicast-suppression


/* Configure Layer 2 VPN. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group irb
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain irb1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface bundle-Ether3.1001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# routed interface BVI100
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-bvi)# split-horizon group core
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-bvi)# evi 10001
```

Running Configuration for EVPN IRB

```
/* Configure LACP */
interface Bundle-Ether3
 lacp system mac 1.1.1
!



/* Configure EVPN Layer 3 VRF per DC tenant. */
vrf irb1
address-family ipv4 unicast
  import route-target
   1000:1
  !
  export route-target
   1000:1
  !

!
!

/* Configure Layer 2 attachment circuit (AC) from multichassis (MC) bundle interface, and
bridge-group virtual interface (BVI) per bridge domain./*


interface Bundle-Ether3.1001 l2transport
 encapsulation dot1q 1001
 rewrite ingress tag pop 1 symmetric
!
interface BVI1001
```

```
 host-routing
 vrf irb1
 ipv4 address 10.0.1.1 255.255.255.0
 mac-address 0000.3030.1
!

/* Configure BGP. */

router bgp 3107
 vrf irb1
  rd auto
  address-family ipv4 unicast
  redistribute connected
  redistribute static
!
!

/* Configure EVPN. */

evpn
evi 10001
  bgp
   route-target import 1000:1
   route-target export 1000:1
  !
  advertise-mac
  unknown-unicast-suppression
!

/* Configure Layer2 VPN. */

l2vpn
bridge group irb
  bridge-domain irb1
   interface Bundle-Ether3.1001
   !
   routed interface BVI1001
    split-horizon group core
   !
   evi 10001
   !
  !
```

Verify EVPN IRB

Verify the Address Resolution Protocol (ARP) protocol entries, and synced entries in multi-homing scenarios.

```
RP/0/RP0/CPU0:router# show arp vrf evpn1

------------------------------------------------------------------
0/1/CPU0
------------------------------------------------------------------
Address    Age        Hardware Addr   State      Type   Interface

10.1.1.1    -         0010.0001.0001  Interface  ARPA   BVI1
10.1.1.11 02:23:46    1000.0001.0001  Dynamic    ARPA   BVI1
10.1.1.93   -         0000.f65a.357c  EVPN_SYNC  ARPA   BVI1
10.1.2.1    -         0011.0112.0001  Interface  ARPA   BVI2
10.1.2.91 02:24:14    0000.f65a.3570  Dynamic    ARPA   BVI2
10.1.2.93 02:21:52    0000.f65a.357d  Dynamic    ARPA   BVI2
------------------------------------------------------------------
```

```
0/0/CPU0
------------------------------------------------------------------
Address   Age         Hardware Addr   State      Type  Interface

10.1.1.1  -           0010.0001.0001  Interface  ARPA  BVI1
10.1.1.11 02:23:46    1000.0001.0001  Dynamic    ARPA  BVI1
10.1.1.93 -           0000.f65a.357c  EVPN_SYNC  ARPA  BVI1
10.1.2.1  -           0011.0112.0001  Interface  ARPA BVI2
10.1.2.91 02:24:14    0000.f65a.3570  Dynamic    ARPA BVI2
10.1.2.93 02:21:52    0000.f65a.357d  Dynamic    ARPA BVI2
```

Verify the adjacency entries, particularly verify newly added information for synced IPv4 and IP ARP entries.

```
RP/0/RP0/CPU0:router# show adjacency ipv4 BVI 1 internal detail location 0/0/CPU0

BVI1, 10.1.1.93 (ipv4)
Version: 1169, references: 2, transient lock: 0
Encapsulation information (14 bytes) 0000f65a357c0000f65a357c0800 MTU: 1500
 Adjacency pointer is: 0x770a9278
 Platform adjacency pointer is: 0x7d7bc380
 Last updated: Feb 28 15:58:21.998
 Adjacency producer: arp (prod_id: 10)
 Flags: incomplete adj,
 Additional Adjacency Information (4 bytes long),
 Upto first 4 bytes (in hex): 01000000
 Netio idb pointer not cached Cached interface type: 78

Adjacency references:
bfd_agent (JID 150, PID 3637), 0 reference
l2fib_mgr (JID 185, PID 4003), 0 reference
fib_mgr (JID 294, PID 3605), 1 reference
aib (JID 314, PID 3590), 1 reference

BVI1, 10.1.1.11 (ipv4) Version: 1493,
references: 3, transient lock: 0
Encapsulation information (14 bytes) 10000001000100100000100010800
MTU: 1500
Adjacency pointer is: 0x770ab778
Platform adjacency pointer is: 0x7d7bcb10
Last updated: Mar 2 17:22:00.544
Adjacency producer: arp (prod_id: 10)
Flags: incomplete adj,
Netio idb pointer not cached Cached interface type: 78
Adjacency references:
bfd_agent (JID 150, PID 3637), 0 reference
l2fib_mgr (JID 185, PID 4003), 1 reference
fib_mgr (JID 294, PID 3605), 1 reference
aib (JID 314, PID 3590), 1 reference
```

Verify the entries to obtain details learnt in L2FIB line cards. In multi-homing active-active scenario, the link-local addresses are also updated and distributed to EVPN peer gateways.

```
RP/0/RP0/CPU0:router# show l2vpn mac-learning mac-ipv4 all location 0/RP0/CPU0

Topo ID  Producer  Next Hop(s)  Mac Address     IP Address

6        0/0/CPU0   BV1        1000.0001.0001      10.1.1.11
7        0/0/CPU0   BV2        0000.f65a.3570      10.1.2.91
7       0/0/CPU0    BV2        0000.f65a.357d      10.1.2.93
```

```
RP/0/RP0/CPU0:router# show l2vpn mac-learning mac-ipv6 all location 0/RP0/CPU0

Topo ID  Producer  Next Hop(s)  Mac Address    IP Address

6      0/0/CPU0    BV1       0000.f65a.357c    fe80::200:f6ff:fe5a:357c
7      0/0/CPU0    BV2       0000.f65a.3570    10:1:2::91
7      0/0/CPU0    BV2       0000.f65a.357d    10:1:2::93
7      0/0/CPU0    BV2       0000.f65a.3570    fe80::200:f6ff:fe5a:3570
```

Verify sequence ID for VM mobility.

```
RP/0/RP0/CPU0:router# show l2route evpn mac-ip all detail

Sun Apr 30 18:09:19.368 PDT
Flags: (Stt)=Static; (L)=Local; (R)=Remote; (F)=Flood;
(N)=No Redistribution; (Rtr)=Router MAC; (B)=Best Route;
(P)=Probe; (S)=Peer Sync; (F)=Flush;
(D)=Duplicate MAC; (Z)=Frozen MAC;

Topo ID    Mac Address     IP Address  Prod   Next Hop(s)         Seq No  Flags
Opaque Data Type    Opaque Data Len   Opaque Data Value
-------    ------------     ----------  ----   ----------          ------  -----
----------------    ---------------   -----------------
33         0022.6730.0001  10.130.0.2  L2VPN  Bundle-Ether6.1300  0       SB 0 12
0x06000000           0x22000080        0x00000000

Last Update: Sun Apr 30 15:00:01.911 PDT

33         0022.6730.0002 10.130.0.3  LOCAL  Bundle-Ether6.1300   0       B           N/A
                 N/A              N/A


RP/0/RP0/CPU0:router# show l2route evpn mac all detail


Flags:  (Stt)=Static; (L)=Local; (R)=Remote; (F)=Flood;
        (N)=No Redistribution; (Rtr)=Router MAC; (B)=Best Route;
        (S)=Peer Sync; (Spl)=Split; (Rcv)=Recd;
        (D)=Duplicate MAC; (Z)=Frozen MAC;

Topo ID  Mac Address     Prod   Next Hop(s)          Seq No  Flags  Slot ESI   Opaque Data
Type  Opaque Data Len Opaque Data Value
-------- --------------  ------ -----------        ------  ------ ---- ----  -----------------
  ---------------- -----------------
36       0022.5830.0001  L2VPN  Bundle-Ether5.1300  0       BSSpl  0    (F)    0
      12             0x06000000 0x25000080 0x00000000

Last Update: Thu Apr 20 09:04:44.358 PDT
```

Configuration Example

```
/* Mac Address Duplicate Detection Configuration */
Router# configure
Router(config)# evpn
Router(config-evpn)# host mac-address duplicate-detection
Router(config-evpn-host-mac-addr-dup-detection)# move-count 2
Router(config-evpn-host-mac-addr-dup-detection)# freeze-time 10
```

```
Router(config-evpn-host-mac-addr-dup-detection)# retry-count 2
Router(config-evpn-host-mac-addr-dup-detection)# commit
```

Running Configuration

Verify the entries to obtain details learnt in L2FIB RP when it is an aggregator. Route processor (RP) entries are aggregated entries obtained from the line cards. In some cases of MAC move, there could be different states for the same MAC. This is displayed in RP aggregated entries. RP determines the update to be sent to L2RIB according to MAC-Learning algorithms.

```
RP/0/RP0/CPU0:router#  show l2vpn mac-learning mac-ipv4 all location 0/RP0/CPU0

Topo ID  Producer        Next Hop(s)        Mac Address         IP Address
-------      --------    -----------        --------------      ----------
6            0/0/CPU0       BV1            1000.0001.0001      10.1.1.11
7            0/0/CPU0       BV2            0000.f65a.3570      10.1.2.91
7            0/0/CPU0       BV2            0000.f65a.357d      10.1.2.93
```

Verify the entries in L2RIB that are updated by RP L2FIB. Note the following when you verify the entries:

- The entries with producer as L2VPN and NH as remote IP are learnt from the remote peer gateways, which are learnt from BGP, updated to EVPN, and then updated to L2RIB. So these entries are not from local IP-MAC learning.
- The entries with producer as L2VPN and NH as local bundle interfaces are synced entries from MH-AA peer gateway.
- The entries with producer as LOCAL and NH as local bundle interfaces are dynamically learnt local entries.

```
RP/0/RP0/CPU0:router# show l2route evpn mac-ip evi 6

Topo ID      Mac Address        IP Address               Prod        Next Hop(s)
--------     --------------     ---------------          ------      -------------
6            0000.f65a.3569     10.1.1.101               L2VPN     172.16.0.2/24014/ME
6            0000.f65a.3575     10.1.1.97                L2VPN     172.16.0.7/24025/ME
6            0000.f65a.3575     10:1:1::97               L2VPN     172.16.0.7/24025/ME
6            0000.f65a.3575     fe80::200:f6ff:fe5a:3575 L2VPN     172.16.0.7/24025/ME
6            0000.f65a.357c     10.1.1.93                L2VPN      Bundle-Ether1.11
6            0000.f65a.357c     10:1:1::93               L2VPN      Bundle-Ether1.11
6            0000.f65a.357c     fe80::200:f6ff:fe5a:357c LOCAL      Bundle-Ether1.11
6            0010.0001.0012     10.1.1.12                L2VPN      172.16.0.7/24025/ME
6            1000.0001.0001     10.1.1.11                LOCAL      Bundle-Ether1.11
6            90e2.ba8e.c0c9     10.1.1.102               L2VPN      172.16.0.2/24014/ME
```

Verify entries to obtain details of EVPN.

```
RP/0/RP0/CPU0:router# show evpn evi vpn-id 1 mac ipv4 10.1.1.93 detail

EVI       MAC address         IP address           Nexthop             Label
----      ---------------     ----------           ----------          -----
1          0000.f65a.357c      10.1.1.93           172.16.0.2          24014

Ethernet Tag : 0
Multi-paths Resolved : True
```

```
Static : No
Local Ethernet Segment : N/A
Remote Ethernet Segment : 0100.6cbc.a77c.c180.0000
Local Sequence Number : N/A
Remote Sequence Number : 0
Local Encapsulation : N/A
Remote Encapsulation : MPLS
```

Verify local BGP entries with appropriate second label and second IP VRF route-target.

```
RP/0/RP0/CPU0:router# show bgp l2vpn evpn rd 172.16.0.1:1
[2][0][48][0000.f65a.357c][32][10.1.1.93]/136

BGP routing table entry for [2][0][48][0000.f65a.357c][32][10.1.1.93]/136, Route
Distinguisher: 172.16.0.1:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 3772 3772
Local Label: 24013
Last Modified: Feb 28 16:06:37.073 for 2d19h
Paths: (2 available, best #1)
Advertised to peers (in unique update groups):
172.16.0.9
Path #1: Received by speaker 0
Advertised to peers (in unique update groups):
172.16.0.9
Local
0.0.0.0 from 0.0.0.0 (172.16.0.1)
Second Label 24027                                  >>>>  Second label when IRB host-routing
 is enabled.
Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
rib-install
Received Path ID 0, Local Path ID 0, version 3772
Extended community: SoO:172.16.0.2:1 RT:100:100
EVPN ESI: 0100.6cbc.a77c.c180.0000
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 (metric 101) from 172.16.0.9 (172.16.0.2)
Received Label 24014, Second Label 24031
Origin IGP, localpref 100, valid, internal, add-path, import-candidate, imported, rib-install
Received Path ID 0, Local Path ID 2, version 3769
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100  >>>  Second RT is IP VRF RT for
remote to import into IP VRF routing table.
Originator: 172.16.0.2, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

```
RP/0/RP0/CPU0:router# show bgp l2vpn evpn rd 172.16.0.1:1
[2][0][48][0000.f65a.357c][128][10:1:1::93]/232

[2][0][48][0000.f65a.357c][128][10:1:1::93]/232
BGP routing table entry for [2][0][48][0000.f65a.357c][128][10:1:1::93]/232, Route
Distinguisher: 172.16.0.1:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 3172 3172
Local Label: 24013
```

```
Last Modified: Feb 28 11:34:33.073 for 3d00h
Paths: (2 available, best #1)
Advertised to peers (in unique update groups):
172.16.0.9
Path #1: Received by speaker 0
Advertised to peers (in unique update groups):
172.16.0.9
Local
0.0.0.0 from 0.0.0.0 (172.16.0.1)
Second Label 24029
Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
rib-install
Received Path ID 0, Local Path ID 0, version 3172
Extended community: SoO:172.16.0.2:1 RT:100:100
EVPN ESI: 0100.6cbc.a77c.c180.0000
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 (metric 101) from 172.16.0.9 (172.16.0.2)
Received Label 24014, Second Label 24033
Origin IGP, localpref 100, valid, internal, add-path, import-candidate, imported, rib-install
Received Path ID 0, Local Path ID 2, version 3167
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

Verify the remote peer gateway BGP entries with correct label and route-target. Particularly verify the local auto-generated RD on a remote EVPN gateway. EVPN type-2 routes are imported into EVPN. The host routes of IPv4 /32 addresses are imported only into IP VRF route-table in the remote EVPN gateway, but not in the local EVPN gateway where local BVI adjacency is used to overwrite RIB entries.

```
RP/0/RP0/CPU0:router#  show bgp l2vpn evpn rd 172.16.0.7:1
[2][0][48][0000.f65a.357c][32][10.1.1.93]/136
BGP routing table entry for [2][0][48][0000.f65a.357c][32][10.1.1.93]/136, Route
Distinguisher: 172.16.0.7:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 16712 16712
Last Modified: Feb 28 16:06:36.448 for 2d19h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24013, Second Label 24027 >>>> First label for L2 MAC unicast bridging;
second label for EVPN IRB host-routing
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported,
rib-install
Received Path ID 0, Local Path ID 0, version 16712
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
```

```
Received Label 24014, Second Label 24031
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported,
rib-install
Received Path ID 0, Local Path ID 1, version 16706
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

```
RP/0/RP0/CPU0:router# show bgp l2vpn evpn rd 172.16.0.7:1
[2][0][48][0000.f65a.357c][128][10:1:1::93]/232

BGP routing table entry for [2][0][48][0000.f65a.357c][128][10:1:1::93]/232, Route
Distinguisher: 172.16.0.7:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 6059 6059
Last Modified: Feb 28 12:03:22.448 for 2d23h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24013, Second Label 24029
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported,
rib-install
Received Path ID 0, Local Path ID 0, version 6043
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24014, Second Label 24033
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported,
rib-install
Received Path ID 0, Local Path ID 1, version 6059
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

Verify the remote peer gateway with host routes of IPv4 /32 addresses imported into the IP VRF routing table.

```
RP/0/RP0/CPU0:router#  show bgp vpnv4 unicast vrf evpn1 10.1.1.93/32

BGP routing table entry for 10.1.1.93/32, Route Distinguisher: 172.16.0.7:11
Versions:
Process bRIB/RIB SendTblVer
Speaker 22202 22202
Last Modified: Feb 28 16:06:36.447 for 2d19h
Paths: (2 available, best #1)
Not advertised to any peer
```

```
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24027
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported
Received Path ID 0, Local Path ID 0, version 22202
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1
>>>> The source from L2VPN and from synced ARP entry.
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24031
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported
Received Path ID 0, Local Path ID 1, version 22201
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 17.0.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
>>>> source from L2VPN and from dynamic ARP entry
```

```
RP/0/RP0/CPU0:router# show bgp vpnv6 unicast vrf evpn1 10:1:1::93/128

BGP routing table entry for 10:1:1::93/128, Route Distinguisher: 172.16.0.7:11
Versions:
Process bRIB/RIB SendTblVer
Speaker 22163 22163
Last Modified: Feb 28 12:09:30.447 for 2d23h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24029
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported
Received Path ID 0, Local Path ID 0, version 22163
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1 >>>
Source from L2VPN and from synced ARP entry.
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24033
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported
Received Path ID 0, Local Path ID 1, version 22163
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1 >>>
Source from L2VPN and from dynamic ARP entry.
```

```
RP/0/RP0/CPU0:router# show bgp vpnv6 unicast vrf evpn1 10:1:1::93/128

BGP routing table entry for 10:1:1::93/128, Route Distinguisher: 172.16.0.7:11
Versions:
Process bRIB/RIB SendTblVer
Speaker 22163 22163
Last Modified: Feb 28 12:09:30.447 for 2d23h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24029
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported
Received Path ID 0, Local Path ID 0, version 22163
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24033
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported
Received Path ID 0, Local Path ID 1, version 22163
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

Verify local forwarding with local adjacency which overwrite the RIB entries, and remote peer that use the IP VRF host route entries for IP VPN forwarding.

```
RP/0/RP0/CPU0:router#  show bgp vpnv4 unicast vrf evpn1 10.1.1.93/32

-- For local routing and forwarding
RP/0/RP0/CPU0:PE11-R1#show route vrf evpn1 10.1.1.93
Routing entry for 10.1.1.93/32
Known via "bgp 3107", distance 200, metric 0, type internal
Installed Feb 28 15:57:28.154 for 2d20h
Routing Descriptor Blocks
172.16.0.2, from 172.16.0.9      >>>  From MH-AA peer.
Nexthop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
Route metric is 0
No advertising protos.

RP/0/RP0/CPU0:PE11-R1# show cef vrf evpn1 10.1.1.93 location 0/0/CPU0
10.1.1.93/32, version 0, internal 0x1120001 0x0 (ptr 0x7b40052c) [1], 0x0 (0x7b286010), 0x0
 (0x0)
Updated Feb 28 15:58:22.688
local adjacency 10.1.1.93
Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
via 10.1.1.93/32, BVI1, 2 dependencies, weight 0, class 0 [flags 0x0]
path-idx 0 NHID 0x0 [0x7f531f88 0x0]
next hop
```

```
local adjacency              >>> Forwarding with local synced ARP adjacency entries.


For remote routing and forwarding:

RP/0/RP0/CPU0:router# show route vrf evpn1 10.1.1.93

Routing entry for 10.1.1.93/32
Known via "bgp 3107", distance 200, metric 0
Number of pic paths 1 , type internal
Installed Feb 28 16:06:36.431 for 2d20h
Routing Descriptor Blocks
172.16.0.1, from 172.16.0.9
Nexthop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
Route metric is 0
172.16.0.2, from 172.16.0.9, BGP backup path
Nexthop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
Route metric is 0
No advertising protos.

RP/0/RP0/CPU0:router# show cef vrf evpn1 10.1.1.93 location 0/0/CPU0

10.1.1.93/32, version 86, internal 0x5000001 0x0 (ptr 0x99fac884) [1], 0x0 (0x0), 0x208
(0x96c58494)
Updated Feb 28 16:06:39.285
Prefix Len 32, traffic index 0, precedence n/a, priority 3
via 172.16.0.1/32, 15 dependencies, recursive [flags 0x6000]
path-idx 0 NHID 0x0 [0x97955380 0x0]
recursion-via-/32
next hop VRF - 'default', table - 0xe0000000
next hop 172.16.0.1/32 via 34034/0/21
next hop 100.0.57.5/32 Te0/0/0/3 labels imposed {ImplNull 24011 24027}
next hop 100.0.67.6/32 Te0/0/0/1 labels imposed {ImplNull 24009 24027}
via 172.16.0.2/32, 11 dependencies, recursive, backup [flags 0x6100]
path-idx 1 NHID 0x0 [0x979554a0 0x0]
recursion-via-/32
next hop VRF - 'default', table - 0xe0000000
next hop 172.16.0.2/32 via 34035/0/21
next hop 100.0.57.5/32 Te0/0/0/3 labels imposed {ImplNull 24012 24031}
next hop 100.0.67.6/32 Te0/0/0/1 labels imposed {ImplNull 24010 24031}
```

The following sections describe how to verify the subnet stretching.

Verify the VRF.

```
RP/0/RP0/CPU0:leafW# show run vrf cust130

vrf cust130
address-family ipv4 unicast
  import route-target
   130:130
  !
  export route-target
   130:130
  !
!
!
```

Verify the BGP configuration.

```
 RP/0/RP0/CPU0:leafW# show run router bgp | begin vrf cust130

vrf cust130
  rd auto
  address-family ipv4 unicast
   label mode per-vrf
   maximum-paths ibgp 10
   redistribute connected
  !
!
```

Verify the L2VPN.

```
RP/0/RP0/CPU0:leafW# show run l2vpn bridge group bg130

l2vpn
bridge group bg130
  bridge-domain bd130
   interface Bundle-Ether1.1300
   !
   interface Bundle-Ether5.1300
   !
   routed interface BVI130
   evi 130
   !
  !
!
!
```

EVPN IPv6 Hosts with Mobility

EVPN IPv6 Hosts with Mobility feature enables you to provide EVPN IPv6 service over IPv4-MPLS core network. This feature supports all-active multihoming and virtual machine (VM) or host move.

Service Providers (SPs) use a stable and established core with IPv4-MPLS backbone for providing IPv4 VPN services. The IPv6 VPN Provider Edge Transport over MPLS (IPv6 on Provider Edge Routers [6PE] and IPv6 on VPN Provider Edge Routers [6VPE]) facilitates SPs to offer IPv6 VPN services over IPv4 backbone without an IPv6 core. The provide edge (PE) routers run MP-iBGP to advertise IPv6 reachability and IPv6 label distribution. For 6PE, the labels are allocated per IPv6 prefix learnt from connected customer edge (CE) routers and for 6VPE, the PE router can be configured to allocate labels on a per-prefix or per-CE and per-VRF level.

Mobility Support

In global VRF, mobility is not supported. However, you can move a host from one ES to another ES within the same bridge domain. The host gets a new MAC address and IP address. The host can have multiple IP addresses for the same MAC address.

In non-default VRF, mobility is supported with the following conditions:

- Basic MAC move: The IP address and MAC address remains the same. You can move a host from one ES to another ES with the same IP address and MAC address

- Same MAC address but with a different IP address: The host gets a new IP address
- Same IP address but with a different MAC address: The host gets a new MAC address but retains the same IP address
- Multiple IP addresses with the same MAC address: Many VMs are involved in the same the MAC move

Restrictions

- In customer VRFs, when host routing is not configured, MAC-IP advertisement is different between zero ESI and none-zero ESI. When host routing is not configured, MAC-IP with non-zero ESI is advertised without L3 RT (VRF RT). MAC-IP with zero ESI is not advertised. The following table lists the behavior of MAC-IP advertisement with respect to ESI and host routing.

ESI Type	With host routing	Without host routing
MAC-IP with non-zero ESI	Advertised with L3 VRF RT	Advertised without L3 VRF RT
MAC-IP with zero ESI	Advertised with L3 VRF RT	Not advertised

- In global VRF, Layer 2 stretch is not supported.
- MAC move in global VRF is only supported if the host is within the same bridge domain. You can move a host from one ES to another ES within the same bridge domain.
- Duplication of IP address detection is not supported.
- Maximum number of leafs allowed per ESI is two.

Configure EVPN IPv6 Hosts with Mobility

Perform the following tasks to configure EVPN IPv6 Hosts with Mobility feature:

- Configure VRF
- Configure ISIS
- Configure BGP
- Configure AC interface
- Configure BVI interface
- Configure EVPN
- Configure L2VPN

Note A device can contain up to 128K MAC address entries. A bridge domain on a device can contain up to 65K MAC address entries.

Note

- You cannot configure the EVPN remote peer using the VPNv4 unicast if you have configured the **advertise vpnv4 unicast re-originated** command under the L2VPN EVPN address-family. You can either configure the VPNv4 unicast or the advertise vpnv4 unicast re-originated under L2VPN EVPN address-family.
- You cannot configure the EVPN remote peer using the VPNv6 unicast if you have configured the **advertise vpnv6 unicast re-originated** command under the L2VPN EVPN address-family. You can either configure the VPNv6 unicast or the advertise vpnv6 unicast re-originated under L2VPN EVPN address-family.

```
/* Configure VRF */

Router# configure
Router(config)# vrf cust102
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import route-target 160102:16102
Router(config-vrf-af)# export route-target 160102:16102
Router(config-vrf-af)# exit
!
Router(config-vrf)# address-family ipv6 unicast
Router(config-vrf-af)# import route-target 6160102:16102
Router(config-vrf-af)# export route-target 6160102:16102
Router(config-vrf-af)# commit
!

/* Configure ISIS */

Router# configure
Route(config)# router isis v6
Route(config-isis)# 49.0001.0000.0160.0005.00
Route(config-isis)# nsr
Route(config-isis)# log adjacency changes
Route(config-isis)# lsp-gen-interval maximum-wait 5000 initial-wait 1 secondary-wait
20
Route(config-isis)# lsp-mtu 1468
Route(config-isis)# lsp-refresh-interval 65000
Route(config-isis)# max-lsp-lifetime 65535
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide
Route(config-isis-af)# microloop avoidance protected
Route(config-isis-af)# spf-interval maximum-wait 5000 initial-wait 1 secondary-wait 20
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface Bundle-Ether10
Route(config-isis-if)# point-to-point
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# fast-reroute per-prefix
Route(config-isis-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-af)# metric 10
Route(config-isis-af)# exit
!
Route(config-isis)# interface Bundle-Ether20
```

```
Route(config-isis-if)# point-to-point
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# fast-reroute per-prefix
Route(config-isis-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-af)# metric 10
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# passive
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback10
Route(config-isis-if)# passive
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 1605
Route(config-isis-af)# commit
Route(config-isis-af)# exit
!

/* Configure Segment Routing */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# commit

/* Configure BGP */

Router(config)# router bgp 100
Router(config-bgp)# bfd minimum-interval 50
Router(config-bgp)# bfd multiplier 3
Router(config-bgp)# bgp router-id 160.0.0.5
Router(config-bgp)# address-family ipv4 unicast      --->  To support V4 Global VRF
Router(config-bgp-af)# maximum-paths ibgp 10 unequal-cost  ---> ECMP
Router(config-bgp-af)# redistribute connected    --> V4 Global VRF
Router(config-bgp-af)# exit
!
Router(config-bgp)# address-family ipv4 unicast      --->  VRF
Router(config-bgp-af)# vrf all
Router(config-bgp-af)# label mode per-vrf
Router(config-bgp-af)# exit
!
Router(config-bgp)# address-family ipv6 unicast   ---> For 6PE
Router(config-bgp-af)# label mode per-vrf
Router(config-bgp-af)# maximum-paths ibgp 8
Router(config-bgp-af)# redistribute static
Router(config-bgp-af)# allocate-label all
Router(config-bgp-af)# exit
!
Router(config-bgp)# address-family vpnv6 unicast   ---> 6 VPE
Router(config-bgp-af)# vrf all
Router(config-bgp-af)# label mode per-vrf
Router(config-bgp-af)# exit
!
Router(config-bgp)# address-family l2vpn evpn   ----> EVPN
Router(config-bgp-af)# bgp implicit-import      ----> Global VRF
Router(config-bgp-af)# exit
!
Router(config-bgp)# neighbor-group evpn-rr
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# update-source loopback0
```

```
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy nh-lo10 out
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp-nbr)# address-family ipv6 labeled-unicast  ----> For 6PE
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy nh-lo10 out
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated -> For Route Type 5
Router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated -> For Route Type 5
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp)# neighbor 160.0.0.1
Router(config-bgp-nbr)# use neighbor-group evpn-rr
Router(config-bgp-nbr)# exit
!
Router(config-bgp)# neighbor 160.0.0.2
Router(config-bgp-nbr)# use neighbor-group evpn-rr
Router(config-bgp-nbr)# exit
!
Router(config-bgp)# vrf all
Router(config-bgp-vrf)# rd 1605:102
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# maximum-paths ibgp 10 unequal-cost
Router(config-bgp-vrf-af)# redistribute connected   --->  Triggers Route Type 5
Router(config-bgp-vrf-af)# exit
!
Router(config-bgp-vrf)# address-family ipv6 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# maximum-paths ibgp 10 unequal-cost
Router(config-bgp-vrf-af)# redistribute connected
Router(config-bgp-vrf-af)# exit
!

/* Configure AC interface */

Router(config)# interface Bundle-Ether1.102 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 102
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

/* Configure BVI interface */

Router(config)# interface BVI100
Router(config-if)# ipv4 address 56.78.100.1 255.255.255.0
Router(config-if)# ipv6 address 56:78:100::1/64
Router(config-if)# mac-address 22.22.22
Router(config-if)# exit
!
Router(config)# interface BVI102
Router(config-if)# host-routing
Router(config-if)# vrf cust102
Router(config-if-vrf)# ipv4 address 56.78.102.1 255.255.255.0
Router(config-if-vrf)# ipv6 address 56:78:100::1/64
Router(config-if-vrf)# ipv6 address 56:78:102::1/64
Router(config-if-vrf)# mac-address 22.22.22
Router(config-if)# commit
```

```
/* Configure CEF */ [Required for dual homing]
Router# configure
Router(config)# cef adjacency route override rib

/* Configure EVPN, and configure main bundle ethernet segment parameters in EVPN */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 102
Router(config-evpn-evi)# bgp
Router(config-evpn-evi)# rd 1605:102
Router(config-evpn-evi-bgp)# route-target import 160102:102
Router(config-evpn-evi-bgp)# route-target export 160102:102
Router(config-evpn-evi-bgp)# exit
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# exit
!
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 56.56.56.56.56.56.56.56.01
Router(config-evpn-ac-es)# exit
!
Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 56.56.56.56.56.56.56.56.02
Router(config-evpn-ac-es)# commit

/* Configure L2VPN */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg102
Router(config-l2vpn-bg)# bridge-domain bd102
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1.102
Router(config-l2vpn-bg-bd-ac)# exit
!
Router(config-l2vpn-bg-bd)# interface Bundle-Ether2.102
Router(config-l2vpn-bg-bd-ac)# exit
!
Router(config-l2vpn-bg-bd)# interface Bundle-Ether3.102
Router(config-l2vpn-bg-bd-ac)# exit
!
Router(config-l2vpn-bg-bd)# interface Bundle-Ether4.102
Router(config-l2vpn-bg-bd-ac)# exit
!
Router(config-l2vpn-bg-bd)# interface Bundle-Ether5.102
Router(config-l2vpn-bg-bd-ac)# routed interface BVI102
Router(config-l2vpn-bg-bd-bvi)# evi 102
Router(config-l2vpn-bg-bd-bvi-evi)# commit
```

Running Configuration

```
/* Configure VRF */

vrf cust102
 address-family ipv4 unicast
 import route-target
 160102:16102
 !
 export route-target
 160102:16102
 !
```

```
 !
 address-family ipv6 unicast
 import route-target
 6160102:16102
 !
 export route-target
 6160102:16102
 !
 !
!

/ * Configure ISIS */

router isis v6
 net 49.0001.0000.0160.0005.00
 nsr
 log adjacency changes
 lsp-gen-interval maximum-wait 5000 initial-wait 1 secondary-wait 20
 lsp-mtu 1468
 lsp-refresh-interval 65000
 max-lsp-lifetime 65535
 address-family ipv4 unicast
 metric-style wide
 microloop avoidance protected
 spf-interval maximum-wait 5000 initial-wait 1 secondary-wait 20
 segment-routing mpls sr-prefer
 segment-routing prefix-sid-map advertise-local
 !
 interface Bundle-Ether10
 point-to-point
 address-family ipv4 unicast
 fast-reroute per-prefix
 fast-reroute per-prefix ti-lfa
 metric 10
 !
 !
 interface Bundle-Ether20
 point-to-point
 address-family ipv4 unicast
 fast-reroute per-prefix
 fast-reroute per-prefix ti-lfa
 metric 10
 !
 !
 interface Loopback0
 passive
 address-family ipv4 unicast
 !
 !
 interface Loopback10
 passive
 address-family ipv4 unicast
 prefix-sid index 1605
 !
 !
!

/ * Configure Segment Routing */

segment-routing
 global-block 16000 23999
!

/ * Configure BGP */
```

```
router bgp 100
 bfd minimum-interval 50
 bfd multiplier 3
 bgp router-id 160.0.0.5
 address-family ipv4 unicast      --->  To support V4 Global VRF
  maximum-paths ibgp 10 unequal-cost  ---> ECMP
  redistribute connected    --> V4 Global VRF
 !
 address-family vpnv4 unicast ---> VRF
  vrf all
   label mode per-vrf
 !
 address-family ipv6 unicast   ---> For 6PE
  label mode per-vrf
  maximum-paths ibgp 8
  redistribute connected
  redistribute static
  allocate-label all
 !
 address-family vpnv6 unicast   ---> 6VPE
  vrf all
   label mode per-vrf
 !
 address-family l2vpn evpn   ----> EVPN
 bgp implicit-import         ----> Global VRF
 !

neighbor-group evpn-rr
 remote-as 100
 bfd fast-detect
 update-source Loopback0
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy nh-lo10 out
 !
 address-family ipv6 labeled-unicast  ----> For 6PE
 route-policy pass-all out
 !
 address-family l2vpn evpn
 route-policy pass-all in
 route-policy nh-lo10 out
 advertise vpnv4 unicast re-originated   ---> For Route Type 5
 advertise vpnv6 unicast re-originated   ----> For Route Type 5
 !
 !
 neighbor 160.0.0.1
 use neighbor-group evpn-rr
 !
 neighbor 160.0.0.2
 use neighbor-group evpn-rr
 !
 vrf cust102
 rd 1605:102
 address-family ipv4 unicast
 label mode per-vrf
 maximum-paths ibgp 10 unequal-cost
 redistribute connected   <----- Triggers Route Type 5
 !
 address-family ipv6 unicast
 label mode per-vrf
 maximum-paths ibgp 10 unequal-cost
 redistribute connected
 !
```

```
 !

/* Configure AC interface */

interface Bundle-Ether1.102 l2transport
 encapsulation dot1q 102
 rewrite ingress tag pop 1 symmetric
!
/* Configure BVI interface */
interface BVI100
 ipv4 address 56.78.100.1 255.255.255.0
 ipv6 address 56:78:100::1/64
 mac-address 22.22.22
!
interface BVI102
 host-routing
 vrf cust102
 ipv4 address 56.78.102.1 255.255.255.0
 ipv6 address 56:78:100::1/64
 ipv6 address 56:78:102::1/64
 mac-address 22.22.22
!

/* Configure CEF */ [ Required for Dual homing]
cef adjacency route override rib

/* Configure EVPN */
evpn
 evi 102
 bgp
 rd 1605:102
 route-target import 160102:102
 route-target export 160102:102
 !
 advertise-mac
 !
 !
!
interface Bundle-Ether1
 ethernet-segment
 identifier type 0 56.56.56.56.56.56.56.56.01
 !
 !
 interface Bundle-Ether2
 ethernet-segment
 identifier type 0 56.56.56.56.56.56.56.56.02
 !
 !

/* Configure L2VPN */

l2vpn
 bridge group bg102
 bridge-domain bd102
 interface Bundle-Ether1.102
 !
 interface Bundle-Ether2.102
 !
 interface Bundle-Ether3.102
 !
 interface Bundle-Ether4.102
 !
 interface Bundle-Ether5.102
 !
 routed interface BVI102
```

```
 !
 evi 102
 !
 !
 !
!
```

Verification

Verify that you have configured EVPN IPv6 Hosts with Mobility feature is configured.

```
/* 6PE and Static Route Advertisement */
Host route is advertised as EVPN Route Type 2

Router# show bgp ipv6 unicast 56:78:100::2
BGP routing table entry for 56:78:100::2/128
Versions:
 Process bRIB/RIB SendTblVer
 Speaker 212 212
 Local Label: 2
Last Modified: Oct 31 19:13:10.998 for 00:00:19
Paths: (1 available, best #1)
 Not advertised to any peer
 Path #1: Received by speaker 0
 Not advertised to any peer
 Local
 160.5.5.5 (metric 20) from 160.0.0.1 (160.0.0.5)
 Received Label 2
 Origin IGP, localpref 100, valid, internal, best, group-best, imported
 Received Path ID 0, Local Path ID 0, version 212
 Extended community: Flags 0x20: SoO:160.5.5.5:100 RT:160100:100
 mac: 00:06:01:00:01:02
 Originator: 160.0.0.5, Cluster list: 100.0.0.4
 Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 1605:100

/* Manually configured static route in global VRF */

Router# show bgp ipv6 unicast 56:78:100::2

BGP routing table entry for 30::1/128
Versions:
 Process bRIB/RIB SendTblVer
 Speaker 9 9
 Local Label: 2
Last Modified: Oct 30 20:25:17.159 for 23:15:55
Paths: (2 available, best #2)
 Advertised to update-groups (with more than one peer):
 0.2
 Path #1: Received by speaker 0
 Not advertised to any peer
 Local
 160.0.0.6 (metric 20) from 160.0.0.1 (160.0.0.6)
 Received Label 2
 Origin incomplete, metric 0, localpref 100, valid, internal, labeled-unicast
 Received Path ID 0, Local Path ID 0, version 0
 mac: 10:11:04:64:f2:7f
 Originator: 160.0.0.6, Cluster list: 100.0.0.4
 Path #2: Received by speaker 0
 Advertised to update-groups (with more than one peer):
 0.2
 Local
 56:78:100::2 from :: (160.0.0.5)
 Origin incomplete, metric 0, localpref 100, weight 32768, valid, redistributed, best,
```

```
group-best
 Received Path ID 0, Local Path ID 0, version 9
 mac: 10:11:04:64:f2:7f

/* Verify Ethernet Segments are peering for Dual homing */

Router# show evpn ethernet-segment int bundle-Ether 1

Ethernet Segment Id Interface Nexthops
----------------------- ---------------------------------- --------------------
0056.5656.5656.5656.5601 BE1 160.5.5.5
                             160.6.6.6
------------------------------------------------------------

/* Verify DF election */

Router# show evpn ethernet-segment int bundle-Ether 1 carving detail
Legend:
 A - Load-balancing mode and Access Protection incompatible,
 B - No Forwarders EVPN-enabled,
 C - Backbone Source MAC missing (PBB-EVPN),
 RT - ES-Import Route Target missing,
 E - ESI missing,
 H - Interface handle missing,
 I - Name (Interface or Virtual Access) missing,
 M - Interface in Down state,
 O - BGP End of Download missing,
 P - Interface already Access Protected,
 Pf - Interface forced single-homed,
 R - BGP RID not received,
 S - Interface in redundancy standby state,
 X - ESI-extracted MAC Conflict
 SHG - No local split-horizon-group label allocated

Ethernet Segment Id Interface Nexthops
----------------------- ---------------------------------- --------------------
0056.5656.5656.5656.5601 BE1 160.5.5.5
 160.6.6.6
 ES to BGP Gates : Ready
 ES to L2FIB Gates : Ready
 Main port :
 Interface name : Bundle-Ether1
 Interface MAC : 008a.9644.acdd
 IfHandle : 0x080004dc
 State : Up
 Redundancy : Not Defined
 ESI type : 0
 Value : 56.5656.5656.5656.5601
 ES Import RT : 5656.5656.5656 (from ESI)
 Source MAC : 0000.0000.0000 (N/A)
 Topology :
 Operational : MH
 Configured : All-active (AApF) (default)
 Primary Services : Auto-selection
 Secondary Services: Auto-selection
 Service Carving Results:
 Forwarders : 161
 Permanent : 10
 EVI:ETag P : 700:1, 701:1, 702:1, 703:1, 704:1, 705:1
 EVI:ETag P : 706:1, 707:1, 708:1, 709:1
 Elected : 76
 EVI E : 100, 102, 104, 106, 108, 110
 EVI E : 112, 114, 116, 118, 120, 122,
 EVI E : 124, 126, 128, 130, 132, 134,
```

```
EVI E : 136, 138, 140, 142, 144, 146,
EVI E : 148, 150, 152, 154, 156, 158,
EVI E : 160, 162, 164, 166, 168, 170,
EVI E : 172, 174, 176, 178, 180, 182,
EVI E : 184, 186, 188, 190, 192, 194,
EVI E : 196, 198, 200, 202, 204, 206,
EVI E : 208, 210, 212, 214, 216, 218,
EVI E : 220, 222, 224, 226, 228, 230,
EVI E : 232, 234, 236, 238, 240, 242,
EVI E : 244, 246, 248, 250
Not Elected : 75
EVI NE : 101, 103, 105, 107, 109, 111
EVI NE : 113, 115, 117, 119, 121, 123,
EVI NE : 125, 127, 129, 131, 133, 135,
EVI NE : 137, 139, 141, 143, 145, 147,
EVI NE : 149, 151, 153, 155, 157, 159,
EVI NE : 161, 163, 165, 167, 169, 171,
EVI NE : 173, 175, 177, 179, 181, 183,
EVI NE : 185, 187, 189, 191, 193, 195,
EVI NE : 197, 199, 201, 203, 205, 207,
EVI NE : 209, 211, 213, 215, 217, 219,
EVI NE : 221, 223, 225, 227, 229, 231,
EVI NE : 233, 235, 237, 239, 241, 243,
EVI NE : 245, 247, 249
MAC Flushing mode : STP-TCN
Peering timer : 3 sec [not running]
Recovery timer : 30 sec [not running]
Carving timer : 0 sec [not running]
Local SHG label : 68663
Remote SHG labels : 1
68670 : nexthop 160.6.6.6
```

Duplicate IP Address Detection

The Duplicate IP Address Detection feature automatically detects any host with a duplicate IP address and blocks all MAC-IP routes that have a duplicate IP address.

This protects the network from hosts that are assigned duplicate IP addresses unintentionally or by malicious intent in an EVPN fabric. Hosts with duplicate IP address cause unnecessary churn in a network and causes traffic loss to either or both the hosts with the same IP address.

The system handles mobility of EVPN hosts by keeping track of MAC and IP addresses as they move from one host to another. If two hosts are assigned the same IP address, the IOS XR system keeps learning and re-learning MAC-IP routes from both the hosts. Each time it learns the MAC-IP route from one host, it is counted as one move since the newly learnt route supersedes the route previously learnt from the other host. This continues back and forth until the IP address is marked as duplicate based on the configured parameters.

It uses the following parameters to determine when an IP address should be marked as duplicate, and frozen or unfrozen as it moves between different hosts. The configurable parameters are:

- **move-interval**: The period within which a MAC or IP address has to move certain number of times between different hosts to be considered as duplicate and frozen temporarily. This number is specified in the **move-count** parameter.
- **move-count**: The number of times a MAC or IP address has to move within the interval specified for the **move-interval** parameter between different hosts to be considered a duplicate.

- **freeze-time**: The length of time a MAC or IP address is locked after it has been detected as a duplicate. After this period, the IP address is unlocked and it is allowed to learn again.
- **retry-count**: The number of times a MAC or IP address is unlocked after it has been detected as a duplicate before it is frozen permanently.

The system maintains a count of the number of times an IP address has been moved from one host to another host, either to another local host or to a host behind a remote Top of Rack (TOR). If an IP address moves certain number of times specified in the **move-count** parameter within the interval specified in the **move-interval** parameter is considered a duplicate IP address. All MAC-IP routes with that IP address is frozen for the time specified in the **freeze-time** parameter. A syslog notifies the user that the particular IP address is frozen. While an IP address is frozen, any new MAC-IP routes or updates to existing MAC-IP routes with the frozen IP address are ignored.

After **freeze-time** has elapsed, the corresponding MAC-IP routes are unfrozen and the value of the **move-count** is reset to zero. For any unfrozen local MAC-IP routes, an ARP probe and flush are initiated while the remote MAC-IP routes are put in the probe mode. This restarts the duplicate detection process.

The system also maintains the information about the number of times a particular IP address has been frozen and unfrozen. If an IP address is marked as duplicate after it is unfrozen **retry-count** times, it is frozen permanently until user manually unfreezes it. Use the following commands to manually unfreeze frozen MAC, IPv4 and IPV6 addresses respectively:

- **clear l2route evpn mac** { *mac-address* } | **all** [**evi** *evi*] **frozen-flag**
- **clear l2route evpn ipv4** { *ipv4-address* } | **all** [**evi** *evi*] **frozen-flag**
- **clear l2route evpn ipv6** { *ipv6-address* } | **all** [**evi** *evi*] **frozen-flag**

Configure Duplicate IP Address Detection

Perfrom these tasks to configure Duplicate IP Address Detection feature.

Configuration Example

```
/* Ipv4 Address Duplicate Detection Configuration */
Router# configure
Router(config)# evpn
Router(config-evpn)# host ipv4-address duplicate-detection
Router(config-evpn-host-ipv4-addr)# move-count 2
Router(config-evpn-host-ipv4-addr)# freeze-time 10
Router(config-evpn-host-ipv4-addr)# retry-count 2
Router(config-evpn-host-ipv4-addr)# commit

/* Ipv6 Address Duplicate Detection Configuration */
Router# configure
Router(config)# evpn
Router(config-evpn)# host ipv6-address duplicate-detection
Router(config-evpn-host-ipv6-addr)# move-count 2
Router(config-evpn-host-ipv6-addr)# freeze-time 10
Router(config-evpn-host-ipv6-addr)# retry-count 2
Router(config-evpn-host-ipv6-addr)# commit
```

Running Configuration

This section shows the running configuration to detect duplicate IP address.

```
evpn
 host ipv4-address duplicate-detection
  move-count 2
  freeze-time 10
  retry-count 2
 !
evpn
 host ipv6-address duplicate-detection
  move-count 2
  freeze-time 10
  retry-count 2
 !
```

Verification

The show output given in the following section display the details of the duplicate IP address detection and recovery parameters.

```
Router#show l2route evpn mac-ip all detail

Flags:  (Stt)=Static; (L)=Local; (R)=Remote; (F)=Flood;
        (N)=No Redistribution; (Rtr)=Router MAC; (B)=Best Route;
        (S)=Peer Sync; (Spl)=Split; (Rcv)=Recd;
        (D)=Duplicate MAC; (Z)=Frozen MAC;

Topo ID    Mac Address     IP Address  Prod   Next Hop(s)        Seq No  Flags
Opaque Data Type    Opaque Data Len   Opaque Data Value
-------    -----------     ----------  ----   ----------         ------  -----
----------------    ---------------   -----------------
33         0022.6730.0001  10.130.0.2  L2VPN  Bundle-Ether6.1300  0      SB 0 12
0x06000000
```

Related Topics

- Duplicate IP Address Detection, on page 397

Associated Commands

- evpn host ipv4-address duplicate-detection
- evpn host ipv6-address duplicate-detection
- show l2route evpn mac-ip all detail

EVPN Automatic Unfreezing of MAC and IP Addresses

The EVPN Automatic Unfreezing of MAC and IP Addresses feature unfreezes the permanently frozen MAC and IP addresses automatically. This feature provides a configurable option to enable a MAC or IP address to undergo infinite duplicate detection and recovery cycles without being frozen permanently. The MAC or IP address is permanently frozen when duplicate detection and recovery events occur three times within a 24-hour window. If any of the duplicate detection events happen outside the 24-hour window, the MAC or IP address undergoes only one duplicate detection event and all previous events are ignored.

Use the **infinity** keyword to prevent freezing of the duplicate MAC or IP address permanently.

Example

```
host ipv4-address duplicate-detection retry-count infinity
host ipv6-address duplicate-detection retry-count infinity
host mac-address duplicate-detection retry-count infinity
```

Use the **no** form of the above command to enable permanent freezing of MAC or IP address after the default retry count.

Example

```
no host ipv4-address duplicate-detection retry-count infinity
no host ipv6-address duplicate-detection retry-count infinity
no host mac-address duplicate-detection retry-count infinity
```

The 24-hour check for consecutive duplicate detection and recovery events before permanent freezing is enabled by default. Use the **reset-freeze-count-interval** keyword to configure a non-default interval after which the retry-count is reset. The range is from is 1 hour to 48 hours. The default is 24 hours.

Example

```
host ipv4-address duplicate-detection reset-freeze-count-interval 20
host ipv6-address duplicate-detection reset-freeze-count-interval 20
host mac-address duplicate-detection reset-freeze-count-interval 20
```

Use the following commands to manually unfreeze frozen MAC, IPv4 and IPV6 addresses respectively:

- **clear l2route evpn mac** { *mac-address* } | **all** [**evi** *evi*] **frozen-flag**
- **clear l2route evpn ipv4** { *ipv4-address* } | **all** [**evi** *evi*] **frozen-flag**
- **clear l2route evpn ipv6** { *ipv6-address* } | **all** [**evi** *evi*] **frozen-flag**

Configure EVPN Automatic Unfreezing of MAC or IP Address

Infinite duplicate detection and recovery is disabled by default. However, you can enable it using the following configuration.

Configuration Example

```
/* IPv4 Address Duplicate Detection Configuration */
Router# configure
Router(config)# evpn
Router(config-evpn)# host ipv4-address duplicate-detection
Router(config-evpn-host-ipv4-addr)# move-count 5
Router(config-evpn-host-ipv4-addr)# move-interval 180
Router(config-evpn-host-ipv4-addr)# freeze-time 30
Router(config-evpn-host-ipv4-addr)# retry-count 3
Router(config-evpn-host-ipv4-addr)# reset-freeze-count-interval 24
Router(config-evpn-host-ipv4-addr)# commit

/* IPv6 Address Duplicate Detection Configuration */
Router# configure
Router(config)# evpn
Router(config-evpn)# host ipv4-address duplicate-detection
Router(config-evpn-host-ipv6-addr)# move-count 5
```

```
Router(config-evpn-host-ipv6-addr)# move-interval 180
Router(config-evpn-host-ipv6-addr)# freeze-time 30
Router(config-evpn-host-ipv6-addr)# retry-count 3
Router(config-evpn-host-ipv6-addr)# reset-freeze-count-interval 24
Router(config-evpn-host-ipv6-addr)# commit

/* MAC Address Duplicate Detection Configuration */
Router# configure
Router(config)# evpn
Router(config-evpn)# host MAC-address duplicate-detection
Router(config-evpn-host-mac-addr-dup-detection)# move-count 5
Router(config-evpn-host-mac-addr-dup-detection)# freeze-time 30
Router(config-evpn-host-mac-addr-dup-detection)# move-interval 180
Router(config-evpn-host-mac-addr-dup-detection)# retry-count infinite
Router(config-evpn-host-mac-addr-dup-detection)# reset-freeze-count-interval 24
Router(config-evpn-host-mac-addr-dup-detection)# commit
```

Running Configuration

This section shows the EVPN automatic unfreezing of MAC or IP address running configuration.

```
evpn
 host ipv4-address duplicate-detection
  move-count 5
  freeze-time 30
  retry-count 3
  reset-freeze-count-interval 24
 !
evpn
 host ipv6-address duplicate-detection
  move-count 5
  freeze-time 30
  retry-count 3
 !
evpn
 host mac-address duplicate-detection
  move-count 5
  freeze-time 30
  move-interval 180
  reset-freeze-count-interval 24
 !
```

Verification

The show output given in this section display the details of the duplicate MAC and IP address detection and recovery parameters.

```
Router#show l2route summary
…
Duplicate Detection Parameters
------ ---------- -------- ------- ---------- ---------- ----------------
Type   Disabled   Freeze   Move    Move       Retry      Freeze-Count
                  Time     Count   Interval   Count      Reset-Interval
------ ---------- -------- ------- ---------- ---------- ----------------
MAC    False      30       5       180        Infinite   24
IPv4   False      30       5       180        3          24
IPv6   False      30       5       180        3          24
--------------------------------------------------------------------------
```

Related Topics

EVPN Automatic Unfreezing of MAC and IP Addresses, on page 399

Associated Commands

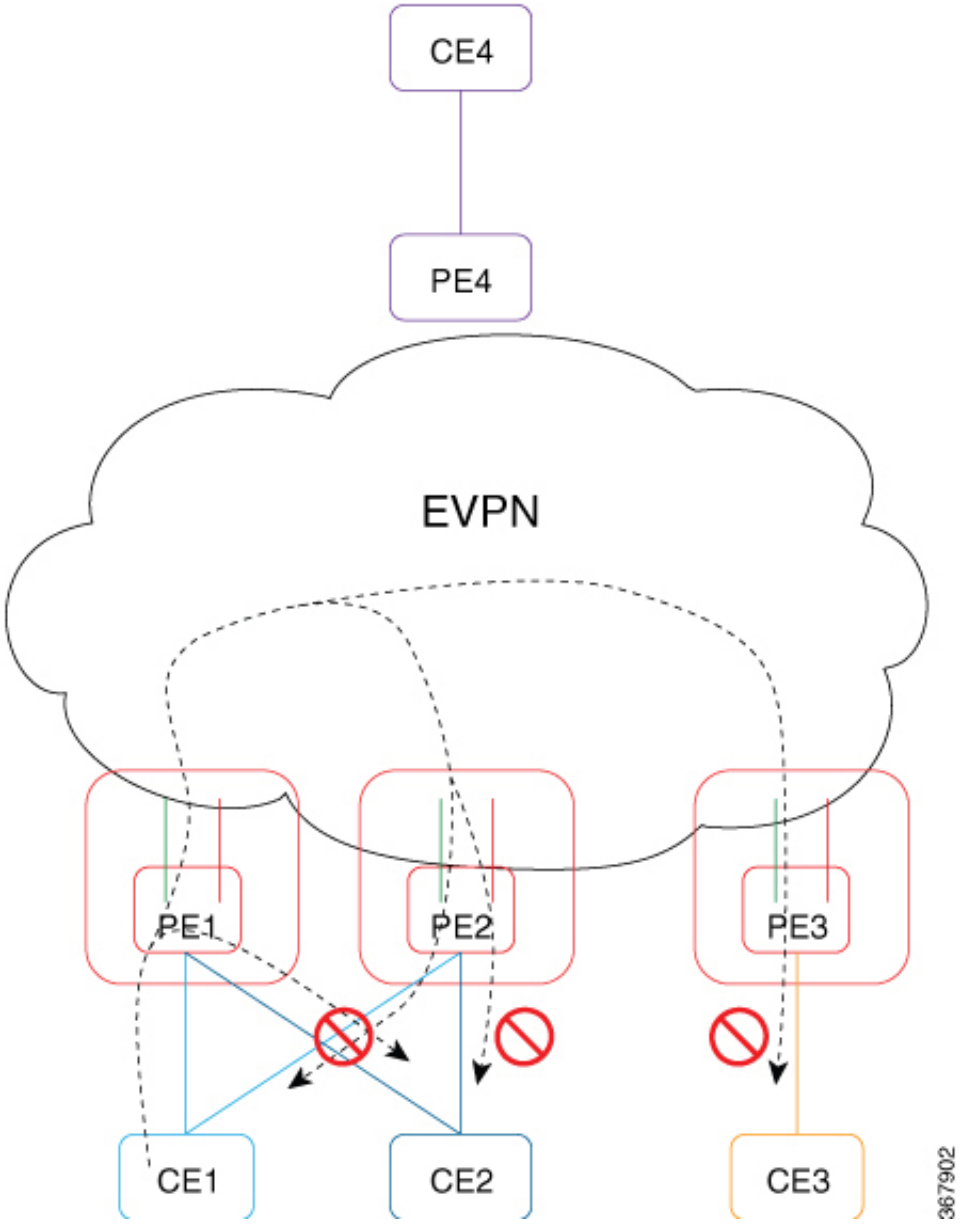
- host ipv4-address duplicate-detection
- host ipv6-address duplicate-detection
- host mac-address duplicate-detection
- show l2route summary

EVPN E-Tree

The EVPN E-Tree feature provides a rooted-multipoint Ethernet service over MPLS core. The EVPN Ethernet Tree (E-Tree) service enables you to define attachment circuits (ACs) as either a root site or a leaf site, which helps in load balancing and avoiding loops in a network.

In this topology, consider PE1, PE2, and PE3 as leaf ACs, and PE4 as root AC. Root ACs can communicate with all other ACs. Leaf ACs can communicate with root ACs but not with other leaf ACs with either L2 unicast or L2 BUM traffic. If a PE is not configured as E-Tree leaf, it is considered as root by default. This feature only supports leaf or root sites per PE.

Figure 60: EVPN E-Tree



E-Tree leaf is configured for each EVI Bridge Domain (BD). Root and leaf EVI of BD exports or imports single Routed Targets (RTs). The configuration of E-Tree leaf per EVI implies the following:

- All ACs inherit the leaf indicator.
- Split-horizon group between the ACs (leaf) on same EVI is enabled automatically.
- Each PE leaf advertises per Ethernet Segment per Ethernet Auto Discovery Route (ES-EAD), Ethernet Segment Identifier (ESI), ES-EAD ESI 0 route with leaf indicator and E-Tree label to BGP.
- All local MACs learned under this EVI are re-advertised to BGP with E-Tree leaf indicator.
- Each PE maintains a list of remote PEs.

Note

- If you modify the E-Tree leaf configuration, all the locally learned MAC addresses are flushed out. All the locally learned MAC addresses are flushed out even when bridge port's "encapsulation" or "rewrite" on sub-interface, or "split-horizon group" configuration is modified under the bridge port.
- A BVI interface configured in the bridge domain always exhibits the root behaviour.

Unicast Rules

The following table describes the unicast rules upon reception of type-2 MAC route on root and leaf.

MAC Route Received	MAC Route Handling
MAC address with non-local ESI from root EVI (BD)	Remote MAC address.
MAC address with local ESI from root EVI (BD)	MAC address synhronization, re-originate.
MAC address with non-local ESI from leaf EVI (BD)	Remote MAC address. Remote MAC route with leaf indicator is dropped.
MAC address with local ESI from leaf EVI (BD)	MAC address synhronization, re-originate. MAC address points to the local AC. Upon local AC failure, synchronization MAC route becomes a remote MAC route. Remote MAC route with leaf indicator is dropped as opposed to pointing to a peering PE.

Multicast Rules

Multicast is used to discover the leaf in the network when:

- RT-1 ES-EAD ESI-0 route with E-Tree extended community is sent per EVI (BD) to indicate to other network PEs which EVIs are setup as E-Tree leaf.
- RT-1 ES-EAD ESI-0 route with E-Tree extended community route and RT-3 IMCAST route are received on a leaf EVI (BD).

Note Per local EVI (BD) split-horizon group prevents local AC to AC traffic flow.

Communication between CE1 and CE4 (Inter-subnet)

1. CE1 sends an ARP request to its gateway, which is IRB interface. CE1 resolves the BVI IP address.
2. ARP request reaches the bridge domain on PE1. It learns the entry and floods it.
3. ARP requests to all remote PEs that have been pruned is dropped. It is replicated to all root remote PEs and to local BVI interface.
4. BVI interface on PE1 sends an ARP response to CE1 using its BVI IP address and BVI MAC address.

5. At the same time, since host routing is configured, PE1 advertises CE1 host route through EVPN using route type-2.
6. After receiving type-2 route, different rules apply based on the PE. After receiving route type-2 on:
   a. PE2: MAC and IP address of ESI match local ESI. Program MAC address as synchronization route. Program IP address in RIB to point to PE1, but MAC address points to CE1. Upon link failure to CE1, MAC address is marked as dropped in the hardware instead of pointing to peering PE1.
   b. PE3: MAC and IP address of ESI are not local. Since local EVI (BD) is leaf, MAC address is marked as dropped in the hardware. Program IP address in RIB pointing to PE1.
   c. PE4: MAC and IP address of ESI are not local. Since local EVI (BD) is root, program MAC as remote. Program IP address in RIB pointing to PE1.
7. PE4 is aware of CE1. CE1 and CE4 communicate with each other.
8. For example, a routing packet coming from CE4 reaches PE4. An IP lookup is performed. PE1 is found as the best destination due to the host route /32. The packet is forwarded to PE1.
9. On PE1, an IP lookup is performed. The BVI interface is found. The packet is encapsulated with CE1 as destination MAC address as learned by ARP. Source MAC address remains as the BVI MAC address. Destination MAC address lookup is performed in the corresponding bridge domain. The packet is forwarded to proper output interface.

Note If CE4 sends packet to CE1 before CE1 starts communication, the packet may go to peering PE2. GLEAN adjacency is affected and traffic is dropped until it is resolved. To resolve the entry, PE2 BVI interface starts probing.

1. ARP probing coming from BVI is sent to all ACs and EVI as well (L2 stretch).
2. PE1 and PE3 receive the ARP probe from EVI interface and replicate to all local ACs. CE1 sends ARP reply where PE1 BVI interface accepts it since IRB on all the leafs are configured in a distributed anycast gateway.

Communication between CE1 and CE3 (Intra-subnet)

1. CE1 and CE3 are within the same subnet.
2. CE1 sends an ARP request to CE3.
3. ARP request reaches the bridge domain on PE1. It learns the entry and floods it.
4. ARP requests for all remote PEs that have been pruned is dropped. It is replicated to all root remote PEs and to local BVI interface.
5. CE3 does not receive ARP request from CE1. CE1 with does not communicate with CE3.
6. If you want CE1 and CE3 to communicate within intra-subnet, then you must configure local_proxy_arp under BVI interface on both local and remote PEs.

Communicatiion between CE1 and CE2 (Intra-subnet)

1. CE1 and CE2 are within the same subnet.
2. CE1 sends an ARP request to CE2.
3. ARP request reaches the bridge domain on PE1. It learns the entry and floods it.
4. ARP requests for all remote PEs that have been pruned is dropped. It is not replicated to any local ACs due to common split-horizon group.
5. CE2 does not receive ARP request from CE1. CE1 does not communication with CE2.

Note Communication between local CE1 and remote CE1:

- The BUM traffic from local CE1 on PE1 to remote CE1 on PE2 is dropped as PE2 is pruned.
- The BUM traffic from local CE1 on PE1 to local CE1 on PE1 in the case of AC-Aware VLAN bundling feature is dropped due to ESI-filtering.

Configure EVPN E-Tree

Perform this task to configure EVPN E-Tree feature on the leaf PEs.

```
/* Configure EVPN E-Tree service on PE1 and PE2 */

Router# configure
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-evi)# etree leaf
```

Configuration Example

```
/* Configure EVPN Multihoming on PE1 and PE2*/

Router# configure
Router(config)# evpn
Router(config-evpn)# interface bundle-Ether 1121
Router(config-evpn-ac)# ethernet-segment  identifier type 0  20.00.00.00.00.00.00.11.21

/* Configure AC interface on PE1 and PE2*/

Router(config)# interface Bundle-Ether1121.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric

/* Configure BVI interface on PE1 and PE2 */

Router(config)# interface BVI1
Router(config-if)# host-routing
Router(config-if)# vrf vpn1
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# proxy-arp
```

```
Router(config-if)# local-proxy-arp
Router(config-if)# ipv6 address 2001:DB8::1/32
Router(config-if)# mac-address 10.1111.aaaa
Router(config-if)# load-interval 30

/* Configure the bridge on PE1 and PE2 */

Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1121.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# routed interface BVI1
Router(config-l2vpn-bg-bd-bvi)# exit
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-evi)# etree leaf
Router(config-evpn-instance)# commit
```

Running Configuration

This section shows EVPN E-Tree running configuration.

```
/* EVPN E-Tree running configuration on PE1 *
evpn
 evi 1
  etree
    leaf
  !
 !
 interface Bundle-Ether1121
  ethernet-segment
   identifier type 0 20.00.00.00.00.00.00.11.21


l2vpn
 bridge group bg1
  bridge-domain bd1
   interface Bundle-Ether1121.1
   routed interface BVI1
  !
  evi 1

interface Bundle-Ethe1121.1
l2transport
 encapsulation dot1q 1
 rewrite ingress tag pop 1 symmetric
 !
!
interface BVI1
 host-routing
 vrf vpn1
 ipv4 address 192.0.2.1 255.255.255.0
 proxy-arp
 local-proxy-arp
 ipv6 address 2001:DB8::1/32
 mac-address 10.1111.aaaa
 load-interval 30
 !
!


/* EVPN E-Tree running configuration On PE2 */
```

```
evpn
 evi 1
  etree
   leaf
  !
 !
 interface Bundle-Ether1121
  ethernet-segment
   identifier type 0 20.00.00.00.00.00.00.11.21


l2vpn
 bridge group bg1
  bridge-domain bd1
   interface Bundle-Ether1121.1
   routed interface BVI1
  !
  evi
 !
interface Bundle-Ethe1121.1
l2transport
 encapsulation dot1q 1
 rewrite ingress tag pop 1 symmetric
 !
!
interface BVI1
 host-routing
 vrf vpn1
 ipv4 address 192.0.2.1 255.255.255.0
 proxy-arp
 local-proxy-arp
 ipv6 address 2001:DB8::1/32
 mac-address 10.1111.aaaa
 load-interval 30
 !
!
```

Verification

The show output given in the following section display the details of the EVPN E-Tree configuration.

```
Router#show bgp l2vpn evpn rd 10.0.0.1:0
Route Distinguisher: 10.0.0.1:0
*> [1][10.0.0.1:1][0000.0000.0000.0000.0000][4294967295]/184
                     0.0.0.0                                0 i
*> [1][10.0.0.1:2][0000.0000.0000.0000.0000][4294967295]/184
                     0.0.0.0                                0 i

Each RT-1 ES0 has up to 200 RTs. Two RT-1 ES0 is displayed if you have 250 RTs.
```

The following output shows Leaf excom advertised in RT-1 ES0.

```
Router#show bgp l2vpn evpn rd 10.0.0.1:0
[1][10.0.0.1:1][0000.0000.0000.0000.0000][4294967295]/184
Extended community: EVPN E-TREE:0x00:824348 RT:100:1 RT:100:2 RT:100:3 RT:100:4 RT:100:5
RT:100:10 RT:100:11
RT:100:12 RT:100:13 RT:100:14 RT:100:15 RT:100:16 RT:100:17 RT:100:18 RT:100:19 RT:100:20
RT:100:21 RT:100:22 RT:100:23
RT:100:24 RT:100:25 RT:100:26 RT:100:27 RT:100:28 RT:100:29 RT:100:30 RT:100:31 RT:100:32
RT:100:33 RT:100:34 RT:100:35
```

```
RT:100:36 RT:100:37 RT:100:38 RT:100:39 RT:100:40 RT:100:41 RT:100:42 RT:100:43 RT:100:44
RT:100:45 RT:100:46 RT:100:47
 RT:100:48 RT:100:49 RT:100:50
```

The following output shows RT-2 of MAC advertisement.

```
Router#show bgp l2vpn evpn rd 10.0.0.1:1 [2][1][48][0011.1100.0001][0]/104
Paths: (2 available, best #1)
  Advertised to peers (in unique update groups):
    172.16.0.1
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    172.16.0.1
  Local
    0.0.0.0 from 0.0.0.0 (10.0.0.1)
      Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
 rib-install
      Received Path ID 0, Local Path ID 1, version 315227
      Extended community: SoO:192.168.0.1:1 EVPN E-TREE:0x01:0 RT:100:1
      EVPN ESI: 0020.0000.0000.0000.1121
```

The following output shows one RT-2 of MAC address and IP address advertisement.

```
Router#show bgp l2vpn evpn rd 10.0.0.1:1 [2][1][48][0011.1100.0001][32][101.0.1.103]/136
Tue Oct  2 16:44:26.755 EDT
BGP routing table entry for [2][1][48][0011.1100.0001][32][101.0.1.103]/136, Route
Distinguisher: 10.0.0.1:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker             313139      313139
    Local Label: 820002
Last Modified: Oct  2 13:26:08.477 for 03:18:18
Paths: (2 available, best #1)
  Advertised to peers (in unique update groups):
    172.16.0.1
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    172.16.0.1
  Local
    0.0.0.0 from 0.0.0.0 (10.0.0.1)
      Second Label 825164
      Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
 rib-install
      Received Path ID 0, Local Path ID 1, version 313139
     Extended community: Flags 0xe: SoO:192.168.0.1:1 EVPN E-TREE:0x01:0 RT:100:1 RT:991:1

      EVPN ESI: 0020.0000.0000.0000.1121
```

The following output shows aggregation of RT-3 inclusive-multicast and RT-1 ES0 routes in EVPN.

```
Router#show evpn evi vpn-id 1 inclusive-multicast detail
1          MPLS   0          192.168.0.1
    TEPid  : 0x02000001
    PMSI Type: 0
    Nexthop: 192.168.0.1
    Label  : 810120
    Source : Remote
    E-Tree: Leaf
1          MPLS   0          10.0.0.1
    TEPid  : 0xffffffff
    PMSI Type: 6
    Nexthop: ::
    Label  : 820120
```

```
    Source : Local
    E-Tree: Leaf
1           MPLS   0          172.16.0.1
    TEPid  : 0x02000003
    PMSI Type: 0
    Nexthop: 172.16.0.1
    Label  : 840120
    Source : Remote
    E-Tree: Root
```

Related Topics

- EVPN E-Tree, on page 402

Associated Commands

- etree leaf
- show bgp l2vpn evpn rd

EVPN E-Tree Using RT Constraints

The EVPN E-Tree using RT constraints feature enables you to configure BGP RT import and export policies for an attachment circuit. This feature allows you to define communication between the leaf and root nodes. The provider edge (PE) nodes can receive L2 traffic either from the attachment circuit (AC) of a bridge domain (BD) or from the remote PE node. For a given BD, L2 communication can only happen from root to leaf and leaf to root. This feature does not allow any L2 communication between the ACs of two or more leafs. This feature uses two BGP RTs for every EVI. Associate one RT with root ACs and the other with leaf ACs. For example, there are two distinct sets of RTs, one for root-rt and another for leaf-rt.

This feature provides you with the following benefits by performing filtering of unicast and multicast traffic at the ingress PE nodes:

- Achieve efficiency of the BGP MAC routes scale
- Reduce the consumption of hardware resources
- Utilize the link bandwidth efficiently

Rules for Import and Export Policies under the BGP of EVPN EVI Instances

- Root PE exports its ROOT-RT using BGP export policy. It also imports other ROOT-RT from the corresponding root PE for the same EVI. This is necessary where there is more than one root for a particular BD and EVPN EVI. For example, in a multihome active-active scenario or multihome port-active and single-active scenarios.
- Root PE imports LEAF-RT using BGP import policy for a EVPN EVI. This enables the root to be aware of all remote L2 MAC addresses through EVPN RT2 advertisement of leaf PE node for a given E-Tree EVI.
- Leaf PE exports its LEAF-RT using BGP export policy to let the root to be aware of the reachability of its directly connected L2 endpoints through EVPN RT2 advertisement.

- Leaf PE imports ROOT-RT using BGP import policy. It helps the leaf to know about the L2 endpoints which are reachable through the AC of BD under EVPN EVI instance of root PE. You must not import LEAF-RT using BGP Import policy to avoid L2 Communication between two leaf PEs.
- Use split-horizon filtering to block traffic among leaf ACs on a BD for a given E-Tree EVI.

The BGP import and export policies applies to all EVPN RTs along with the RT2 advertisement.

MAC Address Learning

- L2 MAC addresses are learnt on AC of a particular BD on leaf PE as type LOCAL. The same MAC address is advertised to root PE as EVPN RT2. On the remote root PE, the MAC table replicates the entry of MAC address with the learn type as L2VPN. Also, it associates the MPLS label of its BGP peer, which advertises RT2 to root PE node.
- L2 MAC addresses are learnt on AC of a particular BD on the root as type LOCAL. The same MAC address is advertised to peer root (except for MH A/A) or leaf PE as EVPN RT2. On the remote root PE or leaf PE, the MAC table replicates the entry of MAC address with the learn type as L2VPN. Also, it associates the MPLS label of its BGP peer, which advertises RT2 to PE node.
- L2 MAC addresses are learnt on AC of a particular BD on the root as type LOCAL. The same MAC address is advertised to peer root for MH A/A as EVPN RT2. The MAC table of the peer root node synchronizes the replicated entry of MAC address with the learn type as L2VPN for same the ESI and with the same AC as the next hop. This avoids flooding and duplication of known unicast traffic.

The following scenario describes the feature topology::

CE with Multihoming Active-Active and CE with Multihoming Active-Active

Consider a topology where you connect CE-02 and CE-03 to PE-01 and PE-02. Both the CEs are in multihoming active-active mode. Connect CE-02 to PE-01 and PE-02 using AC BE-800.305. Connect CE-03 to PE-01 and PE-02 using AC BE-820.305. Connect CE-06 and CE-07 to PE-03 and PE-04. Connect CE-06 to PE-03 and PE-04 using AC BE-700.305. Connect CE-07 to PE-03 and PE-04 using AC BE-720.305. Associate the bridge domain BD-305 with other AC on the respective PEs along with EVI-305 instance. Configure the respective RT on root and leaf with its import and export RTs for EVI-305. Configure PE-01 and PE-02 as root. Configure PE-03 and PE-04 as leaf.

As you are using EVPN E-Tree with RT constraints and rt-leaf indicator set, the rt-leaf configuration causes EVPN to add the ES-import-RT to the mac-only RT-2s to support All-Active syncing for a Bundle Multihomed by Leafs. This is required when using RT constraints, otherwise the PE can end up flooding unicast traffic to its local ACs forever.

```
evpn evi 2001
  etree
   rt-leaf
```

```
RP/0/RP0/CPU0:router# show evpn evi vpn-id 2001 mac
VPN-ID  Encap  MAC address    IP address    Nexthop                Label    SID
------  -----  -------------- ----------    -------------------    -----    ---
2001    MPLS   0000.0100.0003 ::            Bundle-Ether11.2002    26064
```

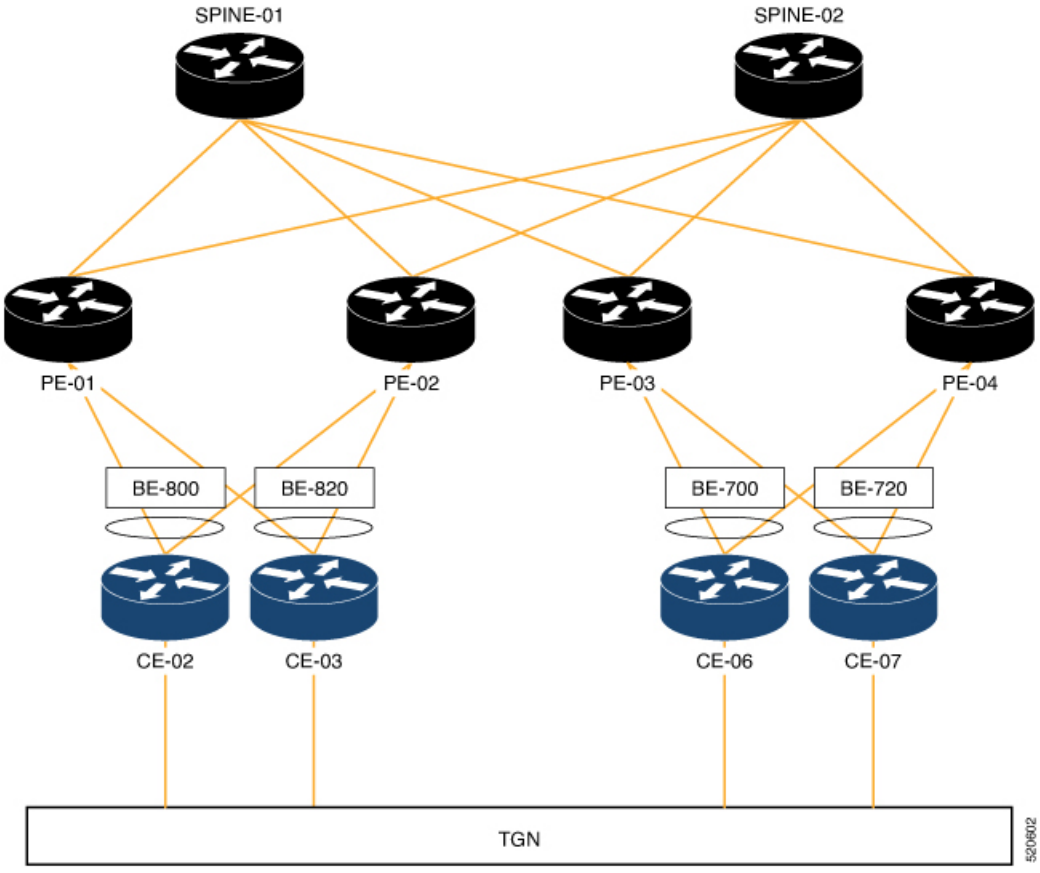
```
RP/0/RP0/CPU0:router# show l2route evpn mac all detail

Flags: (Stt)=Static; (L)=Local; (Lp)=Local-Proxy;
       (R)=Remote;N)=No Redistribution; (Rtr)=Router MAC;
       (B)=Best Route;(S)=Peer Sync; (Spl)=Split; (Rcv)=Recd;
```

```
       (D)=Duplicate MAC; (Z)=Frozen MAC;(Sfa)=Single Flow Active
       (A)=Access; (Gw)=Gateway;

Topo ID  Mac Address     Producer   Next Hop(s)               Seq No Flags   Slot       ESI
  Opaque Data Type  Opaque Data Len  Opaque Data Value
-------  --------------  --------   ------------------------  ------ -----   --------   ---
  ----------------  ---------------  -----------------
16       0000.0100.0003  LOCAL      Bundle-Ether11.2002, N/A  0      BLRcv   0/0/CPU0   (F)
   N/A               N/A              N/A

Last Update: Tue Mar 28 12:58:00.730
```

```
Router# show evpn ethernet-segment interface bundle-Ether 11 carving private
Legend:
Ethernet Segment Id Interface Nexthops
0010.1010.1010.1010.1011 BE11 172.16.45.3
172.16.45.4
ES to BGP Gates : Ready
ES to L2FIB Gates : Ready
Main port :
Interface name : Bundle-Ether11
Interface MAC : bc2c.e654.b8dc
IfHandle : 0x20008034
State : Up
Redundancy : Not Defined
ESI ID : 0x2
ESI type : 0
Value : 0010.1010.1010.1010.1011
ES Import RT : 1010.1010.1010 (from ESI)
Source MAC : 0000.0000.0000 (N/A)
Topology :
Operational : MH, All-active
Configured : All-active (AApF) (default)
Service Carving : Auto-selection
Multicast : Disabled
Convergence : Reroute
Peering Details : 2 Nexthops
172.16.45.3 [MOD:P:00:T][2]
172.16.45.4 [MOD:P:7fff:T][0]
```

```
Router# show bgp l2vpn evpn rd [2][0][48][0000.0100.0003][0]/104 DET
BGP routing table entry for [2][0][48][0000.0100.0003][0]/104, Route Distinguisher:
172.16.45.4:2001
Versions:
Process bRIB/RIB SendTblVer
Speaker 5373 5373
Local Label: 26064 (no rewrite);
Flags: 0x00040001+0x00000000;
Last Modified: Jun 2 03:42:14.557 for 2d12h
Paths: (1 available, best #1)
Advertised to update-groups (with more than one peer):
0.2
Path #1: Received by speaker 0
Flags: 0x202002000504000b+0x00, import: 0x000, EVPN: 0x1
Advertised to update-groups (with more than one peer):
0.2
Local
0.0.0.0 from 0.0.0.0 (172.16.45.4), if-handle 0x00000000
Origin IGP, localpref 100, valid, redistributed, best, group-best,
import-candidate, rib-install
Received Path ID 0, Local Path ID 1, version 5367
Extended community: SoO:172.16.45.4:2001 EVPN ES Import:1010.1010.1010 EVI RT:fd84.0000.07d1
0x060e:0000.007d.2fff RT:2001:64900
EVPN ESI: 0010.1010.1010.1010.101
```

```
RP/0/RP0/CPU0:router# show evpn evi mac 0000.0100.0003 detail

Tue Apr 4 16:50:33.090 NZST

VPN-ID  Encap  MAC address     IP address  Nexthop              Label  SID
------  -----  --------------  ----------  -------------------  -----  ---
2001    MPLS   0000.0100.0003  ::          Bundle-Ether11.2002  26064

   Ethernet Tag                          : 0
   Multi-paths Resolved                  : False
   Multi-paths Internal label            : 0
   Local Static                          : No
   Remote Static                         : No
   Local Ethernet Segment                : 0010.1010.1010.1010.1011
   Remote Ethernet Segment               : N/A
   Local Sequence Number                 : 0
   Remote Sequence Number                : N/A
   Local Encapsulation                   : MPLS
   Remote Encapsulation                  : N/A
   Local E-Tree                          : Leaf
   Remote E-Tree                         : Root
   Remote matching E-Tree RT             : No
   Local AC-ID                           : 0x7d2fff
   Remote AC-ID                          : 0x0
```



Configuration

Perform the following tasks on PE-01, PE-02, PE-03, and PE-04.

- Configure bridge domain
- Configure attachment circuit
- Configure EVPN EVI
- Configure bundle Ethernet
- Configure EVPN interface

Note Use the **etree rt-leaf** command only if the leaf sites are in the EVPN all-active multihoming mode and not required for EVPN single homing mode.

Configuration Example

```
/* Configure PE-01 (as root) */

/* Configure bridge domain */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN_BD
Router(config-l2vpn-bg)# bridge-domain evpn_bvi_305
Router(config-l2vpn-bg-bd)# interface Bundle-Ether800.305
Router(config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# interface Bundle-Ether820.305
Router(config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit

/* Configure attachment circuit */
Router# configure
Router(config)# interface Bundle-Ether800.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

Router# configure
Router(config)# interface Bundle-Ether820.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure EVPN EVI */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target import 1001:305
Router(config-evpn-instance-bgp)# route-target export 1001:305
Router(config-evpn-instance-bgp)# route-target import 1001:5305
Router(config-evpn-instance-bgp)# exit

Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit

/* Configure bundle Ethernet */
Router# configure
```

```
Router(config)# interface Bundle-Ether800
Router(config-if)# lacp system mac 00aa.aabb.2020
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

Router# configure
Router(config)# interface Bundle-Ether820
Router(config-if)# lacp system mac 00aa.aabb.2222
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

/* Configure EVPN interface */
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether800
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.88.88.88.88.88.88.88.00
Router(config-evpn-ac-es)# bgp route-target 0001.0000.0001
Router(config-evpn-ac-es)# commit

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether820
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.88.88.88.88.88.88.88.20
Router(config-evpn-ac-es)# bgp route-target 0001.0000.0020
Router(config-evpn-ac-es)# commit


/* Configure PE-02 (as root) */

/* Configure bridge domain */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN_BD
Router(config-l2vpn-bg)# bridge-domain evpn_bvi_305
Router(config-l2vpn-bg-bd)# interface Bundle-Ether800.305
Router(config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# interface Bundle-Ether820.305
Router(config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit

/* Configure attachment circuit */
Router# configure
Router(config)# interface Bundle-Ether800.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

Router# configure
Router(config)# interface Bundle-Ether820.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure EVPN EVI */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
```

```
Router(config-evpn-instance-bgp)# route-target import 1001:305
Router(config-evpn-instance-bgp)# route-target export 1001:305
Router(config-evpn-instance-bgp)# route-target import 1001:5305
Router(config-evpn-instance-bgp)# exit

Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit

/* Configure bundle Ethernet */
Router# configure
Router(config)# interface Bundle-Ether800
Router(config-if)# lacp system mac 00aa.aabb.2020
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

Router# configure
Router(config)# interface Bundle-Ether820
Router(config-if)# lacp system mac 00aa.aabb.2222
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

/* Configure EVPN interface */
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether800
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.88.88.88.88.88.88.88.00
Router(config-evpn-ac-es)# bgp route-target 0001.0000.0001
Router(config-evpn-ac-es)# commit

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether820
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.88.88.88.88.88.88.88.20
Router(config-evpn-ac-es)# bgp route-target 0001.0000.0020
Router(config-evpn-ac-es)# commit


/* Configure PE-03 (as leaf) */

/* Configure bridge domain */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN_BD
Router(config-l2vpn-bg)# bridge-domain evpn_bvi_305
Router(config-l2vpn-bg-bd)# interface Bundle-Ether700.305
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router (config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface Bundle-Ether720.305
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router (config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit

/* Configure attachment circuit */
Router# configure
Router(config)# interface Bundle-Ether700.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit
```

```
Router# configure
Router(config)# interface Bundle-Ether720.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure EVPN EVI */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target import 1001:305
Router(config-evpn-instance-bgp)# route-target export 1001:5305
Router(config-evpn-instance-bgp)# exit
Router(config-evpn-instance)# etree
Router(config-evpn-instance-etree)# rt-leaf
Router(config-evpn-instance)# exit

Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit

/* Configure bundle Ethernet */
Router# configure
Router(config)# interface Bundle-Ether700
Router(config-if)# lacp system mac 00aa.aabb.1010
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

Router# configure
Router(config)# interface Bundle-Ether720
Router(config-if)# lacp system mac 00aa.aabb.1212
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

/* Configure EVPN interface */
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether700
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.77.77.77.77.77.77.77.00
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0001
Router(config-evpn-ac-es)# commit

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether720
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.77.77.77.77.77.77.77.20
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0020
Router(config-evpn-ac-es)# commit


/* Configure PE-04 (as leaf) */

/* Configure bridge domain */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN_BD
Router(config-l2vpn-bg)# bridge-domain evpn_bvi_305
Router(config-l2vpn-bg-bd)# interface Bundle-Ether700.305
```

```
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router (config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface Bundle-Ether720.305
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router (config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit

/* Configure attachment circuit */
Router# configure
Router(config)# interface Bundle-Ether700.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

Router# configure
Router(config)# interface Bundle-Ether720.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure EVPN EVI */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target import 1001:305
Router(config-evpn-instance-bgp)# route-target export 1001:5305
Router(config-evpn-instance-bgp)# exit
Router(config-evpn-instance)# etree
Router(config-evpn-instance-etree)# rt-leaf
Router(config-evpn-instance)# exit

Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit

/* Configure bundle Ethernet */
Router# configure
Router(config)# interface Bundle-Ether700
Router(config-if)# lacp system mac 00aa.aabb.1010
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

Router# configure
Router(config)# interface Bundle-Ether720
Router(config-if)# lacp system mac 00aa.aabb.1212
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

/* Configure EVPN interface */
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether700
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.77.77.77.77.77.77.77.00
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0001
Router(config-evpn-ac-es)# commit

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether720
```

```
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.77.77.77.77.77.77.77.20
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0020
Router(config-evpn-ac-es)# commit
```

Running Configuration

This section shows the PE-01, PE-02, PE-3, and PE-04 running configuration.

```
/* PE-01 Configuration */
l2vpn
 bridge group EVPN_BD
  bridge-domain evpn_bvi_305
   interface Bundle-Ether800.305
   !
   interface Bundle-Ether820.305
   !
   evi 305
   !
  !
 !
interface Bundle-Ether800.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether820.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
evpn
 evi 305
  bgp
   route-target import 1001:305
   route-target export 1001:305
   route-target import 1001:5305
  !

  advertise-mac
  !
 !
!
interface Bundle-Ether800
 lacp system mac 00aa.aabb.2020
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether820
 lacp system mac 00aa.aabb.2222
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether800
  ethernet-segment
   identifier type 0 00.88.88.88.88.88.88.88.00
   bgp route-target 0001.0000.0001
  !
 !
!
evpn
 interface Bundle-Ether820
```

```
  ethernet-segment
   identifier type 0 00.88.88.88.88.88.88.88.20
   bgp route-target 0001.0000.0020
  !
 !
!


/* PE-02 Configuration */
l2vpn
 bridge group EVPN_BD
  bridge-domain evpn_bvi_305
   interface Bundle-Ether800.305
   !
   interface Bundle-Ether820.305
   !
   evi 305
   !
  !
 interface Bundle-Ether800.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether820.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
evpn
 evi 305
  bgp
   route-target import 1001:305
   route-target export 1001:305
   route-target import 1001:5305
  !

  advertise-mac
  !
 !
interface Bundle-Ether800
 lacp system mac 00aa.aabb.2020
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether820
 lacp system mac 00aa.aabb.2222
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether800
  ethernet-segment
   identifier type 0 00.88.88.88.88.88.88.88.00
   bgp route-target 0001.0000.0001
  !
 !
evpn
 interface Bundle-Ether820
  ethernet-segment
   identifier type 0 00.88.88.88.88.88.88.88.20
   bgp route-target 0001.0000.0020
  !
 !
```

```
/* PE-03 Configuration */
l2vpn
 bridge group EVPN_BD
  bridge-domain evpn_bvi_305
   interface Bundle-Ether700.305
    split-horizon group
   !
   interface Bundle-Ether720.305
    split-horizon group
   !
   evi 305
   !
  !
 !
interface Bundle-Ether700.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether720.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
evpn
 evi 305
  bgp
   route-target import 1001:305
   route-target export 1001:5305
  !
  etree
   rt-leaf
  !

  advertise-mac
  !
 !
!
interface Bundle-Ether700
 lacp system mac 00aa.aabb.1010
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether720
 lacp system mac 00aa.aabb.1212
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether700
  ethernet-segment
   identifier type 0 00.77.77.77.77.77.77.77.00
   bgp route-target 0000.0000.0001
  !
 !
!
evpn
 interface Bundle-Ether720
  ethernet-segment
   identifier type 0 00.77.77.77.77.77.77.77.20
   bgp route-target 0000.0000.0020
  !
```

```
 !
!


/* PE-04 Configuration */
l2vpn
 bridge group EVPN_BD
  bridge-domain evpn_bvi_305
   interface Bundle-Ether700.305
    split-horizon group
   !
   interface Bundle-Ether720.305
    split-horizon group
   !
   evi 305
   !
  !
 !
interface Bundle-Ether700.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether720.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
evpn
 evi 305
  bgp
   route-target import 1001:305
   route-target export 1001:5305
  !
  etree
   rt-leaf
  !

  advertise-mac
  !
 !
!
interface Bundle-Ether700
 lacp system mac 00aa.aabb.1010
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether720
 lacp system mac 00aa.aabb.1212
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether700
  ethernet-segment
   identifier type 0 00.77.77.77.77.77.77.77.00
   bgp route-target 0000.0000.0001
  !
 !
!
evpn
 interface Bundle-Ether720
  ethernet-segment
   identifier type 0 00.77.77.77.77.77.77.77.20
   bgp route-target 0000.0000.0020
```

```
  !
 !
!
```

Verification

This section shows how the L2 MAC addresses are synchronized as LOCAL and L2VPN with multihoming active-active peers PE. Also, the root PE is aware of MAC addresses learnt on leaf PE remotely through RT2 advertisements.

```
Router:PE-01# show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- --------------------------------------
204      001f.0100.0001 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0001 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0002 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0002 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0003 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0003 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0004 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0004 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0005 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0005 L2VPN       Bundle-Ether820.305, N/A
204      0020.0100.0001 L2VPN       26791/I/ME, N/A
204      0020.0100.0002 L2VPN       26791/I/ME, N/A
204      0020.0100.0003 L2VPN       26791/I/ME, N/A
204      0020.0100.0004 L2VPN       26791/I/ME, N/A
204      0020.0100.0005 L2VPN       26791/I/ME, N/A
204      0021.0100.0001 L2VPN       Bundle-Ether800.305, N/A
204      0021.0100.0002 L2VPN       Bundle-Ether800.305, N/A
204      0021.0100.0003 LOCAL       Bundle-Ether800.305, N/A
204      0021.0100.0004 L2VPN       Bundle-Ether800.305, N/A
204      0021.0100.0005 LOCAL       Bundle-Ether800.305, N/A
204      0022.0100.0001 L2VPN       26790/I/ME, N/A
204      0022.0100.0002 L2VPN       26790/I/ME, N/A
204      0022.0100.0003 L2VPN       26790/I/ME, N/A
204      0022.0100.0004 L2VPN       26790/I/ME, N/A
204      0022.0100.0005 L2VPN       26790/I/ME, N/A

Router:PE-02# show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- --------------------------------------
204      001f.0100.0001 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0001 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0002 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0002 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0003 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0003 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0004 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0004 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0005 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0005 L2VPN       Bundle-Ether820.305, N/A
204      0020.0100.0001 L2VPN       27367/I/ME, N/A
204      0020.0100.0002 L2VPN       27367/I/ME, N/A
204      0020.0100.0003 L2VPN       27367/I/ME, N/A
204      0020.0100.0004 L2VPN       27367/I/ME, N/A
204      0020.0100.0005 L2VPN       27367/I/ME, N/A
204      0021.0100.0001 LOCAL       Bundle-Ether800.305, N/A
204      0021.0100.0002 LOCAL       Bundle-Ether800.305, N/A
204      0021.0100.0003 L2VPN       Bundle-Ether800.305, N/A
204      0021.0100.0004 LOCAL       Bundle-Ether800.305, N/A
204      0021.0100.0005 L2VPN       Bundle-Ether800.305, N/A
```

```
204      0022.0100.0001 L2VPN       27366/I/ME, N/A
204      0022.0100.0002 L2VPN       27366/I/ME, N/A
204      0022.0100.0003 L2VPN       27366/I/ME, N/A
204      0022.0100.0004 L2VPN       27366/I/ME, N/A
204      0022.0100.0005 L2VPN       27366/I/ME, N/A
```

The following output shows how the multihoming PE is aware of its local L2 MAC addresses as well as the MAC addresses learnt on the root node only. Leaf multihoming PE is not aware of any other MAC addresses learnt on other leaf PE nodes except if they are learnt on a multihoming active-active ethernet-segment on the peer leaf PE.

```
Router:PE-03# show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- --------------------------------------
200      0011.0100.0003 L2VPN       30579/I/ME, N/A
200      0011.0100.0005 L2VPN       30579/I/ME, N/A
204      001f.0100.0001 L2VPN       30588/I/ME, N/A
204      001f.0100.0002 L2VPN       30588/I/ME, N/A
204      001f.0100.0003 L2VPN       30588/I/ME, N/A
204      001f.0100.0004 L2VPN       30588/I/ME, N/A
204      001f.0100.0005 L2VPN       30588/I/ME, N/A
204      0020.0100.0001 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0001 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0002 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0002 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0003 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0003 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0004 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0004 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0005 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0005 L2VPN       Bundle-Ether720.305, N/A
204      0021.0100.0001 L2VPN       30587/I/ME, N/A
204      0021.0100.0002 L2VPN       30587/I/ME, N/A
204      0021.0100.0003 L2VPN       30587/I/ME, N/A
204      0021.0100.0004 L2VPN       30587/I/ME, N/A
204      0021.0100.0005 L2VPN       30587/I/ME, N/A
204      0022.0100.0001 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0001 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0002 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0002 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0003 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0003 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0004 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0004 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0005 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0005 L2VPN       Bundle-Ether700.305, N/A

Router:PE-04# show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- --------------------------------------
200      0011.0100.0003 L2VPN       30545/I/ME, N/A
200      0011.0100.0005 L2VPN       30545/I/ME, N/A
204      001f.0100.0001 L2VPN       30550/I/ME, N/A
204      001f.0100.0002 L2VPN       30550/I/ME, N/A
204      001f.0100.0003 L2VPN       30550/I/ME, N/A
204      001f.0100.0004 L2VPN       30550/I/ME, N/A
204      001f.0100.0005 L2VPN       30550/I/ME, N/A
204      0020.0100.0001 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0001 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0002 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0002 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0003 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0003 L2VPN       Bundle-Ether720.305, N/A
```

```
204      0020.0100.0004 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0004 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0005 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0005 L2VPN       Bundle-Ether720.305, N/A
204      0021.0100.0001 L2VPN       30549/I/ME, N/A
204      0021.0100.0002 L2VPN       30549/I/ME, N/A
204      0021.0100.0003 L2VPN       30549/I/ME, N/A
204      0021.0100.0004 L2VPN       30549/I/ME, N/A
204      0021.0100.0005 L2VPN       30549/I/ME, N/A
204      0022.0100.0001 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0001 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0002 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0002 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0003 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0003 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0004 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0004 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0005 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0005 L2VPN       Bundle-Ether700.305, N/A
```

Related Topics

- #unique_337

Associated Commands

- etree rt-leaf
- show l2route evpn mac all

EVPN E-Tree Per-PE (Scenario 1b)

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-Tree Per-PE (Scenario 1b)	Release 7.5.1	This feature allows you to configure an attachment circuit on a PE device either as a root site or a leaf site using the **etree leaf** label for an EVPN Instance (EVI) or for a given bridge-domain. By preventing communication among leaf ACs connected to the same PE and belonging to the same EVI, you can segregate traffic received and sent from different geographical locations. This segregation helps in load balancing traffic and avoiding traffic from going into loops in a network.

EVPN Ethernet Tree (E-Tree) is a rooted-multipoint Ethernet service over MPLS core and enables you to define attachment circuits (ACs) as either a root site or a leaf site. The provider edge (PE) nodes can receive L2 traffic either from the attachment circuit (AC) of a bridge domain (BD) or from the remote PE node. For a given BD or EVI, L2 communication can only happen from root to leaf and leaf to root, and root to root. L2 communication between the ACs of two or more leafs is not allowed.

You can implement E-Tree in the following two ways:

- Scenario 1 - All ACs at a particular PE for a given EVI or BD can be either root or leaf site and all traffic for an EVI from a PE in the network is from either a root or a leaf. In this scenario you have two options to configure E-Tree:
  - Scenario 1a - You can configure E-Tree with route-targets (RT) constraints using two RTs per EVI. For more information, see the *EVPN E-Tree Using RT Constraints* section.
  - Scenario 1b - You can configure E-Tree without route-targets (RT) constraints and using **etree leaf** label.

Scenario 1b

In this scenario, you can configure E-Tree without route-targets (RT) constraints and using **etree leaf** label.

For known unicast traffic, MAC advertisements originating from a leaf site is identified with an **etree leaf** label to classify that the source is a leaf. Ingress filtering is performed, and traffic originating at leaf AC destined for a remote leaf MAC is dropped. If the remote PE is also a leaf, the ingress traffic from the source leaf is dropped. If the remote PE is a root, the ingress traffic from the source leaf is forwarded.

For BUM traffic, egress filtering is performed and leaf nodes transmit an **etree leaf** label to identify that leaf sites are connected to the PE. Then, at the ingress node, BUM traffic originating from a leaf node is tagged with the corresponding remote **etree leaf** label. At the egress PE, traffic is tagged with the matching **etree leaf** label that is dropped at leaf ACs.

For E-Tree with IRB, BVI interfaces are considered as a root site. However, if you configure the PE as a leaf site that has ACs with BVI, ingress filtering is performed instead of egress filtering as defined by Option B in RFC 8317. Tagging of ingress BUM traffic with **etree leaf** label is performed for the packets destined for a remote node.

Scenario 1b Behavior

- E-Tree leaf is configured per bridge domain or EVI. No leaf configuration means the bridge domain or EVI is a root.
- All ACs inherit E-Tree leaf designation from the bridge domain or EVI.
- Split-horizon group between ACs of a leaf is enabled automatically.
- All local MACs learned under the BD or EVI is advertised to BGP with **etree leaf** indicator.
- Upon first leaf configuration, a special E-Tree ethernet segment with ESI-0 is created to allocate a split-horizon label, referred to as the local etree leaf label.
- ES/EAD with ESI-0 (ES-0/EAD) is advertised to BGP with etree leaf label.
- EVPN E-Tree with IRB is also supported, but BVI interfaces are always treated as a root sites, even if BD or EVI itself is a leaf.

Restrictions

- If a BVI interface is part of a bridge domain, we recommend you to configure **etree leaf** under EVPN EVI configuration. When BVI is associated with an AC, etree leaf under the bridge domain is not supported due to hardware limitation.
- If an AC is not associated with BVI under a bridge domain, you can configure **etree leaf** under EVPN EVI or bridge domain configuration.

- Scenario 1a and Scenario 1b with IRB may not interopeate with any other flavors on E-Tree; this misconfiguration cannot be detected automatically.
- For non-supported interfaces, such as VNI, BVI, and PW, are not be marked as a leaf when the BD is configured as a leaf.
- You cannot configure a BD as E-Tree leaf when AC is a BVI interface. If an IRB interface is required, you must use the Cisco implementation (which is non-RFC compliant) of E-Tree where the EVI is configured as an E-Tree leaf.

Configure EVPN E-Tree Per-PE (Scenario1b)

Perform this task to configure EVPN E-Tree Per-PE (Scenario1b).

Configure EVPN E-Tree leaf per bridge domain.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg_201
Router(config-l2vpn-bg)# bridge-domain bd_201
Router(config-l2vpn-bg-bd)# etree leaf
Router(config-l2vpn-bg-bd)# interface Bundle-Ether3501.3601
Router(config-l2vpn-bg-bd-ac)# evi 201
Router(config-l2vpnbg-bd-evi)# commit
```

Configure EVPN E-Tree leaf per EVI.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 200
Router(config-evpn-instance)# bgp route-target import 64600:200
Router(config-evpn-instance)# bgp route-target export 64600:200
Router(config-evpn-instance)# etree leaf
Router(config-evpn-instance)# control-word-disable
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit
```

Running Configuration

Note The E-Tree configuration under EVI is required only when BVI is used in the BD.

```
/* EVPN E-Tree leaf per bridge domain */
l2vpn
 bridge group bg_201
  bridge-domain bd_201
   etree
    leaf
   !
   interface Bundle-Ether3501.3601
   !
   evi 201
   !
  !
/* EVPN E-Tree leaf per EVI */
evpn
 evi 200
```

```
  bgp
   route-target import 64600:200
   route-target export 64600:200
  !
  etree
   leaf
  !
  control-word-disable
  advertise-mac
  !
 !
/* EVPN E-Tree with BVI */
l2vpn
 bridge group bg1
  bridge-domain bd1
   interface Bundle-Ether200.1
   !
   interface Bundle-Ether201.1001
   !
   routed interface BVI1
    split-horizon group core
   !
   evi 200
   !
  !
 !
```

Verification

Verify that the etree leaf is configured per bridge domain.

```
Router# show l2vpn bridge-domain bd-name bd_201 detail
Legend: pp = Partially Programmed.
Bridge group: etree_bg, bridge-domain: bd_201, id: 162, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: EVPN Native
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 64000, Action: none, Notification: syslog
  MAC limit reached: no, threshold: 75%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
  Split Horizon Group: none
  E-Tree: Leaf
  Dynamic ARP Inspection: disabled, Logging: disabled
  IP Source Guard: disabled, Logging: disabled
  DHCPv4 Snooping: disabled
  DHCPv4 Snooping profile: none
  IGMP Snooping: disabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  Storm Control: disabled
  Bridge MTU: 1500
  MIB cvplsConfigIndex: 163
  Filter MAC addresses:
  P2MP PW: disabled
```

```
  Multicast Source: Not Set
  Create time: 25/10/2021 15:50:01 (00:50:39 ago)
  No status change since creation
  ACs: 1 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
      evi: 1 (MPLS)
      XC ID 0x8000009f
     Statistics:
       packets: received 0 (unicast 0), sent 0
       bytes: received 0 (unicast 0), sent 0
       MAC move: 0
  List of ACs:
    AC: Bundle-Ether3501.3601, state is up (Segment-up)
      Type VLAN; Num Ranges: 1
      Rewrite Tags: []
      VLAN ranges: [3601, 3601]
      MTU 1500; XC ID 0xa00001e0; interworking none; MSTi 2
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 64000, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: enabled (inherited)
      E-Tree: Leaf (inherited)
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control: bridge-domain policer
      Static MAC addresses:
      PD System Data: AF-LIF-IPv4: 0x00000000  AF-LIF-IPv6: 0x00000000 FRR-LIF: 0x00000000

  List of Access PWs:
  List of VFIs:
  List of Access VFIs:


Router# show l2vpn bridge-domain bd-name bd1 detail
Legend: pp = Partially Programmed.
Bridge group: bg1, bridge-domain: bd1, id: 6, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: EVPN-IRB
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 64000, Action: none, Notification: syslog
  MAC limit reached: no, threshold: 75%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
  Split Horizon Group: none
  E-Tree: Leaf (inherited)
```

```
Dynamic ARP Inspection: disabled, Logging: disabled
IP Source Guard: disabled, Logging: disabled
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: disabled
Bridge MTU: 1500
MIB cvplsConfigIndex: 7
Filter MAC addresses:
P2MP PW: disabled
Multicast Source: Not Set
Create time: 25/10/2021 15:50:01 (00:47:35 ago)
No status change since creation
ACs: 3 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
List of EVPNs:
  EVPN, state: up
    evi: 200 (MPLS)
    XC ID 0x80000003
   Statistics:
     packets: received 0 (unicast 0), sent 0
     bytes: received 0 (unicast 0), sent 0
     MAC move: 0
List of ACs:
  AC: BVI1, state is up (Segment-up)
    Type Routed-Interface
    MTU 1514; XC ID 0x800007d3; interworking none
    BVI MAC address:
      0011.1111.1111
    Split Horizon Group: Core
    PD System Data: AF-LIF-IPv4: 0x00000000  AF-LIF-IPv6: 0x00000000 FRR-LIF: 0x00000000

  AC: Bundle-Ether200.1, state is up (Segment-up)
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1500; XC ID 0xa00000be; interworking none; MSTi 9
    MAC learning: enabled
    Flooding:
      Broadcast & Multicast: enabled
      Unknown unicast: enabled
    MAC aging time: 300 s, Type: inactivity
    MAC limit: 64000, Action: none, Notification: syslog
    MAC limit reached: no, threshold: 75%
    MAC port down flush: enabled
    MAC Secure: disabled, Logging: disabled
    Split Horizon Group: enabled (inherited)
    E-Tree: Leaf (inherited)
    Dynamic ARP Inspection: disabled, Logging: disabled
    IP Source Guard: disabled, Logging: disabled
    DHCPv4 Snooping: disabled
    DHCPv4 Snooping profile: none
    IGMP Snooping: disabled
    IGMP Snooping profile: none
    MLD Snooping profile: none
    Storm Control: bridge-domain policer
    Static MAC addresses:
    PD System Data: AF-LIF-IPv4: 0x00012678  AF-LIF-IPv6: 0x00012679 FRR-LIF: 0x00000000

  AC: Bundle-Ether201.1001, state is up (Segment-up)
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1001, 1001]
```

```
    MTU 1500; XC ID 0xa000017c; interworking none; MSTi 9
    MAC learning: enabled
    Flooding:
      Broadcast & Multicast: enabled
      Unknown unicast: enabled
    MAC aging time: 300 s, Type: inactivity
    MAC limit: 64000, Action: none, Notification: syslog
    MAC limit reached: no, threshold: 75%
    MAC port down flush: enabled
    MAC Secure: disabled, Logging: disabled
    Split Horizon Group: enabled (inherited)
    E-Tree: Leaf (inherited)
    Dynamic ARP Inspection: disabled, Logging: disabled
    IP Source Guard: disabled, Logging: disabled
    DHCPv4 Snooping: disabled
    DHCPv4 Snooping profile: none
    IGMP Snooping: disabled
    IGMP Snooping profile: none
    MLD Snooping profile: none
    Storm Control: bridge-domain policer
    Static MAC addresses:
    PD System Data: AF-LIF-IPv4: 0x0001272c  AF-LIF-IPv6: 0x0001272d FRR-LIF: 0x00000000

List of Access PWs:
List of VFIs:
List of Access VFIs:
```

DHCPv4 Relay on IRB

DHCPv4 Relay on Integrated Routing and Bridging (IRB) feature provides DHCP support for the end users in EVPN all-active multihoming scenario. This feature enables reduction of traffic flooding, increase in load sharing, optimize traffic, faster convergence during link and device failures, and simplification of data center automation.

DHCPv4 relay agent sends request packets coming over access interface towards external DHCPv4 server to request address (/32) allocation for the end user. DHCPv4 relay agent acts as stateless for end users by not maintaining any DHCPv4 binding and respective route entry for the allocated address.

DHCPv4 relay profiles are configured on bridge-group virtual interface (BVI) interfaces which act as access interfaces by integrating routing and bridge domains for the end users. It relays DHCPv4 requests from Layer 2 attachment circuit (AC) to external DHCP servers for host IPv4 addresses (/32).

Multihoming All-Active EVPN Gateways

Multihoming all-active EVPN gateways are configured with anycast IP address and MAC addresses. The Cisco routers have centralized L2 or L3 gateway. Based on native EVPN and MAC learning, IRB uses distributed anycast IP address and anycast MAC address. Static clients are configured with anycast gateway address as the default gateway. DHCP client sends DHCP requests for IP address allocation over the BVI interface. L2 access can be either single homing or multihoming, not all access protocols are supported with IRB. BVI IP address acts as a default gateway for the end user. The external DHCPv4 server provides this BVI interface IP address as default gateway in route options. No EVPN is configured on the Internet gateway.

EVPN IRB Route Distribution

In EVPN IRB DHCPv4, DHCP application processes and DHCP packet forwarding are independent of EVPN IRB L2 and L3 routing. There is no subscriber routing information with the stateless DHCP relay. But DHCP clients work similar to static clients in the EVPN core for L2 and L3 bridging and routing. When the **relay**

information option and **relay information option vpn** commands are configured on the DHCP relay agent, the DHCP relay agent inserts the sub options of DHCP Option 82, such as subnet selection and VPN ID options. These options are considered by DHCP server while allocating the IP addresses.

The IP address allocation for the end user at DHCPv4 server is based on **relay agent information** option (Remote-ID+ Circuit-ID) values. DHCP clients use the L2 AC interface to access EVPN bridge domain and use BVI interface as default gateway. So the clients must get the IP addresses from the DHCP server from the same subnet of BVI interface.

After the DHCPv4 application receive the access side DHCPv4 packets over BVI interface based on **relay-option policy {encapsulate | drop | keep}** command, DHCPv4 application includes option-82 Relay-Agent Information, Remote-ID, and Circuit-ID for DHCPv4 Server.

The following table provides the attributes that qualify the DHCPv4 relay packets for the configured Relay-Information details. The information given in the table is used for configuring **relay-option policy {encapsulate | drop | keep}** command.

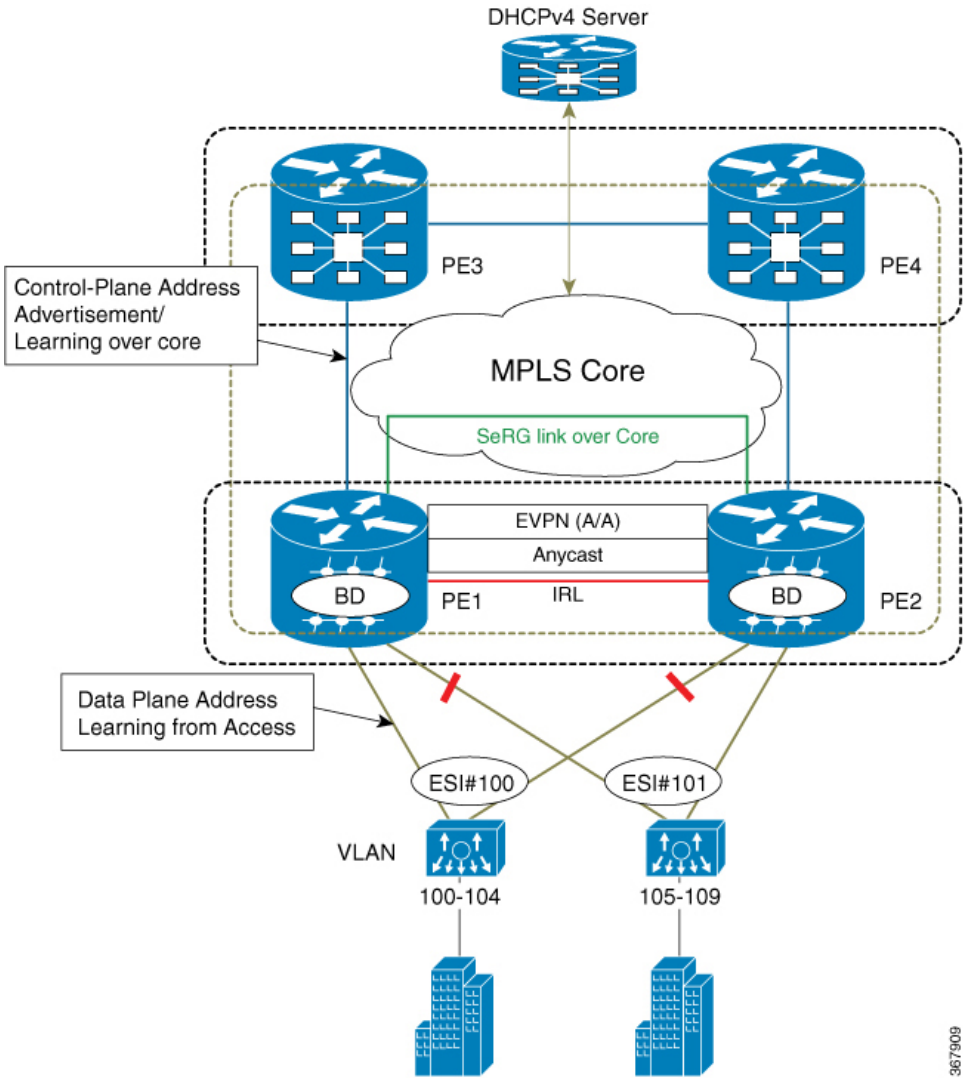
Relay-Option Policy	DHCPv4 Access Side Packet	Local Configuration	DHCPv4 Relay Packet Decision
Encapsulate	No Relay-Information	DHCPv4-Profile with Remote-ID L2Transport AC with Circuit-ID	Relay-Agent with Remote-ID and Circuit-ID
Encapsulate	Relay-Information (Remote-ID and Circuit-ID)	DHCPv4-Profile with Remote-ID L2Trasnsport AC with Circuit-ID	Override Relay-Agent Information with Local Configuration (Remote-ID and Circuit-ID)
Encapsulate	No Relay-Information	DHCPv4-Profile with Remote-ID and VPN-Information L2Transport AC with Circuit-ID	Relay-Agent with Remote-ID, Circuit-ID and VPN-Information
Keep	Relay-Information (Remote-ID and Circuit-ID)	No configuration	DHCPv4 Relay-Agent does not change any Relay-Information
Keep	Relay-Information (Remote-ID and Circuit-ID)	DHCPv4-Profile with Remote-ID L2 Transport AC with Circuit-ID	DHCPv4 Relay-Agent does not change any Relay-Information
Keep	Relay-Information (Remote-ID and Circuit-ID)	DHCPv4-Profile with Remote-ID and VPN-Information L2 Transport AC with Circuit-ID	DHCPv4 Relay-Agent does not change any Relay-Information

Relay-Option Policy	DHCPv4 Access Side Packet	Local Configuration	DHCPv4 Relay Packet Decision
Drop	Relay-Information (Remote-ID and Circuit-ID)	No configuration	Exclude Relay-Agent Information and include `None` in Relayed-Packet
Drop	Relay-Information (Remote-ID and Circuit-ID)	DHCPv4-Profile with Remote-ID L2 Transport AC with Circuit-ID	Exclude Relay-Agent Information and include `None` in Relayed-Packet
Drop	Relay-Information (Remote-ID and Circuit-ID)	DHCPv4-Profile with Remote-ID and VPN-Information L2 Transport AC with Circuit-ID	Exclude Relay-Agent Information and include `None` in Relayed-Packet

DHCP Request Forwarding Path

Clients broadcast requests to the access switch with DH-AA to EVPN PE routers. The access switch does load balancing. The load balancing configurations in access switch impacts PE in DH-AA and DHCP to send the DHCP requests. The DHCP request reaches the Bridge Domain (BD) BVI interface which is configured with DHCP relay. Because all-active PE routers are configured with the same IP address, BVI IP addresses cannot be used as DHCP relay source IP address. For DHCPv4 relay, access (BVI) interface is tied-up with relay profile. The device intercept packets are received over BVI interface and each relay profile is defined with Gateway IP Address (GIADDR), which acts as source IP address for initiated relayed packets towards DHCPv4 server. This GIADDR is unique across Top of Racks (ToRs) for respective BVI interfaces. Loopback interface with unique IPv4 address can be configured in VRF that is reachable to DHCP servers. Configuring DHCP relay source address is not supported.

Figure 61: PON behavior in handling DHCPv4 Server for EVPN All-Active Multihoming



PON behavior in handling DHCPv4 Server for EVPN All-Active Multihoming

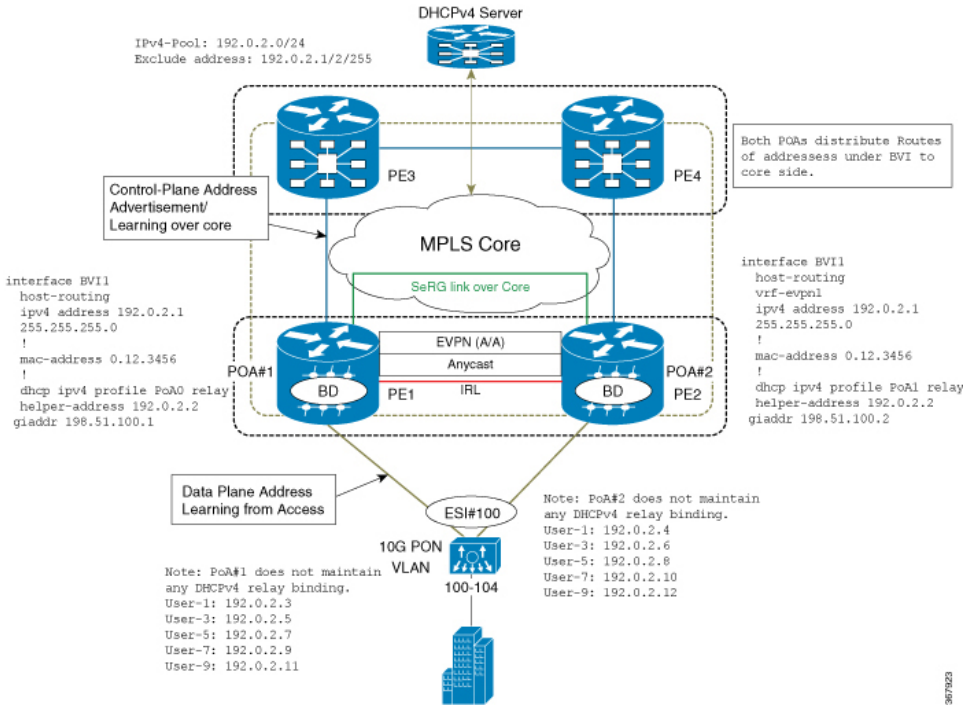
In this topology, PE1 and PE2 are edge routers for access side, which serve CEs (10G-OLT) over BVI interfaces by associating routing and bridging domains to process DHCPv4 packets. CEs (L2 OLT, PONs, any L2 domain switches) hashes the incoming control packets (DHCPv4 packets) towards port channels that are connected to respective PEs. The CEs leverage the hashing mechanism based on five tuples (src mac, dst mac, src-ip, dst-ip, L4 (tcp/udp) dst/src port) of packets that are received from the end user. Defines the forwarding mechanism by selecting the port channel on load balancing the control packets to respective PEs in dual-home active-active model.

DHPCv4 Relay Handling for EVPN and DHCPv4 Server in Default VRF

DHCPv4 relay over EVPN IRB and DHCPv4 servers resides in the same default VRFs. The DHCPv4 relay profiles are associated with helper-addresses of DHCPv4 address under default VRFs. In this particular scenario, PEs do not include any relay-agent information in relayed DHCPv4 packets towards DHCPv4 server.

However, DHCPv4 relay profile is defined in unique GIADDR across ToRs other than the anycast IRB address. Else, it is difficult for DHCPv4 server to perform address allocation for end user of not having link selection or subnet selection. The PEs include relay-agent information by including VPN information with VPN value as 0xFF.

Figure 62: DHPCv4 Relay Handling for EVPN and DHCPv4 Server in Default VRF

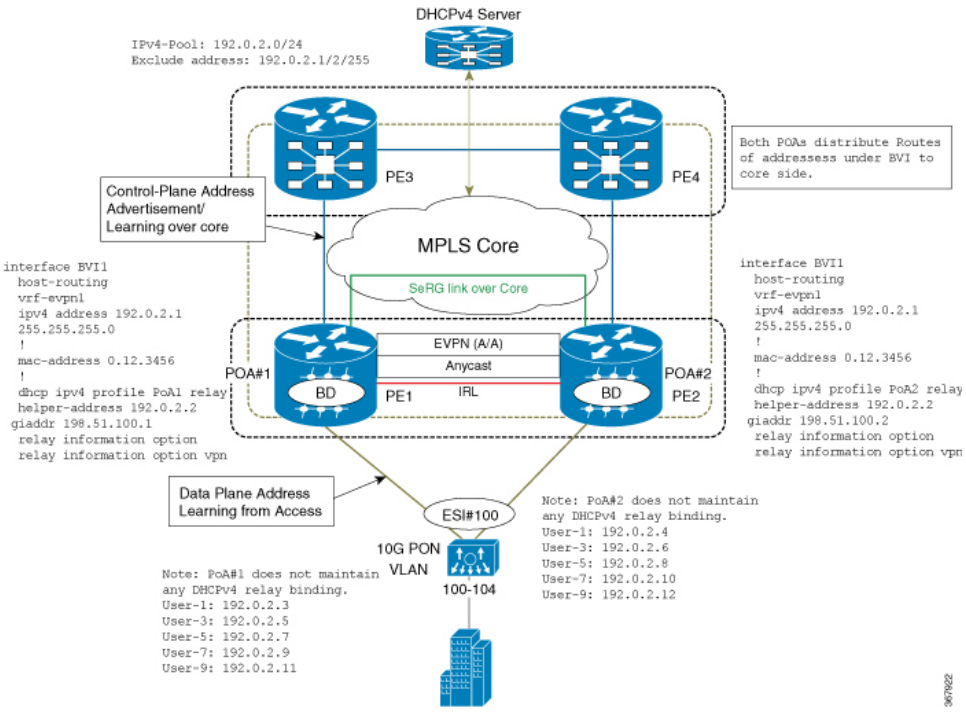


DHPCv4 Relay Handling for EVPN and DHCPv4 Server in Different VRF

DHCPv4 relay over EVPN IRB and DHCPv4 servers reside in different VRFs or DHCPv4 server has an unique GIADDR across ToRs which is different from the anycast IRB address. Else, it is difficult for DHCPv4 server to perform address allocation for end user of not having link selection or subnet selection. To ensure DHCPv4 server to provide address allocation from pool of subnet of related anycast IRB address of evpn, there is a way that ToRs of DHCPv4 relay agent intimate Virtual-Subnet-Selection (link-selection, server-id, vrf-id) by including Relay-Agent-Information (Option-82) in DHCPv4 relayed Discover and Request packets towards DHCPv4 Server.

In this topology, the 10G PON distributes equally the DHCP broadcast towards respective point of attachment (PoA) #1, #2, and packets are relayed to external DHCPv4 server.

Figure 63: DHPCv4 Relay Handling for EVPN and DHCPv4 Server in Different VRF



Configure DHCPv4 Relay on IRB

Perfrom these tasks to configure DHCPv4 Relay on IRB.

Configuration Example

```
/* PE1 configuration */

Router# configure
Router(config)# interface BVI1
Router(config-if)# host-routing
Router(config-if)# vrf-evpn1
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# exit
Router(config)# mac-address 0.12.3456
!
Router# configure
Router(config)# dhcp ipv4
Router(config-dhcpv4)# profile PoA1 relay
Router(config-dhcpv4-relay-profile)# helper-address 192.0.2.2 giaddr 198.51.100.1
Router(config-dhcpv4-relay-profile)# relay information option vpn
Router(config-dhcpv4-relay-profile)# relay information option vpn-mode rfc
Router(config-dhcpv4-relay-profile)# commit

/* PE2 configuration */

Router# configure
Router(config)# interface BVI1
```

```
Router(config-if)# host-routing
Router(config-if)# vrf-evpn1
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# exit
Router(config)# mac-address 0.12.3456
!
Router# configure
Router(config)# dhcp ipv4
Router(config-dhcpv4)# profile PoA2 relay
Router(config-dhcpv4-relay-profile)# helper-address 192.0.2.2 giaddr 198.51.100.2
Router(config-dhcpv4-relay-profile)# relay information option vpn
Router(config-dhcpv4-relay-profile)# relay information option vpn-mode rfc
Router(config-dhcpv4-relay-profile)# commit
```

The following example shows a configuration of DHCPv4 relay agent to include Relay-Agent Information with Remote-ID and Circuit-ID. The Remote-ID is configured under DHCPv4-Relay-Profile, which is associated under BVI interface. DHCPv4 is configured with L2Transport ACs with Circuit-ID.

```
Dhcp ipv4
 Profile RELAY relay
  Relay information option remote-id format-type asci cisco
  Relay information policy encapsulate
 !

interface BE1.100 relay information option circuit-id format-type hex cisco
!
 interface bvi relay RELAY
!
```

Running Configuration

This section shows DHCPv4 relay on IRB running configuration.

```
/* PE1 Configuration */
interface BVI1
 host-routing
 vrf-evpn1
 ipv4 address 192.0.2.1 255.255.255.0
 !
 mac-address 0.12.3456
!
 dhcp ipv4 profile PoA1 relay
 helper-address 192.0.2.2 giaddr 198.51.100.1
 relay information option
 relay information option vpn-mode rfc

/* PE2 Configuration */
interface BVI1
 host-routing
 vrf-evpn1
 ipv4 address 192.0.2.1 255.255.255.0
 !
 mac-address 0.12.3456
!
 dhcp ipv4 profile PoA2 relay
 helper-address 192.0.2.2 giaddr 198.51.100.2
 relay information option
 relay information option vpn-mode rfc
```

Verification

Verify DHCPv4 Relay on IRB configuration.

```
/* Verify DHCPv4 relay statistics
Router# show dhcp vrf default ipv4 relay statistics

DHCP IPv4 Relay Statistics for VRF default:

     TYPE          |    RECEIVE    |    TRANSMIT   |     DROP      |
-------------------------------------------------------------------
 DISCOVER          |          2000 |          2000 |             0 |
 OFFER             |          2000 |          2000 |             0 |
 REQUEST           |          5500 |          5500 |             0 |
 DECLINE           |             0 |             0 |             0 |
 ACK               |          5500 |          5500 |             0 |
 NAK               |             0 |             0 |             0 |
 RELEASE           |           500 |           500 |             0 |
 INFORM            |             0 |             0 |             0 |
 LEASEQUERY        |             0 |             0 |             0 |
 LEASEUNASSIGNED   |             0 |             0 |             0 |
 LEASEUNKNOWN      |             0 |             0 |             0 |
 LEASEACTIVE       |             0 |             0 |             0 |
 BOOTP-REQUEST     |             0 |             0 |             0 |
 BOOTP-REPLY       |             0 |             0 |             0 |
 BOOTP-INVALID     |             0 |             0 |             0 |


/* Verify DHCPv4 relay profile details */
Router# show dhcp ivp4 profile name PoA1 relay

Profile: PoA1 relay
Helper Addresses:
        192.0.2.2, vrf default, giaddr 198.51.100.1
Remote-Id Format  : [ascii | hex]
Remote-Id value   : cisco
Information Option: Enabled
Information Option Allow Untrusted: Enabled
Information Option VPN: Enabled
Information Option VPN Mode: RFC
Information Option Policy: Replace
```

Related Topics

- DHCPv4 Relay on IRB, on page 431

Associated Commands

- show dhcp vrf default ipv4 relay statistics
- show dhcp ivp4 profile name

DHCPv4 Relay Synchronization for All-Active Multihoming

DHCPv4 Relay Synchronization for All-active Multihoming feature enables a transitory entity between the end user and DHCPv4 server and does not create any DHCPv4 binding. This feature supports the equal distribution of DHCP control-plane packets among end users across Point of Attachments (PoAs). All DHCP

control packets for single users exist on the same DHCPv4 relay (PoA) so that end users can lease IP address allocation without any intervention and delay.

Multiprotocol extension BGP session is established between PEs to edge routers over MPLS-SR so that the learned MAC-IP information is sent over BGP to the edge router. MP-BGP advertises the learned MAC-IP information using route type-2 for a given Ethernet Segment Identifier (ESI) and Ethernet tag. The edge router has the capability of redistributing the routes to other PEs that are learnt from PE1 or PE2, and vice-versa. This mechanism ensures that the MAC-IP routes are distributed to the edge router so that individual PEs have complete MAC-IP routing information.

This feature ensures forwarding of bidirectional traffic. For high availability, during node (PoA#1 or PoA#2) failures, access interface failures, or core link failures, the other PoA forwards data traffic.

DHCPv6 Relay IAPD on IRB

The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Identity Association for Prefix Delegation (IAPD) on IRB feature allows the user to manage link, subnet, and site addressing changes. This feature automates the process of assigning prefixes to a customer for use within their network. The prefix delegation occurs between a provider edge (PE) device and customer edge (CE) device using the DHCPv6 prefix delegation option. After the delegated prefixes are assigned to a user, the user may further subnet and assign prefixes to the links in the network.

DHCPv6 relay transmits all request packets that comes over access interface towards external DHCPv6 server to request IAPD (::/64 or ::/48) allocation for the end user. DHCPv6 relay also receives response packets from DHCPv6 server and forwards the packets towards the end users over access interface. DHCPv6 relay acts as stateful for the end users by maintaining DHCPv6 PD binding and respective route entry for the allocated IAPD. DHCPv6 relay supports Internet Assigned Numbers Authority (IANA) and Identity Association for Prefix Delegation (IAPD) address allocation for the end-user. The IAPD prefix is based on prefix-pool that is configured on DHCPv6 server.

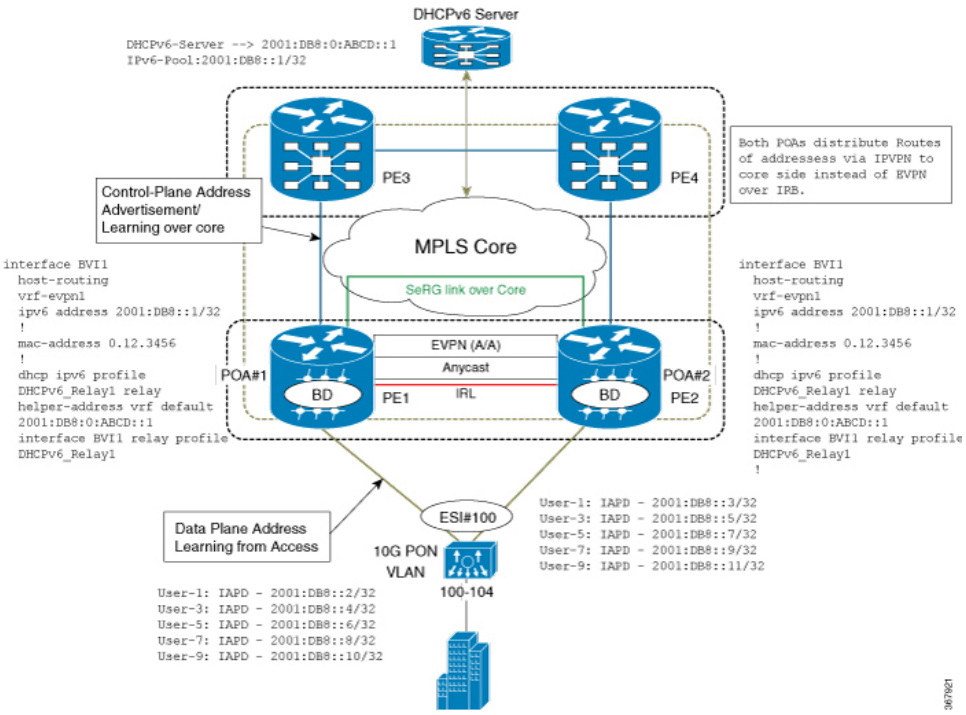
For DHCPv6 relay, access (BVI) interface is tied up with relay profile. Whenever ToRs relay the DHCPv6 packets that are received from client to DHCPv6 server, ToR discovers the best source IP address for a given defined VRF of DHCPv6 server IP address. ToRs maintain unique source IP address for each VRF to reach out DHCPv6 server. DHCPv6 relay has unique IPv4 source IP address defined under loopback interfaces for the defined VRFs of DHCPv6 helper-addresses and routable through MPLS core network.

Anycast IP address configured on the BVI interface acts as a default gateway for end users and address allocation occurs on the same subnet. ToRs maintain unique source IP address to relay DHCPv6 packets towards DHCPv6 server over IPVPN of MPLS core network. The same ToRs receive response packets from external DHCPv6 server. Unique source address on each ToR under DHCPv6 relay is required for DHCPv6 process to maintain the context of packet received over access interface and relayed packet. This mechanism helps to send reply response to end users over BVI interface.

DHPCv6 relay Handling for EVPN and DHCPv6 Server in Default VRF

DHCPv6 relay over EVPN IRB and DHCPv6 servers resides in the same default VRFs. The DHCPv6 relay profiles are associated with helper-addresses of DHCPv6 address under default VRFs. The PEs do not include Relay-Information option in DHPCv6-Relayed packets unlike DHCPv4.

Figure 64: DHPCv6 relay Handling for EVPN and DHCPv6 Server in Default VRF



Configure DHCPv6 Relay IAPD on IRB

Perfrom these tasks to configure DHCPv6 Relay IAPD on IRB.

Configuration Example

```
/* PE1 configuration */

Router# configure
Router(config)# interface BVI1
Router(config-if)# host-routing
Router(config-if)# vrf-evpn1
Router(config-if)# ipv6 address 2001:DB8::1/32
Router(config-if)# exit
Router(config)# mac-address 0.12.3456
!
Router# configure
Router(config)# dhcp ipv6
Router(config-dhcpv6)# profile DHCPv6_Relay1 relay
Router(config-dhcpv6-relay-profile)# helper-address vrf default 2001: DB8:0:ABCD::1
Router(config-dhcpv6-relay-profile)# interface BVI1 relay profile DHCPv6_Relay
Router(config-dhcpv6-relay-profile)# commit

/* PE2 configuration */

Router# configure
Router(config)# interface BVI1
Router(config-if)# host-routing
```

```
Router(config-if)# vrf-evpn1
Router(config-if)# ipv6 address 2001:DB8::1/32
Router(config-if)# exit
Router(config)# mac-address 0.12.3456
!
Router# configure
Router(config)# dhcp ipv6
Router(config-dhcpv6)# profile DHCPv6_Relay1 relay
Router(config-dhcpv6-relay-profile)# helper-address vrf default 2001: DB8:0:ABCD::1
Router(config-dhcpv6-relay-profile)# interface BVI1 relay profile DHCPv6_Relay
Router(config-dhcpv6-relay-profile)# commit
```

Running Configuration

This section shows DHCPv6 Relay IAPD on IRB running configuration.

```
/* PE1 Configuration */
interface BVI1
 host-routing
 vrf-evpn1
 ipv6 address 2001:DB8::1/32
 !
 mac-address 0.12.3456
!
 dhcp ipv6 profile DHCPv6_Relay1 relay
 helper-address vrf default 2001: DB8:0:ABCD::1
 interface BVI1 relay profile DHCPv6_Relay1
!

/* PE2 Configuration */interface BVI1
 host-routing
 vrf-evpn1
 ipv6 address 2001:DB8::1/32
 !
 mac-address 0.12.3456
!
 dhcp ipv6 profile DHCPv6_Relay1 relay
 helper-address vrf default 2001: DB8:0:ABCD::1
 interface BVI1 relay profile DHCPv6_Relay1
!
```

Verification

Verify DHCPv46 Relay IAPD on IRB configuration.

```
/* Verify DHCPv6 relay statistics
Router# show dhcp vrf default ipv6 relay statistics

DHCP IPv6 Relay Statistics for VRF default:

     TYPE          |    RECEIVE    |    TRANSMIT   |     DROP      |
-----------------------------------------------------------------------
 DISCOVER          |          2000  |          2000  |             0  |
 OFFER             |          2000  |          2000  |             0  |
 REQUEST           |          5500  |          5500  |             0  |
 DECLINE           |             0  |             0  |             0  |
 ACK               |          5500  |          5500  |             0  |
 NAK               |             0  |             0  |             0  |
 RELEASE           |           500  |           500  |             0  |
 INFORM            |             0  |             0  |             0  |
 LEASEQUERY        |             0  |             0  |             0  |
```

```
LEASEUNASSIGNED   |            0   |            0   |            0   |
LEASEUNKNOWN      |            0   |            0   |            0   |
LEASEACTIVE       |            0   |            0   |            0   |
BOOTP-REQUEST     |            0   |            0   |            0   |
BOOTP-REPLY       |            0   |            0   |            0   |
BOOTP-INVALID     |            0   |            0   |            0   |
```

Related Topics

- DHCPv6 Relay IAPD on IRB, on page 439

Associated Commands

- show dhcp ipv6 relay statistics vrf default

DHCPv6 PD Synchronization for All-Active Multihoming using Session Redundancy

DHCPv6 PD Synchronization for All-Active Multihoming using Session Redundancy feature provides load balancing for both control and data packets. This feature helps in efficient utilization of devices with respect to throughput (line rate) and processing power.

Prior to this release, Session Redundancy (SeRG) mechanism supported active-standby to address access failure, core failure, and node or chassis failures. In all these cases, one active PoA is responsible to create sessions and synchronize binding information using SeRG across the PoA. This mechanism did not serve the purpose of EVPN all-active multihoming as PoAs are in primary-secondary mode for a given access-link in SeRG group. This restricts only one node that acts as primary to process control packets, create bindings, and forward data path.

With DHCPv6 PD Synchronization for All-active Multihoming feature using SeRG group configuration, you can define both POAs to be active unlike in primary-secondary mode. Also, there is no need to exchange or negotiate the roles of respective PoAs.

SeRG does not distribute IAPD prefix routes over BGP in any of the route types. The routed BVI interface is configured with DHCPv6 relay to provide PD allocation for the end user.

Each individual multihoming peer SeRG role is `ACTIVE` only. SeRG does not support any roles other than `NONE` and `ACTIVE`. Define interface-list under SeRG as BVI interface, typically use one or more BVI interfaces. However, it is not recommended to define L2 transport ACs under SeRG interface list because the L2 transport ACs are defined under L2VPN BD, and SeRG-client DHCPv6 is unaware of these AC information.

In SeRG active-active mode, IPv6-ND synchronization is supressed across POAs.

Restrictions

- SeRG does not support core link failures.
- SeRG does not support core and access tracking mechanism.
- Ensure that there are no bindings while configuring `ACTIVE-ACTIVE` mode.
- Ensure that you have the same configuration on all PoAs. The Bundle-Ether L2transport ACs configuration has to be same on both the sides along with BD and BVI configuration.

- **clear session-redundancy** command is not supported in any mode to avoid system inconsistency.
- In SeRG active-active mode, ensure that both PoAs are reachable over core links always. It is recommended to configure EVPN Core Isolation feature, which maps core links to access link. This mechanism ensures to eliminate respective access links whenever core links are down.
- SeRG is supported on the NCS 5500 and NCS 5700 platforms, excluding NCS-57C1, NCS-57B1, and NCS-57D2.

Configure DHCPv6 PD Synchronization

Perfrom these tasks to configure DHCPv6 PD synchronization using SeRG.

Configuration Example

```
/* PoA1 configuration */
Router# configure
Router(config)# session redundancy
Router(config-session-red)# source-interface Loopback0
Router(config-session-red)# group 1
Router(config-session-red-group)# peer 192.0.2.1
Router(config-session-red-group)# mode active-active
Router(config-session-red-group)# interface-list
Router(config-session-red-group-inft)# interface BVI1 id 1
Router(config-session-red-group-intf)# commit

/* PoA2 configuration */
Router# configure
Router(config)# session redundancy
Router(config-session-red)# source-interface Loopback0
Router(config-session-red)# group 1
Router(config-session-red-group)# peer 198.51.100.1
Router(config-session-red-group)# mode active-active
Router(config-session-red-group)# interface-list
Router(config-session-red-group-intf)# interface BVI1 id 1
Router(config-session-red-group-intf)# commit
```

Running Configuration

This section shows DHCPv6 PD synchronization running configuration.

```
/* PoA1 Configuration */
session-redundancy
source-interface Loopback0
group 1
  peer 192.0.2.1
  mode active-active
  interface-list
   interface BVI1 id 1
  !
!
!
/* PoA2 Configuration */
session-redundancy
source-interface Loopback0
group 1
```

```
  peer 198.51.100.1
  mode active-active
  interface-list
   interface BVI1 id 1
  !
!

!
```

Verification

Verify DHCPv6 PD synchronization configuration.

```
/* Verify the session redundancy group */

Router# show session-redundancy group
Wed Nov 28 16:00:36.559 UTC
Session Redundancy Agent Group Summary
Flags    : E - Enabled, D - Disabled, M - Preferred Master, S - Preferred Slave
           H - Hot Mode, W - Warm Mode, T - Object Tracking Enabled
P/S      : Peer Status
           I - Initialize, Y - Retry, X - Cleanup, T - Connecting
           L - Listening, R- Registered, C - Connected, E - Established
I/F-P Count: Interface or Pool Count
SS Count : Session Count
------------------------------------------------------------------------------------------------------------

   Node Name    | Group ID | Role | Flags  |         Peer Address        | P/S | I/F-P Count
 |  SS Count  | Sync Pending
------------------------------------------------------------------------------------------------------------
0/RP0/CPU0             1  Active   E-H-   120.1.1.1                      E            1
         1               0
0/RP0/CPU0             2  Active   E-H-   120.1.1.1                      E            1
         0               0
0/RP0/CPU0             3  Active   E-H-   120.1.1.1                      E            1
         0               0
0/RP0/CPU0             4  Active   E-H-   120.1.1.1                      E            1
         0               0
0/RP0/CPU0             5  Active   E-H-   120.1.1.1                      E            1
         0               0
------------------------------------------------------------------------------------------------------------
Session Summary Count(Master/Slave/Active/Total): 0/0/1/1

/* Verify IPv6 relay binding */

Router# show dhcp ipv6 relay binding
Summary:
Total number of clients: 1

IPv6 Prefix: 60:1:1:1::/64 (BVI1)
    Client DUID: 000100015bfeb921001094000000
    IAID: 0x0
    VRF: default
    Lifetime: 120 secs (00:02:00)
    Expiration: 91 secs (00:01:31)
    L2Intf AC: Bundle-Ether1.1
    SERG State: SERG-ACTIVE
    SERG Intf State: SERG-ACTIVE
```

Related Topics

- DHCPv6 PD Synchronization for All-Active Multihoming using Session Redundancy , on page 442

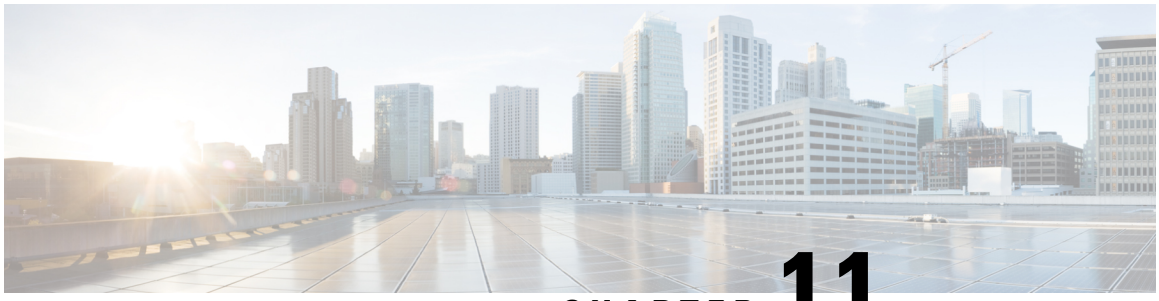
Associated Commands

- show session-redundancy group
- show dhcp ipv6 relay binding

IAPD Route Distribution and Withdrawal in DHCPv6 Relay

If there is an EVPN Multi-Homing Active-Active scenario, DHCPv6 relay agent is supported over L2VPN bridge domain associated with Attachment Circuits (ACs) and BVI interface with allocation of Identity Association for Prefix Delegation (IAPD) routes. Also, DHCPv6 relay agent performs route distribution using iBGP over the MPLS core network. During core-to-subscriber traffic, few ACs can be down, but BVI is still up because not all ACs are down. This scenario can result in unreported traffic drop for subscribers in ACs that are down. The cause being the IAPD routes that are still intact with the MPLS core network though the ACs are down.

To prevent unreported traffic drop, the DHCPv6 relay agent is enabled to perform IAPD route withdrawal from the MPLS core network over iBGP for sessions. The route withdrawals occur whenever the L2VPN bridge domain ACs are down. Also, whenever the ACs return to the up state, the DHCPv6 relay agent can distribute IAPD routes to the MPLS core network over iBGP.



CHAPTER 11

EVPN Virtual Private Wire Service (VPWS)

The EVPN-VPWS is a BGP control plane solution for point-to-point services. It implements the signaling and encapsulation techniques for establishing an EVPN instance between a pair of PEs. It has the ability to forward traffic from one network to another without MAC lookup. The use of EVPN for VPWS eliminates the need for signaling single-segment and multi-segment PWs for point-to-point Ethernet services. The EVPN-VPWS technology works on IP and MPLS core; IP core to support BGP and MPLS core for switching packets between the endpoints.

EVPN-VPWS support both single-homing and multi-homing.

- Supported Modes for EVPN-VPWS, on page 447
- EVPN-VPWS Single Homed, on page 448
- EVPN-VPWS Multi-Homed, on page 450
- Flow Label Support for EVPN VPWS, on page 453

Supported Modes for EVPN-VPWS

EVPN-VPWS supports the following modes:

- Single-homed - Enables you to connect a customer edge (CE) device to one provider edge (PE) device.
- Multi-homed - Enables you to connect a customer edge (CE) device to more than one provider edge (PE) device. Multihoming ensures redundant connectivity. The redundant PE device ensures that there is no traffic disruption when there is a network failure. Following are the types of multihoming:
  - Single-Active - In single-active mode only a single PE among a group of PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.
  - All-Active - In all-active mode all the PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.

Benefits of EVPN-VPWS

The following are the benefits of EVPN-VPWS:

- Scalability is achieved without signaling pseudowires.
- Ease of provisioning

- Pseudowires (PWs) are not used.
- Leverages BGP best path selection (optimal forwarding).

Prerequisites for EVPN-VPWS

- Ensure BGP is configured for EVPN SAFI.
- BGP session between PEs with 'address-family l2vpn evpn' to exchange EVPN routes.

Restrictions for EVPN-VPWS

- The VPN ID is unique per router.
- When specifying a list of route targets, they must be unique per PE (per BGP address-family).
- On versions earlier than IOS XR release 6.7.1, MTU is not signaled and the MTU mismatch is ignored with no interoperability issues.
- On versions starting from IOS XR release 6.7.1, L3 MTU is advertised by default and the MTU mismatch is enforced by default.
- On versions earlier than IOS XR release 7.0.x, MTU is not signaled and the MTU mismatch is ignored with no interoperability issues.

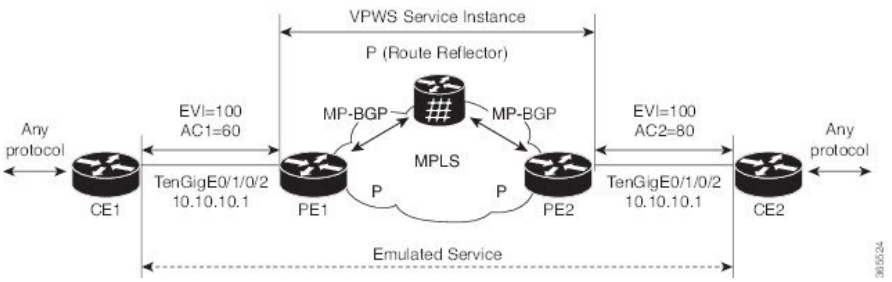
  On versions later than IOS XR release 7.0.x, L3 MTU is advertised by default and the MTU mismatch is enforced by default. But this results in interoperability issues with IOS XR release 7.3.2, if transmit-l2-mtu is configured since L3 and L2 MTUs do not match. You can configure **transmit-mtu-zero** and **ignore-mtu-mismatch** commands to avoid this situation.

  On versions later than IOS XR release 7.3.2, MTU of 0 is advertised by default, and the MTU mismatch is ignored by default. L2 MTU can be advertised using the **transmit-l2-mtu** command, and MTU mismatch can be enforced with **enforce-mtu-mismatch** command.

EVPN-VPWS Single Homed

The EVPN-VPWS single homed solution requires per EVI Ethernet Auto Discovery route. EVPN defines a new BGP Network Layer Reachability Information (NLRI) used to carry all EVPN routes. BGP Capabilities Advertisement used to ensure that two speakers support EVPN NLRI (AFI 25, SAFI 70) as per RFC 4760.

The architecture for EVPN VPWS is that the PEs run Multi-Protocol BGP in control-plane. The following image describes the EVPN-VPWS configuration:



- The VPWS service on PE1 requires the following three elements to be specified at configuration time:
  - The VPN ID (EVI)
  - The local AC identifier (AC1) that identifies the local end of the emulated service.
  - The remote AC identifier (AC2) that identifies the remote end of the emulated service.

  PE1 allocates a MPLS label per local AC for reachability.
- The VPWS service on PE2 is set in the same manner as PE1. The three same elements are required and the service configuration must be symmetric.

  PE2 allocates a MPLS label per local AC for reachability.
- PE1 advertise a single EVPN per EVI Ethernet AD route for each local endpoint (AC) to remote PEs with the associated MPLS label.

  PE2 performs the same task.
- On reception of EVPN per EVI EAD route from PE2, PE1 adds the entry to its local L2 RIB. PE1 knows the path list to reach AC2, for example, next hop is PE2 IP address and MPLS label for AC2.

  PE2 performs the same task.

Configure EVPN-VPWS Single Homed

This section describes how you can configure single-homed EVPN-VPWS feature.

```
/* Configure PE1 */

Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.10.10.1
Router(config-bgp-nbr))# address-family l2vpn evpn
Router(config-bgp-nbr-af)# commit
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/1/0/2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 100 target 12 source 10
Router(config-l2vpn-xc-p2p)# commit

Router(config-l2vpn-xc-p2p)# exit

/* Configure PE2 */

Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.10.10.1
Router(config-bgp-nbr))# address-family l2vpn evpn
Router(config-bgp-nbr-af)# commit
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
```

```
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/1/0/2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 100 target 10 source 12
Router(config-l2vpn-xc-p2p)# commit

Router(config-l2vpn-xc-p2p)# exit
```

Running Configuration

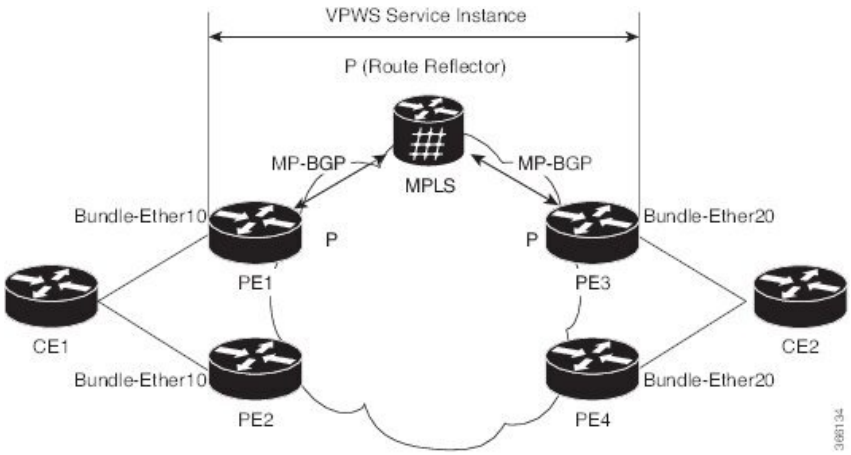
```
/* On PE1 */

configure
router bgp 100
 address-family l2vpn evpn
 neighbor 10.10.10.1
  address-family l2vpn evpn
!

configure
l2vpn
 xconnect group evpn-vpws
  p2p evpn1
   interface TenGigE0/1/0/2
   neighbor evpn evi 100 target 12 source 10
!

/* On PE2 */

configure
router bgp 100
 address-family l2vpn evpn
 neighbor 10.10.10.1
  address-family l2vpn evpn
!

configure
l2vpn
 xconnect group evpn-vpws
  p2p evpn1
   interface TenGigE0/1/0/2
   neighbor evpn evi 100 target 10 source 12
!
```

EVPN-VPWS Multi-Homed

The EVPN VPWS feature supports all-active multihoming capability that enables you to connect a customer edge device to two or more provider edge (PE) devices to provide load balancing and redundant connectivity. The load balancing is done using equal-cost multipath (ECMP).

When a CE device is multi-homed to two or more PEs and when all PEs can forward traffic to and from the multi-homed device for the VLAN, then such multihoming is referred to as all-active multihoming.

Figure 65: EVPN VPWS Multi-Homed



Consider the topology in which CE1 is multi-homed to PE1 and PE2; CE2 is multi-homed to PE3 and PE4. PE1 and PE2 will advertise an EAD per EVI route per AC to remote PEs which is PE3 and PE4, with the associated MPLS label. The ES-EAD route is advertised per ES (main interface), and it will not have a label. Similarly, PE3 and PE4 advertise an EAD per EVI route per AC to remote PEs, which is PE1 and PE2, with the associated MPLS label.

Consider a traffic flow from CE1 to CE2. Traffic is sent to either PE1 or PE2. The selection of path is dependent on the CE implementation for forwarding over a LAG. Traffic is encapsulated at each PE and forwarded to the remote PEs (PE 3 and PE4) through MPLS core. Selection of the destination PE is established by flow-based load balancing. PE3 and PE4 send the traffic to CE2. The selection of path from PE3 or PE4 to CE2 is established by flow-based load balancing.

If there is a failure and when the link from CE1 to PE1 goes down, the PE1 withdraws the ES-EAD route; sends a signal to the remote PEs to switch all the VPWS service instances associated with this multi-homed ES to backup PE, which is PE2.

Configure EVPN-VPWS All-Active Multi-Homed

This section describes how to configure all-active multi-homed EVPN-VPWS feature.

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether10.2
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)# exit

Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
Router(config-evpn-ac-es)# commit


/* Configure PE2 */
```

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether10.2
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)# exit

Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
Router(config-evpn-ac-es)# commit


/* Configure PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether20.1
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)# exit

Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit


/* Configure PE4 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether20.1
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
/* On PE1 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
```

```
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00

!


/* On PE2 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00

!

/* On PE3 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
      identifier type 0 00.01.00.ac.ce.55.00.14.00

!

/* On PE4 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
      identifier type 0 00.01.00.ac.ce.55.00.14.00

!
```

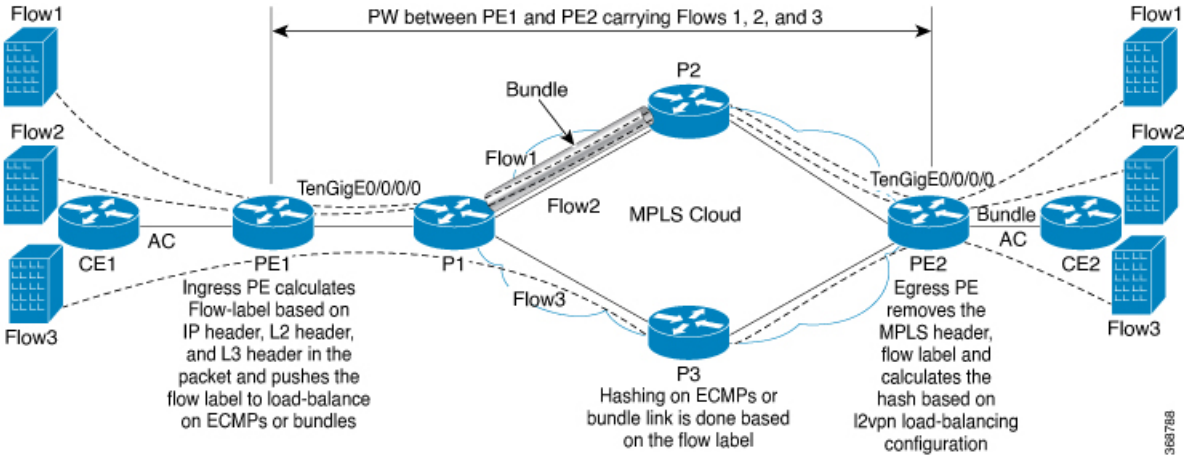
Flow Label Support for EVPN VPWS

The Flow Label support for EVPN VPWS feature enables provider (P) routers to use the flow-based load balancing to forward traffic between the provider edge (PE) devices. This feature uses Flow-Aware Transport (FAT) of pseudowires (PW) over an MPLS packet switched network for load-balancing traffic across BGP-based signaled pseudowires for Ethernet VPN (EVPN) Virtual Private Wire Service (VPWS).

FAT PWs provide the capability to identify individual flows within a PW and provide routers the ability to use these flows to load-balance the traffic. FAT PWs are used to load balance the traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on indivisible packet flows entering an imposition PE. This flow label is inserted as the lower most label in the packet. P routers use the flow label for load balancing to provide better traffic distribution across ECMP paths or link-bundled paths in the core. A flow is identified either by the source and destination IP address of the traffic, or the source and destination MAC address of the traffic.

The following figure shows a FAT PW with two flows distributing over ECMPs and bundle links.

Figure 66: FAT PW with Two Flows Distributing over ECMPs and Bundle



An extra label is added to the stack, called the flow label, which is generated for each unique incoming flow on the PE. A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from source and destination MAC addresses, and source and destination IP addresses. The flow label contains the end of label stack (EOS) bit set. The flow label is inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The FAT PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

Core routers perform load balancing using the flow-label in the FAT PW with other information like MAC address and IP address. The flow-label adds greater entropy to improve traffic load balancing. Therefore, it is possible to distribute flows over ECMPs and link bundles.

In this topology, the imposition router, PE1, adds a flow label in the traffic. The disposition router, PE2, allows mixed types of traffic of which some have flow label, others do not. The P router uses flow label to load balance the traffic between the PEs. PE2 ignores the flow label in traffic, and uses one EVPN label for all unicast traffic.

Restrictions

To configure flow label for EVPN VPWS, the following restrictions are applicable:

- This feature is not supported for EVPN Point-to-Multipoint (P2MP) of VPLS and Ethernet LAN (E-LAN) service.
- This feature is supported only for EVPN VPWS single homing. AC bundle interfaces must be configured with ESI-0 only.
- This feature is not supported for EVPN flexible cross-connect service.
- This feature is not supported for EVPN VPWS multihoming.

Configure Flow Label for EVPN VPWS

Configuration Example

Perform this task to configure flow label for EVPN VPWS on both PE1 and PE2.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/0
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 2 source 1
Router(config-l2vpn-xc-p2p)# exit
!
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 1

Router(config-evpn-instance)# load-balancing
Router(config-evpn-instance-lb)# flow-label static
Router(config-evpn-instance-lb)# commit
```

Running Configuration

This section shows the running configuration of flow label for EVPN VPWS.

```
l2vpn
 xconnect group evpn-vpws
  p2p evpn1
   interface TenGigE0/0/0/0
   neighbor evpn evi 1 target 2 source 1
   !
  !
evpn
 evi 1
    load-balancing
   flow-label static
  !
!
```

Verification

Verify EVPN VPWS flow label configuration.

```
Router# show l2vpn xconnect detail
Group evpn-vpws, XC evpn1, state is up; Interworking none
  AC: TenGigE0/0/0/0, state is up
    Type Ethernet
    MTU 1500; XC ID 0x1; interworking none
    Statistics:
      packets: received 21757444, sent 0
      bytes: received 18226521128, sent 0
  EVPN: neighbor 100.100.100.2, PW ID: evi 1, ac-id 2, state is up ( established )
    XC ID 0xc0000001
    Encapsulation MPLS
   Encap type Ethernet, control word disabled
    Sequencing not set
    LSP : Up
    Flow Label flags configured (Tx=1,Rx=1) statically
```

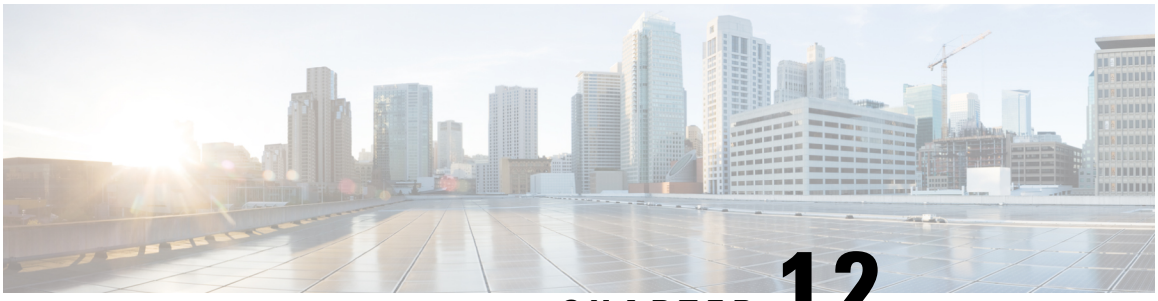
```
  EVPN         Local                          Remote
  ------------ ------------------------------ ------------------------------
  Label        64002                          64002
  MTU          1500                           1500
  Control word disabled                       disabled
  AC ID        1                              2
  EVPN type    Ethernet                       Ethernet

  ------------ ------------------------------ ------------------------------
Create time: 30/10/2018 03:04:16 (00:00:40 ago)
Last time status changed: 30/10/2018 03:04:16 (00:00:40 ago)
Statistics:
  packets: received 0, sent 21757444
  bytes: received 0, sent 18226521128
```

Related Topics

- Flow Label Support for EVPN VPWS, on page 453

Associated Commands

- show evpn evi



CHAPTER 12

L2VPN Preferred path

All L2VPN services such as VPLS, VPWS, and so on must use L2VPN preferred-path while using TE (SR-TE, and RSPV-TE) services as transport. Preferred-path CLI should be set to ensure that the L2VPN traffic is tunnel bound. This will bring up or tear down the L2VPN session based on the tunnel status.

The use of auto-route announce is not recommended as it impacts the way L2VPN tracks the nexthop reachability and causes the L2VPN to be independent of tunnel status.

- L2VPN Services over Segment Routing for Traffic Engineering Policy, on page 457
- EVPN VPWS Preferred Path over SR-TE Policy, on page 458
- L2VPN VPWS Preferred Path over SR-TE Policy, on page 471
- EVPN VPWS On-Demand Next Hop with SR-TE, on page 484
- Overview of Segment Routing , on page 498
- How Segment Routing Works , on page 499
- Segment Routing Global Block , on page 500

L2VPN Services over Segment Routing for Traffic Engineering Policy

Segment Routing (SR) is a flexible and scalable way of performing source routing. The source device selects a path and encodes it in the packet header as an ordered list of segments. Segments are identifiers for any type of instruction.

Segment routing for traffic engineering (SR-TE) takes place through a tunnel between a source and destination pair. SR-TE uses the concept of source routing, where the source calculates the path and encodes it in the packet header as a segment. In SR-TE preferred path, each segment is an end-to-end path from the source to the destination, and instructs the routers in the provider core network to follow the specified path instead of the shortest path calculated by the IGP. The destination is unaware of the presence of the tunnel.

The user can achieve better resilience and convergence for the network traffic, by transporting MPLS L2VPN services using segment routing, instead of MPLS LDP. Segment routing can be directly applied to the MPLS architecture without changing the forwarding plane. In a segment-routing network that uses the MPLS data plane, LDP or other signaling protocol is not required; instead label distribution is performed by IGP. Removing protocols from the network simplifies its operation and makes it more robust and stable by eliminating the need for protocol interaction. Segment routing utilizes the network bandwidth more effectively than traditional MPLS networks and offers lower latency.

Preferred tunnel path functionality allows you map pseudowires to specific traffic-engineering tunnel paths. Attachment circuits are cross-connected to specific SR traffic engineering tunnel interfaces instead of remote PE router IP addresses reachable using IGP or LDP. Using preferred tunnel path, the traffic engineering tunnel transports traffic between the source and destination PE routers. A path is selected for an SR Policy when the path is valid and its preference is the best (highest value) among all the candidate paths of the SR Policy.

The following L2VPN services are supported over SR-TE policy:

EVPN VPWS Preferred Path over SR-TE Policy

EVPN VPWS Preferred Path over SR-TE Policy feature allows you to set the preferred path between the two end-points for EVPN VPWS pseudowire (PW) using SR-TE policy. SR policy allows you to choose the path on a per EVPN instance (EVI) basis. This feature is supported on bundle attachment circuit (AC) and physical AC.

Restrictions

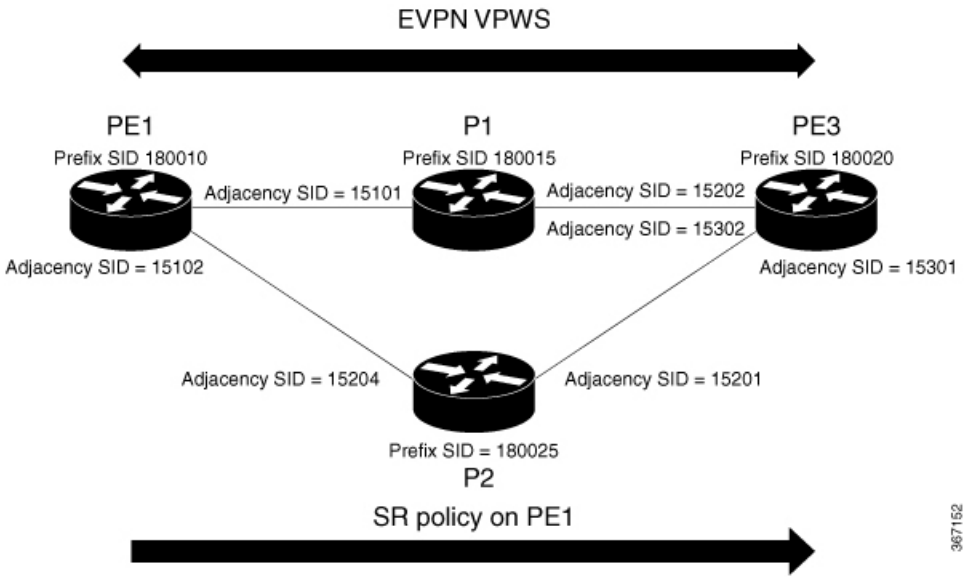
- If EVPN VPWS with On Demand Next Hop (ODN) is configured, and EVPN VPWS with preferred path is also configured for the same PW, then the preferred-path will take precedence.
- EVPN VPWS SR policy is not supported on EVPN VPWS dual homing.
- EVPN validates if the route is for a single home next hop, otherwise it issues an error message about a dangling SR TE policy, and continue to set up EVPN-VPWS without it. EVPN relies on ESI value being zero to determine if this is a single home or not. If the AC is a Bundle-Ether interface running LACP then you need to manually configure the ESI value to zero to overwrite the auto-sense ESI as EVPN VPWS multihoming is not supported.

  To disable EVPN dual homing, configure bundle-Ether AC with ESI value set to zero.

```
evpn
interface Bundle-Ether12
  ethernet-segment
   identifier type 0 00.00.00.00.00.00.00.00.00
/* Or globally */
Evpn
 ethernet-segment type 1 auto-generation-disable
```

Topology

Figure 67: EVPN VPWS Preferred Path over SR-TE Policy



Consider a topology where PE1 and PE3 are the two EVPN VPWS PW end-points. Traffic is sent from PE1 to PE3 through SR in the core. Traffic from PE1 can be sent to PE3 either through P1 or P2 node. In this example, the EVPN VPWS preferred path over SR policy is configured to show the traffic flow from PE1 to PE3 using prefix-SID. Using adjacency-SID, you can steer traffic flow from PE1 to PE3 and specify whether it should pass through P1 or P2 node.

Configure EVPN VPWS Preferred Path over SR-TE Policy

You must complete these tasks to ensure the successful configuration of EVPN VPWS Preferred Path over SR-TE Policy feature:

- Configure Prefix-SID on IGP — The following examples show how to configure prefix-SID in IS-IS.
- Configure Adjacency-SID on IGP — The following examples show how to configure Adjacency-SID in IS-IS.
- Configure segment-list
- Configure SR-TE policy
- Configure EVPN VPWS over SR-TE policy

Configure Prefix-SID in ISIS

Configure Prefix-SID on PE1, P1, P2, and PE3.

```
/* Configure Prefix-SID on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
```

```
Router(config-sr)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0031.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id 10.0.0.1
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 180010
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on P1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# exit
!
Router# configure
Router(config)# router isis core
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 49.0002.0330.2000.0021.00
Router(config-isis)# nsr
Router(config-isis)# nsf ietf
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide level 2
Router(config-isis-af)# mpls traffic-eng level-2-only
Router(config-isis-af)# mpls traffic-eng router-id loopback0
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# segment-routing prefix-sid-map advertise-local
Router(config-isis-af)# exit
!
Router(config-isis)# interface loopback 0
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-af)# prefix-sid index 180015
Router(config-isis-af)# commit
Router(config-isis-af)# exit

/* Configure Prefix-SID on P2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block  180000 200000
Router(config-sr)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0022.00
Route(config-isis)# nsr
```

```
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 180025
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on PE3 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.3030.0030.0035.00
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 180020
Route(config-isis-af)# commit
Route(config-isis-af)# exit
```

Configure Adjacency-SID in ISIS

Configure Adjacency-SID on PE1, P1, P2, and PE3.

```
/* Configure Adjacency-SID on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15101
Route(config-isis-if-af)# exit
```

```
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/1/6
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15102
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on P1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# metric 20
Route(config-isis-if-af)# adjacency-sid absolute 15200
Route(config-isis-if-af)# commit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/7
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15202
Route(config-isis-if-af)# commit
!
/* Configure Adjacency-SID on P2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/7
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# metric 20
Route(config-isis-if-af)# adjacency-sid absolute 15201
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/5
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# metric 20
```

```
Route(config-isis-if-af)# adjacency-sid absolute 15204
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on PE3 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/1
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15301
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/2
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15302
Route(config-isis-if-af)# commit
```

Configure Segment-list

```
/* Configure Segment-list on PE1 using prefix-SID */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# traffic-eng
Router(config-sr-te)# logging
Router(config-sr-te-log)# policy status
Router(config-sr-te-log)# exit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list name pref_sid_to_PE3
Router(config-sr-te-sl)# index 1 mpls label 180020 <------------------------using prefix-SID
Router(config-sr-te-sl)# exit

/* Configure Segment-list on PE1 using adjacency-SID */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block  15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# logging
Router(config-sr-te-log)# policy status
Router(config-sr-te-log)# exit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
```

```
Router(config-sr-te)# segment-list name pref_adj_sid_to_PE3
Router(config-sr-te-sl)# index 1 mpls label 15101 <-----------------------using adjacency-SID
Router(config-sr-te-sl)# index 2 mpls label 15202 <-----------------------using adjacency-SID
Router(config-sr-te-sl)# exit
```

Configure SR-TE Policy

```
/* Configure SR-TE Policy */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy pref_sid_to_PE3
Router(config-sr-te-policy)# color 9001 end-point ipv4 172.16.0.1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 10
Router(config-sr-te-pp-info)# explicit segment-list pref_sid_to_PE3
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy pref_adj_sid_to_PE3
Router(config-sr-te-policy)# color 9001 end-point ipv4 172.16.0.1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 200
Router(config-sr-te-pp-info)# explicit segment-list pref_adj_sid_to_PE3
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit

/* You can configure multiple preferences for an SR policy. Among the configured preferences,
 the largest number takes the highest precedence */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 1013
Router(config-sr-te-policy)# color 1013 end-point ipv4 2.2.2.2
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-pp-info)# explicit segment-list PE1-P1_BE121
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy-path)# preference 200
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-P1-t0016
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy-path)# preference 700 <----------largest number takes the
precendence
Router(config-sr-te-pp-info)# explicit segment-list PE1-P1
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit
```

Configure EVPN VPWS over SR-TE Policy

Note Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to display the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name pref_sid_to_PE3

SR-TE policy database
---------------------
Color: 9001, End-point: 172.16.0.1
 Name: srte_c_9001_ep_172.16.0.1

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class 1001
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_9001_ep_172.16.0.1 fallback
 disable
Router(config-l2vpn-pwc-mpls)# commit
Router(config-l2vpn-pwc-mpls)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p evpn_vpws_1001
Router(config-l2vpn-xc-p2p)# interface tengi0/1/0/1.1001
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1001 target 10001 source 20001
Router(config-l2vpn-xc-p2p-pw)# pw-class 1001
Router(config-l2vpn-xc-p2p-pw)# commit
Router(config-l2vpn-xc-p2p-pw)# exit



/* If Fallback Enable is configured, which is the default option, and if the SR-policy is
down, then EVPN VPWS will still continue to be UP using the regular IGP path, and not using
 the SR-policy */
show l2vpn xconnect detail
  EVPN: neighbor 172.16.0.1, PW ID: evi 1001, ac-id 10001, state is up ( established )
    Preferred path Inactive : SR TE srte_c_9001_ep_172.16.0.1, Statically configured,
fallback enabled
    Tunnel : Down
    LSP: Up

/* If Fallback Disable is configured, and if the SR-policy is down, or if it misconfigured
 in dual homed mode, then the L2VPN PW will be down */
show l2vpn xconnect detail
EVPN: neighbor 172.16.0.1, PW ID: evi 1001, ac-id 10001, state is down ( local ready )
Preferred path Active : SR TE srte_c_9001_ep_172.16.0.1, Statically configured, fallback
disabled
Tunnel : Down
```

Running Configuration

```
/* Configure Prefix-SID in ISIS */
PE1:
```

```
configure
 segment-routing
  global-block 180000 200000
!
router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0031.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id 10.0.0.1
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 180010
```

P1:

```
configure
 segment-routing
  global-block 180000 200000

router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0021.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 180015
```

P2:

```
configure
 segment-routing
  global-block 180000 200000

router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0022.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local
```

```
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 180025
```

PE3:

```
configure
 segment-routing
  global-block 180000 200000

router isis core
 is-type level-2-only
 net 49.0002.0330.2000.3030.0030.0035.00
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 180020

/* Configure Adjacency-SID in ISIS */
```

PE1:

```
configure
 segment-routing
  local-block 15000 15999
!

router isis core
 !
interface Bundle-Ether121
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15101

interface TenGigE0/0/1/6
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15102
```

P1:

```
configure
 segment-routing
  local-block 15000 15999

router isis core
 !
interface Bundle-Ether121
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   metric 20
   adjacency-sid absolute 15200
```

```
interface TenGigE0/0/0/0/7
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   metric 20
   adjacency-sid absolute 15202
```

P2:

```
configure
 segment-routing
  local-block 15000 15999

router isis core
 !
interface TenGigE0/0/0/5
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   metric 20
   adjacency-sid absolute 15204

interface TenGigE0/0/0/0/7
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   metric 20
   adjacency-sid absolute 15201
```

PE3:

```
configure
 segment-routing
  local-block 15000 15999

router isis core
  !
interface TenGigE0/0/0/1
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15301
  !
 !
 interface TenGigE0/0/0/2
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15302

/* Configure Segment-list */
```

PE1:

```
configure
 segment-routing
  global-block 180000 200000
  traffic-eng
   logging
```

```
    policy status

segment-routing
 traffic-eng
  segment-list name pref_sid_to_PE3
   index 1 mpls label 180020
  !
 !
configure
 segment-routing
  local-block 15000 15999
  traffic-eng
   logging
    policy status

segment-routing
 traffic-eng
  segment-list name pref_adj_sid_to_PE3
   index 1 mpls label 15101
   index 2 mpls label 15202
  !
 !

/* Configure SR-TE policy */

segment-routing
 traffic-eng
  policy pref_sid_to_PE3
   color 9001 end-point ipv4 172.16.0.1
   candidate-paths
    preference 10
     explicit segment-list pref_sid_to_PE3
     !
    !
segment-routing
 traffic-eng
  policy pref_adj_sid_to_PE3
   color 9001 end-point ipv4 172.16.0.1
   candidate-paths
    preference 200
     explicit segment-list pref_adj_sid_to_PE3
     !
    !

/* You can configure multiple preferences for an SR policy. Among the configured preferences,
 the largest number takes the highest precedence */

segment-routing
 traffic-eng
  policy 1013
   color 1013 end-point ipv4 2.2.2.2
   candidate-paths
    preference 100
     explicit segment-list PE1-P1_BE121
     !
    preference 200
     explicit segment-list PE1-PE3-P1-t0016
     !
    preference 700
     explicit segment-list PE1-P1
     !

/* Configure EVPN VPWS over SR-TE policy */
PE1:
```

```
configure
 l2vpn
  pw-class 1001
   encapsulation mpls
    preferred-path sr-te policy srte_c_9001_ep_172.16.0.1 fallback disable
xconnect group evpn_vpws
  p2p evpn_vpws_1001
   interface tengi0/1/0/1.1001
   neighbor evpn evi 1001 target 10001 source 20001
    pw-class 1001
   !
```

Verify EVPN VPWS Preferred Path over SR-TE Policy Configuration

```
PE1#show segment-routing traffic-eng forwarding policy name pref_sid_to_PE3 detail
Policy        Segment         Outgoing    Outgoing            Next Hop        Bytes
Name          List            Label       Interface                           Switched
------------- --------------- ----------- ------------------- --------------- ------------
pref_sid_to_PE3

                              15102       TenGigE0/0/1/6       172.16.0.1       81950960
                Label Stack (Top -> Bottom): { 15101, 15102 }
                Path-id: 1, Weight: 0
                Packets Switched: 787990
   Local label: 34555
   Packets/Bytes Switched: 1016545/105720680
   (!): FRR pure backup


PE1#show segment-routing traffic-eng policy candidate-path name pref_sid_to_PE3

SR-TE policy database
---------------------

Color: 9001, End-point: 172.16.0.1
  Name: srte_c_9001_ep_172.16.0.1


PE1#show mpls forwarding tunnels sr-policy name  pref_sid_to_PE3
Tunnel        Outgoing    Outgoing     Next Hop        Bytes
Name          Label       Interface                    Switched
------------- ----------- ------------ --------------- ------------
pref_sid_to_PE3 (SR) 15102 TenGigE0/0/1/6 172.16.0.1       836516512


PE1#show l2vpn xconnect group evpn_vpws xc-name evpn_vpws_1001 detail
Group evpn_vpws, XC evpn_vpws_1001, state is up; Interworking none
  AC: Bundle-Ether12.1001, state is up
    Type VLAN; Num Ranges: 1
    Outer Tag: 1000
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1500; XC ID 0xc0000018; interworking none
    Statistics:
      packets: received 642304, sent 642244
      bytes: received 61661184, sent 61655424
      drops: illegal VLAN 0, illegal length 0
  EVPN: neighbor 172.16.0.1, PW ID: evi 1001, ac-id 10001, state is up ( established )
    XC ID 0xa0000007
    Encapsulation MPLS
    Source address 10.10.10.10
    Encap type Ethernet, control word enabled
    Sequencing not set
```

```
Preferred path Active : SR TE pref_sid_to_PE3, Statically configured, fallback disabled
Tunnel : Up
Load Balance Hashing: src-dst-mac
```

Associated Commands

- adjacency-sid
- index
- prefix-sid
- router isis
- segment-routing

The applicable segment routing commands are described in the *Segment Routing Command Reference for Cisco NCS 5500 Series Routers and Cisco NCS 540 Series Routers*

Related Topics

- Overview of Segment Routing , on page 498
- How Segment Routing Works , on page 499
- Segment Routing Global Block , on page 500

L2VPN VPWS Preferred Path over SR-TE Policy

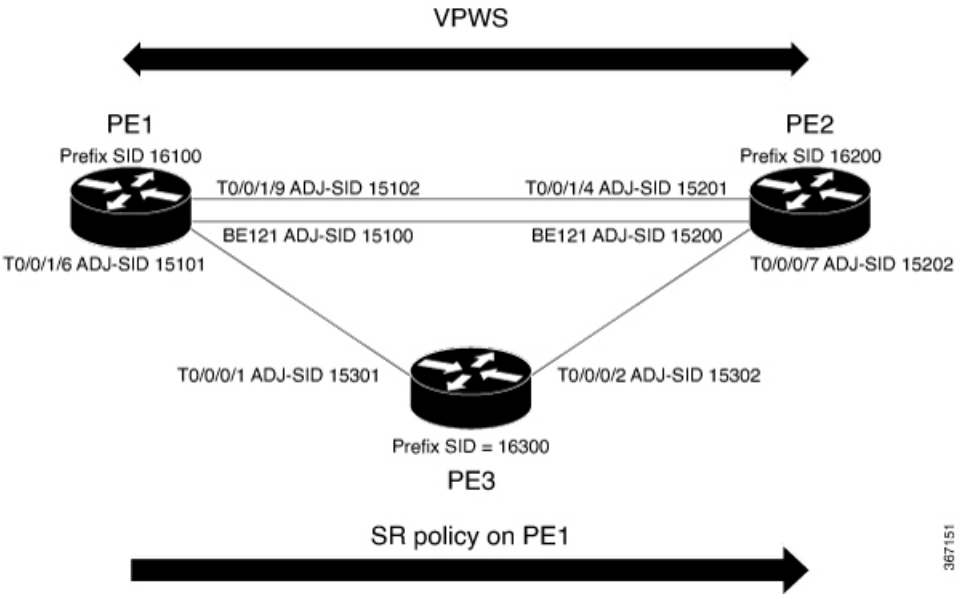
All L2VPN services such as VPLS, VPWS, and so on must use L2VPN preferred-path while using TE (SR-TE, and RSPV-TE) services as transport.

L2VPN VPWS Preferred Path over SR-TE Policy feature allows you to set the preferred path between the two end-points for L2VPN Virtual Private Wire Service (VPWS) using SR-TE policy.

Configure L2VPN VPWS Preferred Path over SR-TE Policy

Perform the following steps to configure L2VPN VPWS Preferred Path over SR-TE Policy feature. The following figure is used as a reference to explain the configuration steps.

Figure 68: L2VPN VPWS Preferred Path over SR-TE Policy



- Configure Prefix-SID on IGP — The following examples show how to configure prefix-SID in IS-IS.
- Configure Adjacency-SID on IGP — The following examples show how to configure Adjacency-SID in IS-IS.
- Configure segment-list
- Configure SR-TE policy
- Configure VPWS over SR-TE policy

Configure Prefix-SID in IS-IS

Configure Prefix-SID on PE1, PE2, and PE3.

```
/* Configure Prefix-SID on PE1 */

Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0031.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id 10.0.0.1
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16100
```

```
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on PE2 */

Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0021.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16200
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on PE3 */

Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.3030.0030.0035.00
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16300
Route(config-isis-af)# commit
Route(config-isis-af)# exit
```

Configure Adjacency-SID in IS-IS

Configure Adjacency-SID on PE1, PE2, and PE3.

```
/* Configure Adjacency-SID on PE1 */

Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15100
```

```
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15101
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15102
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on PE2 */

Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15200
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15201
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/7
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15202
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on PE3 */

Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/1
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
```

```
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15301
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/2
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15302
Route(config-isis-if-af)# commit
```

Configure Segment-list

Configure segment-list on PE1, PE2, and PE3.

```
/* Configure segment-list on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE1-PE2
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE3
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE2-PE3
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# index 2 mpls label 16300
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE2_bad
Router(config-sr-te-sl)# index 1 mpls label 16900
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# index 2 mpls label 16200
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE2_BE121
Router(config-sr-te-sl)# index 1 mpls label 15100
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2_link
Router(config-sr-te-sl)# index 1 mpls label 15101
Router(config-sr-te-sl)# index 2 mpls label 15302
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2-t0016
Router(config-sr-te-sl)# index 1 mpls label 15101
Router(config-sr-te-sl)# index 2 mpls label 16200
Router(config-sr-te-sl)# commit
```

```
/* Configure segment-list on PE2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE2-PE1
Router(config-sr-te-sl)# index 1 mpls label 16100
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE2-PE3-PE1
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# index 2 mpls label 16100
Router(config-sr-te-sl)# commit

/* Configure segment-list on PE3 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE3-PE1
Router(config-sr-te-sl)# index 1 mpls label 16100
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE3-PE2-PE1
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# index 2 mpls label 16100
Router(config-sr-te-sl)# commit
```

Configure SR-TE Policy

```
/* Configure SR-TE policy */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 100
Router(config-sr-te-policy)# color 1 end-point ipv4 172.16.0.1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 400
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 500 <-----------------largest number takes the
precedence
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 1013
Router(config-sr-te-policy)# color 1013 end-point ipv4 172.16.0.1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 100
```

```
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2_BE121
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 200
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2-t0016
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 500
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 600
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 700
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2_link
Router(config-sr-te-pp-info)# commit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 1300
Router(config-sr-te-policy)# color 1300 end-point ipv4 192.168.0.1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 100
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3
Router(config-sr-te-pp-info)# commit
!
```

Configure VPWS over SR-TE Policy

Note Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to display the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name 1300

SR-TE policy database
---------------------
Color: 1300, End-point: 192.168.0.1
 Name: srte_c_1300_ep_192.168.0.1

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pw1300
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# load-balancing
Router(config-l2vpn-pwc-mpls-load-bal)# flow-label both
Router(config-l2vpn-pwc-mpls-load-bal)# exit
!
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_1300_ep_192.168.0.1
fallback disable
Router(config-l2vpn-pwc-mpls)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xcon1
Router(config-l2vpn-xc)# p2p vplw1002
```

```
Router(config-l2vpn-xc-p2p)# interface TenGigE
Router(config-l2vpn-xc-p2p)# neighbor 192.168.0.1 pw-id 1002
Router(config-l2vpn-xc-p2p-pw)# pw-class pw1300
Router(config-l2vpn-xc-p2p-pw)# commit
Router(config-l2vpn-xc-p2p-pw)# exit
```

Running Configuration

```
/* Configure prefix-SID */
PE1:
router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0031.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id 10.0.0.1
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 16100

PE2:
router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0021.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 16200

PE3:
router isis core
 is-type level-2-only
 net 49.0002.0330.2000.3030.0030.0035.00
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 16300

/* Configure Adjacency-SID */
```

PE1:

```
router isis core
!
 interface Bundle-Ether121
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15100
  !
 interface TenGigE

  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15101
  !
 interface TenGigE
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15102
```

PE2

```
router isis core
 !
interface Bundle-Ether121
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15200

interface TenGigE0/0/0/0/4
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15201

interface TenGigE0/0/0/0/7
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15202
```

PE3:

```
router isis core
!
interface TenGigE0/0/0/1
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15301
  !
 !
 interface TenGigE0/0/0/2
  circuit-type level-2-only
  point-to-point
  hello-padding disable
```

```
  address-family ipv4 unicast
   adjacency-sid absolute 15302

/* Configure segment-list */
PE1:
segment-routing
 global-block 16000 23999
 local-block 15000 15999
  traffic-eng
 segment-list name PE1-PE2
   index 1 mpls label 16200
  !
  segment-list name PE1-PE3
   index 1 mpls label 16300
  !
  segment-list name PE1-PE2-PE3
   index 1 mpls label 16200
   index 2 mpls label 16300
  !
  segment-list name PE1-PE2_bad
   index 1 mpls label 16900
  !
  segment-list name PE1-PE3-PE2
   index 1 mpls label 16300
   index 2 mpls label 16200
  !
  segment-list name PE1-PE2_BE121
   index 1 mpls label 15100
!
  segment-list name PE1-PE3-PE2_link
   index 1 mpls label 15101
   index 2 mpls label 15302
  !

  segment-list name PE1-PE3-PE2-t0016
   index 1 mpls label 15101
   index 2 mpls label 16200

PE2:
segment-routing
 global-block 16000 23999
 local-block 15000 15999
 traffic-eng
  segment-list name PE2-PE1
   index 1 mpls label 16100
  !
  segment-list name PE2-PE3-PE1
   index 1 mpls label 16300
   index 2 mpls label 16100

PE3:
segment-routing
 global-block 16000 23999
 local-block 15000 15999
 traffic-eng
  segment-list name PE3-PE1
   index 1 mpls label 16100
  !
  segment-list name PE3-PE2-PE1
   index 1 mpls label 16200
   index 2 mpls label 16100

/* Configure SR-TE policy */
```

```
segment-routing
 traffic-eng
  policy 100
   color 1 end-point ipv4 172.16.0.1
   candidate-paths
    preference 400
     explicit segment-list PE1-PE3-PE2
     !
    preference 500
     explicit segment-list PE1-PE2

 policy 1013
   color 1013 end-point ipv4 172.16.0.1
   candidate-paths
    preference 100
     explicit segment-list PE1-PE2_BE121
     !
    preference 200
     explicit segment-list PE1-PE3-PE2-t0016
     !
    preference 500
     explicit segment-list PE1-PE2
      !
    preference 600
     explicit segment-list PE1-PE3-PE2
     !
    preference 700
     explicit segment-list PE1-PE3-PE2_link
     !
 policy 1300
   color 1300 end-point ipv4 192.168.0.1
   candidate-paths
    preference 100
     explicit segment-list PE1-PE3
     !


/*Configure VPWS over SR-TE policy
l2vpn
 pw-class pw1300
  encapsulation mpls
    load-balancing
    flow-label both
   preferred-path sr-te policy srte_c_1300_ep_192.168.0.1 fallback disable


Xconnect group xcon1
  p2p vplw1002
   interface TenGigE
   neighbor 192.168.0.1 pw-id 1002
    pw-class pw1300
```

Verify L2VPN VPWS Preferred Path over SR-TE Policy Configuration

```
/* The prefix-sid and Adjacency-sid must be in the SR topology */

PE1#show segment-routing traffic-eng ipv4 topology | inc Prefix
Thu Feb  1 20:28:43.343 EST
  Prefix SID:
    Prefix 10.0.0.1, label 16100 (regular)
  Prefix SID:
    Prefix 192.168.0.1, label 16300 (regular)
```

```
  Prefix SID:
    Prefix 172.16.0.1, label 16200 (regular)


PE1#show segment-routing traffic-eng ipv4 topology | inc Adj SID
Thu Feb  1 20:30:25.760 EST
    Adj SID: 61025 (unprotected) 15102 (unprotected)
    Adj SID: 61023 (unprotected) 15101 (unprotected)
    Adj SID: 65051 (unprotected) 15100 (unprotected)
    Adj SID: 41516 (unprotected) 15301 (unprotected)
    Adj SID: 41519 (unprotected) 15302 (unprotected)
    Adj SID: 46660 (unprotected) 15201 (unprotected)
    Adj SID: 24003 (unprotected) 15202 (unprotected)
    Adj SID: 46675 (unprotected) 15200 (unprotected)
PE1# show segment-routing traffic-eng policy candidate-path name 100

SR-TE policy database
---------------------

Color: 100, End-point: 172.16.0.1
 Name: srte_c_1_ep_172.16.0.1



PE1#show segment-routing traffic-eng policy name 100
Thu Feb  1 23:16:58.368 EST

SR-TE policy database
---------------------

Name: 100 (Color: 1, End-point: 172.16.0.1)
  Status:
    Admin: up  Operational: up for 05:44:25 (since Feb  1 17:32:34.434)
  Candidate-paths:
    Preference 500:
      Explicit: segment-list PE1-PE2 (active)
        Weight: 0, Metric Type: IGP
          16200 [Prefix-SID, 172.16.0.1]
    Preference 400:
      Explicit: segment-list PE1-PE3-PE2 (inactive)
      Inactive Reason: unresolved first label
        Weight: 0, Metric Type: IGP
  Attributes:
    Binding SID: 27498
      Allocation mode: dynamic
      State: Programmed
      Policy selected: yes
    Forward Class: 0

PE1#show segment-routing traffic-eng policy name 1013
Thu Feb  1 21:20:57.439 EST

SR-TE policy database
---------------------
Name: 1013 (Color: 1013, End-point: 172.16.0.1)
  Status:
    Admin: up  Operational: up for 00:06:36 (since Feb  1 21:14:22.057)
  Candidate-paths:
    Preference 700:
      Explicit: segment-list PE1-PE3-PE2_link (active)
        Weight: 0, Metric Type: IGP
          15101 [Adjacency-SID, 13.1.1.1 - 13.1.1.2]
          15302
    Preference 600:
```

```
      Explicit: segment-list PE1-PE3-PE2 (inactive)
      Inactive Reason:
        Weight: 0, Metric Type: IGP
    Preference 500:
      Explicit: segment-list PE1-PE2 (inactive)
      Inactive Reason:
        Weight: 0, Metric Type: IGP
    Preference 200:
      Explicit: segment-list PE1-PE3-PE2-t0016 (inactive)
      Inactive Reason: unresolved first label
        Weight: 0, Metric Type: IGP
    Preference 100:
      Explicit: segment-list PE1-PE2_BE121 (inactive)
      Inactive Reason: unresolved first label
        Weight: 0, Metric Type: IGP
  Attributes:
    Binding SID: 27525
      Allocation mode: dynamic
      State: Programmed
      Policy selected: yes
    Forward Class: 0

PE1#show segment-routing traffic-eng forwarding policy name 100
Thu Feb  1 23:19:28.951 EST
Policy        Segment         Outgoing    Outgoing            Next Hop        Bytes
Name          List            Label       Interface                           Switched
------------- --------------- ----------- ------------------- --------------- ------------
100           PE1-PE2         Pop         Te           12.1.9.2        0
                                         Pop         BE121               121.1.0.2       0


PE1#show segment-routing traffic-eng forwarding policy name 1013 detail
Thu Feb  1 21:22:46.069 EST
Policy        Segment         Outgoing    Outgoing            Next Hop        Bytes
Name          List            Label       Interface                           Switched
------------- --------------- ----------- ------------------- --------------- ------------
1013          PE1-PE3-PE2_link
                              15302       Te           13.1.1.2        0
                 Label Stack (Top -> Bottom): { 15302 }
                 Path-id: 1, Weight: 0
                 Packets Switched: 0
   Local label: 24005
   Packets/Bytes Switched: 0/0
   (!): FRR pure backup

PE1#show mpls forwarding tunnels sr-policy name 1013
Thu Feb  1 21:23:22.743 EST
Tunnel        Outgoing    Outgoing     Next Hop        Bytes
Name          Label       Interface                    Switched
------------- ----------- ------------ --------------- ------------
1013         (SR) 15302       Te    13.1.1.2        0
```

Associated Commands

- adjacency-sid
- index
- prefix-sid

- router isis
- segment-routing

The applicable segment routing commands are described in the *Segment Routing Command Reference for Cisco NCS 5500, NCS 540 Series Routers, and NCS 560 Series Routers*.

Related Topics

- Overview of Segment Routing , on page 498
- How Segment Routing Works , on page 499
- Segment Routing Global Block , on page 500

EVPN VPWS On-Demand Next Hop with SR-TE

The EVPN VPWS On-Demand Next Hop with SR-TE feature enables you to fetch the best path to send traffic from the source to destination in a point-to-point service using IOS XR Traffic Controller (XTC). On-Demand Next Hop (ODN) with SR-TE is supported on EVPN Virtual Private Wire Service (VPWS) and Flexible Cross Connect (FXC) VLAN-unaware service.

When redistributing routing information across domains, provisioning of multi-domain services (Layer2 VPN and Layer 3 VPN) poses complexity and scalability issues. ODN with SR-TE feature delegates computation of an end-to-end Label Switched Path (LSP) to a path computation element (PCE). This PCE includes constraints and policies without any redistribution. It then installs the reapplied multi-domain LSP for the duration of the service into the local forwarding information base(FIB).

ODN uses BGP dynamic SR-TE capabilities and adds the path to the PCE. The PCE has the ability to find and download the end-to-end path based on the requirements. ODN triggers an SR-TE auto-tunnel based on the defined BGP policy. The PCE learns real-time topologies through BGP and/or IGP.

IOS XR Traffic Controller (XTC)

The path computation element (PCE) describes a set of procedures by which a path computation client (PCC) reports and delegates control of head-end tunnels sourced from the PCC to a PCE peer. The PCE peer requests the PCC to update and modify parameters of LSPs it controls. It also enables a PCC to allow the PCE to initiate computations and to perform network-wide orchestration.

Restrictions

- Maximum number of auto-provisioned TE policies is 1000.
- EVPN validates if the route is for a single home next hop, otherwise it issues an error message about a dangling SR-TE policy, and continue to setup EVPN-VPWS without it. EVPN relies on ESI value being zero to determine if this is a single home or not.

  To disable EVPN dual homing, configure bundle-Ether AC with ESI value set to zero.

```
evpn
interface Bundle-Ether12
ethernet-segment
identifier type 0 00.00.00.00.00.00.00.00.00
/* Or globally */
```

```
evpn
ethernet-segment type 1 auto-generation-disable
```

Configure EVPN VPWS On Demand Next Hop with SR-TE

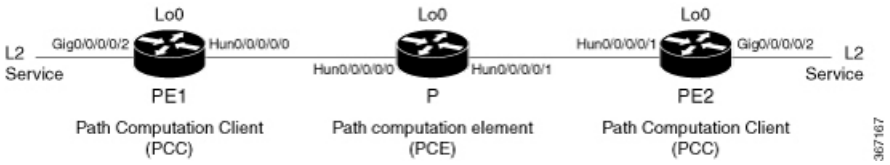
Perform the following steps to configure EVPN VPWS On Demand Next Hop with SR-TE. The following figure is used as a reference to explain the configuration steps:

- Configure Prefix-SID in ISIS
- Configure SR-TE
- Configure PCE and PCC
- Configure SR color
- Configure EVPN route policy
- Configure BGP
- Configure EVPN VPWS
- Configure Flexible Cross-connect Service (FXC) VLAN-unaware

Topology

Consider a topology where EVPN VPWS is configured on PE1 and PE2. Traffic is sent from PE1 to PE2 using SR-TE in the core. The PCE, which is configured on the P router, calculates the best path from PE1 to PE2. Path computation client (PCC) is configured on PE1 and PE2.

Figure 69: EVPN VPWS On Demand Next Hop with SR-TE



Configuration Example

Configure Prefix-SID in ISIS

Configure Prefix-SID in ISIS and topology-independent loop-free alternate path (TI-LFA) in the core such that each router uses a unique segment identifier associated with the prefix.

```
/* Configure Prefix-SID in ISIS and TI-LFA  on PE1 */

Router# configure
Route(config)# router isis ring
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0001.1921.6800.1001.00
Route(config-isis)# segment-routing global-block 30100 39100
Route(config-isis)# nsr
Route(config-isis)# distribute link-state
Route(config-isis)# nsf cisco
Route(config-isis)# address-family ipv4 unicast
```

```
Route(config-isis-af)# metric-style wide
Route(config-isis-af)# mpls traffic-eng level-1
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 30101
Route(config-isis-af)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/0
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# fast-reroute per-prefix
Route(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-if-af)# commit

/*Configure Prefix-SID in ISIS and TI-LFA  on P router */

Router# configure
Route(config)# router isis ring
Route(config-isis)# net 49.0001.1921.6800.1002.00
Route(config-isis)# segment-routing global-block 30100 39100
Route(config-isis)# nsr
Route(config-isis)# distribute link-state
Route(config-isis)# nsf cisco
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide
Route(config-isis-af)# mpls traffic-eng level-1
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 30102
Route(config-isis-af)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/0
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# fast-reroute per-prefix
Route(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-if-af)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/1
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# fast-reroute per-prefix
Route(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-if-af)# commit

/* Configure Prefix-SID in ISIS and TI-LFA  on PE2 */

Router# configure
Route(config)# router isis ring
Route(config-isis)# net 49.0001.1921.6800.1003.00
Route(config-isis)# segment-routing global-block 30100 39100
Route(config-isis)# nsr
```

```
Route(config-isis)# distribute link-state
Route(config-isis)# nsf cisco
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide
Route(config-isis-af)# mpls traffic-eng level-1
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 30103
Route(config-isis-af)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/1
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# fast-reroute per-prefix
Route(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-if-af)# commit
```

Configure SR-TE

Configure SR-TE for P and PE routers.

```
/Configure SR-TE on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# on-demand color 1
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pcep
Router(config-sr-te-color-dyn-mpls)# exit
!
Router(config-sr-te)# on-demand color 2
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pcep
Router(config-sr-te-color-dyn-mpls)# exit
!
Router(config-sr-te)# on-demand color 3
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pcep
Router(config-sr-te-color-dyn-mpls)# commit

/*Configure SR-TE on P router */
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# commit


/Configure SR-TE on PE2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# on-demand color 11
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pcep
```

```
Router(config-sr-te-color-dyn-mpls)# exit
!
Router(config-sr-te)# on-demand color 12
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pcep
Router(config-sr-te-color-dyn-mpls)# exit
!
Router(config-sr-te)# on-demand color 13
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pcep
Router(config-sr-te-color-dyn-mpls)# commit
```

Configure PCE and PCC

Configure PCE on P router, and PCC on PE1 and PE2. Optionally, you can configure multiple PCEs as well.

```
/* Configure PCC on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# pcc
Router(config-sr-te-pcc)# source-address ipv4 205.1.0.1
Router(config-sr-te-pcc)# pce address ipv4 205.2.0.2
Router(config-sr-te-pcc)# commit

/* Configure PCE on P router */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# exit
Router(config)# pce
Router(config-pce)# address ipv4 205.2.0.2
Router(config-pce)# commit

/* Configure PCC on PE2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# pcc
Router(config-sr-te-pcc)# source-address ipv4 205.3.0.3
Router(config-sr-te-pcc)# pce address ipv4 205.2.0.2
Router(config-sr-te-pcc)# commit
```

Configure SR Color

Configure SR colors on PE routers.

```
/* Define SR color on PE1 */

Router# configure
Router(config)# extcommunity-set opaque color1
Router(config-ext)# 1
Router(config-ext)# end-set
!
Router(config)# extcommunity-set opaque color2
Router(config-ext)# 2
```

```
Router(config-ext)# end-set
!
Router(config)# extcommunity-set opaque color3
Router(config-ext)# 3
Router(config-ext)# end-set
!
/* Define SR color on PE2 */

Router# configure
Router(config)# extcommunity-set opaque color11
Router(config-ext)# 11
Router(config-ext)# end-set
!
Router(config)# extcommunity-set opaque color12
Router(config-ext)# 12
Router(config-ext)# end-set
!
Router(config)# extcommunity-set opaque color13
Router(config-ext)# 13
Router(config-ext)# end-set
!
```

Configure EVPN Route Policy

Configure EVPN route policy on PE1 and PE2. This example shows how to define the route policy language and track the EVPN route. The "rd" refers to the address of the PE and acts as Ethernet virtual interconnect for the L2 service.

```
/* Configure EVPN route policy on PE1 */

Router# configure
Router(config)# route-policy evpn_odn_policy
Router(config-rpl)# if rd in (205.3.0.3:2) then
Router(config-rpl-if)# set extcommunity color color1
Router(config-rpl-if)# set next-hop 205.3.0.3
Router(config-rpl-if)# elseif rd in (205.3.0.3:3) then
Router(config-rpl-elseif)# set extcommunity color color2
Router(config-rpl-elseif)# set next-hop 205.3.0.3
Router(config-rpl-elseif)# elseif rd in (205.3.0.3:4) then
Router(config-rpl-elseif)# set extcommunity color color3
Router(config-rpl-elseif)# set next-hop 205.3.0.3
Router(config-rpl-elseif)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy


/* Configure EVPN route policy on PE2 */

Router# configure
Router(config)# route-policy evpn_odn_policy
Router(config-rpl)# if rd in (205.1.0.1:2) then
Router(config-rpl-if)# set extcommunity color color11
Router(config-rpl-if)# set next-hop 205.1.0.1
Router(config-rpl-if)# elseif rd in (205.1.0.1:3) then
Router(config-rpl-elseif)# set extcommunity color color12
Router(config-rpl-elseif)# set next-hop 205.1.0.1
Router(config-rpl-elseif)# elseif rd in (205.1.0.1:4) then
Router(config-rpl-elseif)# set extcommunity color color13
Router(config-rpl-elseif)# set next-hop 205.1.0.1
Router(config-rpl-elseif)# endif
```

```
Router(config-rpl)# pass
Router(config-rpl)# end-policy
```

Configure BGP

Configure BGP on PE1 and PE2.

```
/* Configure BGP on PE1 */

Router# configure
Router(config)# router bgp 100
Routerconfig-bgp)# bgp router-id 205.1.0.1
Routerconfig-bgp)# bgp graceful-restart
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
!
Router(config-bgp)# neighbor 205.3.0.3
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy evpn_odn_policy in
Router(config-rpl)# commit

/* Configure BGP on PE2 */

Router# configure
Router(config)# router bgp 100
Routerconfig-bgp)# bgp router-id 205.3.0.3
Routerconfig-bgp)# bgp graceful-restart
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
!
Router(config-bgp)# neighbor 205.1.0.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy evpn_odn_policy in
Router(config-rpl)# commit
```

Configure EVPN VPWS

Configure EVPN VPWS on PE1 and PE2.

```
/* Configure EVPN VPWS on PE1 */

Router# configure
Router(config)# interface GigE0/0/0/2.2 l2transport
Router(config-subif)# encapsulation dot1q 1
Router# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e1_10
Router(config-l2vpn-xc-p2p)# interface GigE0/0/0/2.2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 2 target 10 source 10
Router(config-l2vpn-xc-p2p)#commit

/* Configure EVPN VPWS on PE2 */
```

```
Router# configure
Router(config)# interface GigE0/0/0/2.4 l2transport
Router(config-subif)# encapsulation dot1q 1
Router# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e3_30
Router(config-l2vpn-xc-p2p)# interface GigE0/0/0/2.4
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 2 target 10 source 10
Router(config-l2vpn-xc-p2p)#commit
```

Configure Flexible Cross-connect Service (FXC) VLAN-unaware

```
/* Configure FXC on PE1 */

Router# configure
Router(config)# interface GigE0/0/0/2.3 l2transport
Router(config-subif)# encapsulation dot1q 3
Router# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware evpn_vu
Router(config-l2vpn-fxs-vu)# interface GigE0/0/0/2.3
Router(config-l2vpn-fxs-vu)# neighbor evpn evi 3 target 20
Router(config-l2vpn-fxs-vu)#commit

/* Configure FXC on PE2 */

Router# configure
Router(config)# interface GigE0/0/0/2.3 l2transport
Router(config-subif)# encapsulation dot1q 3
Router# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware evpn_vu
Router(config-l2vpn-fxs-vu)# interface GigE0/0/0/2.3
Router(config-l2vpn-fxs-vu)# neighbor evpn evi 3 target 20
Router(config-l2vpn-fxs-vu)#commit
```

Running Configuration

```
/* Configure Prefix-SID in ISIS and TI-LFA */

PE1:

configure
 router isis ring
  net 49.0001.1921.6800.1001.00
  segment-routing global-block 30100 39100
  nsr
  distribute link-state
  nsf cisco
  address-family ipv4 unicast
   metric-style wide
   mpls traffic-eng level-1
   mpls traffic-eng router-id Loopback0
   segment-routing mpls
```

```
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 30101
  !
 !
 interface HundredGigE0/0/0/0
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa
  !
 !
```

P:

```
configure
 router isis ring
  net 49.0001.1921.6800.1002.00
  segment-routing global-block 30100 39100
  nsr
  distribute link-state
  nsf cisco
  address-family ipv4 unicast
   metric-style wide
   mpls traffic-eng level-1
   mpls traffic-eng router-id Loopback0
   segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 30102
  !
 !
 interface HundredGigE0/0/0/0
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa
  !
 !
 interface HundredGigE0/0/0/1
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa
  !
```

PE2:

```
configure
 router isis ring
  net 49.0001.1921.6800.1003.00
  segment-routing global-block 30100 39100
  nsr
  distribute link-state
  nsf cisco
  address-family ipv4 unicast
```

```
   metric-style wide
   mpls traffic-eng level-1
   mpls traffic-eng router-id Loopback0
   segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 30103
  !
 !
 interface HundredGigE0/0/0/1
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa
  !
 !


/* Configure SR-TE */

PE1:

configure
 segment-routing
  traffic-eng
   on-demand color 1
    dynamic mpls
     pce
    !
   !
   on-demand color 2
    dynamic mpls
     pce
     !
    !
   !
   on-demand color 3
    dynamic mpls
     pce
     !
P:

configure
 segment-routing
  traffic-eng
!

PE2:

configure
 segment-routing
  traffic-eng
   on-demand color 11
    dynamic mpls
     pce
    !
   !
   on-demand color 12
    dynamic mpls
     pce
     !
```

```
   !
  !
  on-demand color 13
   dynamic mpls
    pce
    !

/* Configure PCE and PCC */

PE1:

configure
 segment-routing
  traffic-eng
   pcc
    source-address ipv4 205.1.0.1
    pce address ipv4 205.2.0.2
    !
P:

configure
 segment-routing
  traffic-eng
 pce
  address ipv4 205.2.0.2
    !

PE2:

configure
 segment-routing
  traffic-eng
   pcc
    source-address ipv4 205.3.0.3
    pce address ipv4 205.2.0.2
    !

/* Configure SR Color */

PE1:

configure
 extcommunity-set opaque color1
  1
end-set
!
 extcommunity-set opaque color2
  2
end-set
!
 extcommunity-set opaque color3
  3
end-set
!

PE2:

configure
 extcommunity-set opaque color11
  11
end-set
!
 extcommunity-set opaque color12
  12
```

```
end-set
!
 extcommunity-set opaque color13
  13
end-set
!

/* Configure EVPN route policy */

PE1:

configure
 route-policy evpn_odn_policy
  if rd in (205.3.0.3:2) then
    set extcommunity color color1
    set next-hop 205.3.0.3
  elseif rd in (205.3.0.3:3) then
    set extcommunity color color2
    set next-hop 205.3.0.3
  elseif rd in (205.3.0.3:4) then
    set extcommunity color color3
    set next-hop 205.3.0.3
  endif
 pass
end-policy

PE2:

configure
 route-policy evpn_odn_policy
  if rd in (205.1.0.1:2) then
    set extcommunity color color11
    set next-hop 205.1.0.1
  elseif rd in (205.1.0.1:3) then
    set extcommunity color color12
    set next-hop 205.1.0.1
  elseif rd in (205.1.0.1:4) then
    set extcommunity color color13
    set next-hop 205.1.0.1
  endif
 pass
end-policy

/* Configure BGP */

PE1:

configure
 router bgp 100
  bgp router-id 205.1.0.1
  bgp graceful-restart
  address-family l2vpn evpn
 !
 neighbor 205.3.0.3
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
   route-policy evpn_odn_policy in
  !

PE2:

configure
 router bgp 100
```

```
  bgp router-id 205.3.0.3
  bgp graceful-restart
  address-family l2vpn evpn
 !
 neighbor 205.1.0.1
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
   route-policy evpn_odn_policy in
  !

/* Configure EVPN VPWS */

PE1:

configure
 interface GigE0/0/0/2.2 l2transport
  encapsulation dot1q 1
!
l2vpn
 xconnect group evpn_vpws
  p2p e1_10
   interface GigE0/0/0/2.2
   neighbor evpn evi 2 target 10 source 10
  !
 !

PE2:

configure
 interface GigE0/0/0/2.4 l2transport
  encapsulation dot1q 1
!
l2vpn
 xconnect group evpn_vpws
  p2p e3_30
   interface GigE0/0/0/2.4
   neighbor evpn evi 2 target 10 source 10
  !
 !
!

/* Configure Flexible Cross-connect Service (FXC) */

PE1:

configure
 interface GigE0/0/0/2.3 l2transport
  encapsulation dot1q 3
!
l2vpn
 flexible-xconnect-service vlan-unaware evpn_vu
  interface GigE0/0/0/2.3
  neighbor evpn evi 3 target 20
 !
!

PE2:

configure
 interface GigE0/0/0/2.3 l2transport
  encapsulation dot1q 3
!
l2vpn
```

```
 flexible-xconnect-service vlan-unaware evpn_vu
  interface GigE0/0/0/2.3
  neighbor evpn evi 3 target 20
 !
!
```

Verify EVPN VPWS On Demand Next Hop with SR-TE Configuration

Verify if SR-TE policy is auto-provisioned for each L2 service configured on EVPN ODN.

```
PE1# show segment-routing traffic-eng policy

SR-TE policy database
---------------------

Name: bgp_AP_1 (Color: 1, End-point: 205.3.0.3)
    Status:
      Admin: up  Operational: up for 07:16:59 (since Oct  3 16:47:04.541)
    Candidate-paths:
      Preference 100:
        Dynamic (pce 205.2.0.2) (active)
          Weight: 0
            30103 [Prefix-SID, 205.3.0.3]
    Attributes:
      Binding SID: 68007
        Allocation mode: dynamic
        State: Programmed
        Policy selected: yes
      Forward Class: 0
      Distinguisher: 0
    Auto-policy info:
      Creator: BGP
      IPv6 caps enable: no
PE1#show l2vpn xconnect group evpn_vpws xc-name evpn_vpws_1001 detail
Group evpn_vpws, XC evpn_vpws_1001, state is up; Interworking none
  AC: Bundle-Ether12.1001, state is up
    Type VLAN; Num Ranges: 1
    Outer Tag: 1000
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1500; XC ID 0xc0000018; interworking none
    Statistics:
      packets: received 642304, sent 642244
      bytes: received 61661184, sent 61655424
      drops: illegal VLAN 0, illegal length 0
  EVPN: neighbor 20.20.20.20, PW ID: evi 1001, ac-id 10001, state is up ( established )
    XC ID 0xa0000007
    Encapsulation MPLS
    Source address 10.10.10.10
    Encap type Ethernet, control word enabled
    Sequencing not set
    Preferred path Active : SR TE pref_sid_to_PE3, On-Demand, fallback enabled
    Tunnel : Up
    Load Balance Hashing: src-dst-mac

PE1#show bgp l2vpn evpn route-type 1

BGP router identifier 205.1.0.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0 RD version: 0
```

```
BGP main routing table version 36
BGP NSR Initial initsync version 25 (Reached)
BGP NSR/ISSU Sync-Group versions 36/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
Network Next Hop Metric LocPrf Weight Path
Route Distinguisher: 205.1.0.1:2 (default for vrf VPWS:2)
*>i[1][0000.0000.0000.0000.0000][1]/120
205.3.0.3 T:bgp_AP_1
100 0 i

PE1# show evpn evi ead detail

EVI Ethernet Segment Id EtherTag Nexthop Label SRTE IFH
----- ---------------------- -------- ------------------------------------ --------
----------
2 0000.0000.0000.0000.0000 1 205.3.0.3 24000 0x5a0
Source: Remote, MPLS
```

Associated Commands

- adjacency-sid
- index
- prefix-sid
- router isis
- segment-routing

The applicable segment routing commands are described in the *Segment Routing Command Reference for Cisco NCS 5500 Series Routers, Cisco NCS 540 Series Routers, and Cisco NCS 560 Series Routers*.

Related Topics

- Overview of Segment Routing , on page 498
- How Segment Routing Works , on page 499
- Segment Routing Global Block , on page 500

Overview of Segment Routing

Segment Routing (SR) is a flexible, scalable way of doing source routing. The source chooses a path and encodes it in the packet header as an ordered list of segments. Segments are identifier for any type of instruction. Each segment is identified by the segment ID (SID) consisting of a flat unsigned 32-bit integer. Segment instruction can be:

- Go to node N using the shortest path
- Go to node N over the shortest path to node M and then follow links Layer 1, Layer 2, and Layer 3
- Apply service S

With segment routing, the network no longer needs to maintain a per-application and per-flow state. Instead, it obeys the forwarding instructions provided in the packet.

Segment Routing relies on a small number of extensions to Cisco Intermediate System-to-Intermediate System (IS-IS) and Open Shortest Path First (OSPF) protocols. It can operate with an MPLS (Multiprotocol Label Switching) or an IPv6 data plane, and it integrates with the rich multi service capabilities of MPLS, including Layer 3 VPN (L3VPN), Virtual Private Wire Service (VPWS), and Ethernet VPN (EVPN).

Segment routing can be directly applied to the Multiprotocol Label Switching (MPLS) architecture with no change in the forwarding plane. Segment routing utilizes the network bandwidth more effectively than traditional MPLS networks and offers lower latency. A segment is encoded as an MPLS label. An ordered list of segments is encoded as a stack of labels. The segment to process is on the top of the stack. The related label is popped from the stack, after the completion of a segment.

Segment Routing provides automatic traffic protection without any topological restrictions. The network protects traffic against link and node failures without requiring additional signaling in the network. Existing IP fast re-route (FRR) technology, in combination with the explicit routing capabilities in Segment Routing guarantees full protection coverage with optimum backup paths. Traffic protection does not impose any additional signaling requirements.

How Segment Routing Works

A router in a Segment Routing network is capable of selecting any path to forward traffic, whether it is explicit or Interior Gateway Protocol (IGP) shortest path. Segments represent subpaths that a router can combine to form a complete route to a network destination. Each segment has an identifier (Segment Identifier) that is distributed throughout the network using new IGP extensions. The extensions are equally applicable to IPv4 and IPv6 control planes. Unlike the case for traditional MPLS networks, routers in a Segment Router network do not require Label Distribution Protocol (LDP) and Resource Reservation Protocol - Traffic Engineering (RSVP-TE) to allocate or signal their segment identifiers and program their forwarding information.

There are two ways to configure segment routing:

- SR-TE policy under "segment-routing traffic-eng" sub-mode
- TE tunnel with SR option under "mpls traffic-eng" sub-mode

Note However, you can configure the above mentioned L2VPN and EVPN services using only "segment-routing traffic-eng" sub-mode.

Each router (node) and each link (adjacency) has an associated segment identifier (SID). Node segment identifiers are globally unique and represent the shortest path to a router as determined by the IGP. The network administrator allocates a node ID to each router from a reserved block. On the other hand, an adjacency segment ID is locally significant and represents a specific adjacency, such as egress interface, to a neighboring router. Routers automatically generate adjacency identifiers outside of the reserved block of node IDs. In an MPLS network, a segment identifier is encoded as an MPLS label stack entry. Segment IDs direct the data along a specified path. There are two kinds of segment IDS:

- Prefix SID: A segment ID that contains an IP address prefix calculated by an IGP in the service provider core network. Prefix SIDs are globally unique. A prefix segment represents the shortest path (as computed by IGP) to reach a specific prefix; a node segment is a special prefix segment that is bound to the loopback address of a node. It is advertised as an index into the node specific SR Global Block or SRGB.

- Adjacency SID: A segment ID that contains an advertising router's adjacency to a neighbor. An adjacency SID is a link between two routers. Since the adjacency SID is relative to a specific router, it is locally unique.

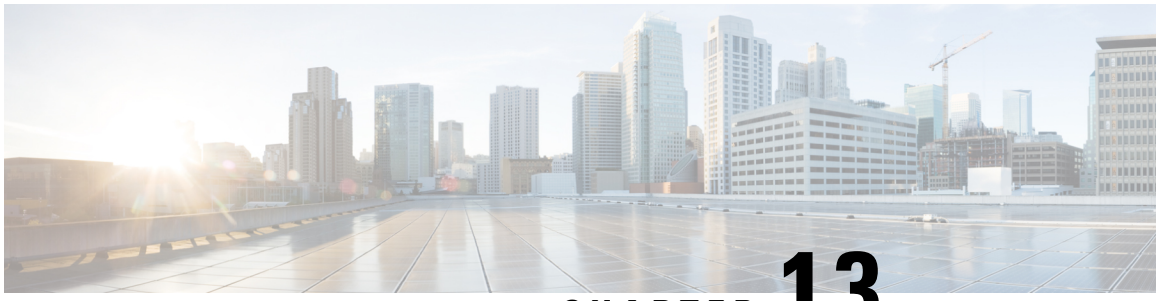
  A node segment can be a multi-hop path while an adjacency segment is a one-hop path.

Segment Routing Global Block

Segment Routing Global Block (SRGB) is the range of labels reserved for segment routing. SRGB is local property of an segment routing node. In MPLS, architecture, SRGB is the set of local labels reserved for global segments. In segment routing, each node can be configured with a different SRGB value and hence the absolute SID value associated to an IGP Prefix Segment can change from node to node.

The SRGB default value is 16000 to 23999. The SRGB can be configured as follows:

```
Router(config)# router isis 1
Router(config-isis)#segment-routing global-block 45000 55000
```



CHAPTER 13

Configure BPDU Transparency with MACsec

This chapter describes the BPDU Transparency with MACsec feature which enables you to create tunnel between a source customer edges (CE) device and a destination CE device and use this tunnel to carry traffic between these two CEs.

- Layer 2 Control Plane Tunneling in MACsec, on page 501
- MACsec and MKA Overview, on page 501
- L2CP Tunneling, on page 502
- L2CP Tunneling in MACsec, on page 502
- Configuration , on page 502

Layer 2 Control Plane Tunneling in MACsec

The punt decision in Layer 2 Control Plane Tunneling depends on the interface that is configured with MACsec. If the main interface is configured with MACsec policy, all the MACsec packets are punted so that MACsec sessions are established between customer edge (CE) device and the provider edge (PE) device. If the main interface is not configured with MACsec, all MACsec packets are tunnelled to the remote CE.

MACsec and MKA Overview

MACsec is an IEEE 802.1AE standards based Layer 2 hop-by-hop encryption that provides data confidentiality and integrity for media access independent protocols.

MACsec, provides MAC-layer encryption over wired networks by using out-of-band methods for encryption keying. The MACsec Key Agreement (MKA) Protocol provides the required session keys and manages the required encryption keys. Only host facing links (links between network access devices and endpoint devices such as a PC or IP phone) can be secured using MACsec.

The 802.1AE encryption with MACsec Key Agreement (MKA) is supported on downlink ports for encryption between the host devices.

MACsec encrypts the entire data except for the Source and Destination MAC addresses of an Ethernet packet.

To provide MACsec services over the WAN or Metro Ethernet, service providers offer Layer 2 transparent services such as E-Line or E-LAN using various transport layer protocols such as Ethernet over Multiprotocol Label Switching (EoMPLS) and L2TPv3.

The packet body in an EAP-over-LAN (EAPOL) Protocol Data Unit (PDU) is referred to as a MACSec Key Agreement PDU (MKPDU). When no MKPDU is received from participants after 3 heartbeats (each heartbeat is of 2 seconds), peers are deleted from the live peer list. For example, if a client disconnects, the participant on the switch continues to operate MKA until 3 heartbeats have elapsed after the last MKPDU is received from the client.

The MKA feature support provides tunneling information such as VLAN tag (802.1Q tag) in the clear so that the service provider can provide service multiplexing such that multiple point to point services can co-exist on a single physical interface and differentiated based on the now visible VLAN ID.

In addition to service multiplexing, VLAN tag in the clear also enables service providers to provide quality of service (QoS) to the encrypted Ethernet packet across the SP network based on the 802.1P (CoS) field that is now visible as part of the 802.1Q tag.

L2CP Tunneling

The Layer 2 control plane is divided into many customer and provider control planes. As defined in the IEEE Standard 802.1Q-2011, an L2CP frame is a frame that contains a destination MAC address that is one among the 32 addresses which are reserved for control protocols. You can transport traffic using VPWS or VPLS service.

L2CP Tunneling in MACsec

The decision to punt depends on the interface that is configured with MACsec. If the interface is configured with MACsec policy, all MACsec packets are punted so that MACsec sessions are established between two customer edge (CE) devices. If the interface is not configured with MACSec, all MACsec packets are tunneled to the remote CE. MACsec cannot be configured on a sub-interface.

When CEs are configured with MACsec and PEs are configured with L2VPN VPWS, all MACsec packets are tunneled through VPWS.

When MACsec is configured on PE on any CE connected interface, all MACsec packets on this interface are punted. These packets are not forwarded to remote CEs. When MACsec is configured on the PE's interface, MACsec session is not established between PE and CE devices.

Configuration

The following sections describes the procedure for configuring BPDU Transparency with MACsec feature.

- Configure an MPLS core
- Configure L2VPN Xconnect
- Configure MACsec on CE device

Configuring L2VPN Xconnect

Configure IPv4 address on an interface connecting to the core.

```
Router# configure
Router(config)# interface tengige 0/1/0/8/2.1
```

```
Router(config-subif)# no shut
Router(config-subif)# ipv4 address 192.0.2.1/24
```

Configure an IPv4 loopback interface.

```
Router# configure
Router(config)#  interface loopback 0
Router(config)# ipv4 address 10.0.0.1/32
```

Configure OSPF as IGP.

```
Router# configure
Router(config)# router ospf 100 area 0
Router(config-ospf-ar)# interface Tengige 0/1/0/8/3
Router(config-ospf-ar-if)# exit
Router(config-ospf-ar)# interface loopback 1
```

Configure MPLS LDP for the physical core interface.

```
Router(config-ospf-ar)# mpls ldp
Router(config-ldp)# interface TenGigE 0/1/8/3
```

Configure IPv4 address on an interface that connects to the core.

```
Router# configure
Router(config)#  router bgp 100
Router(config-bgp)# bgp router-id 10.10.10.1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# update-source loopback 2
Router(config-bgp-nbr)# address-family l2vpn vpls-vpws
Router(config-bgp-nbr-af)# next-hop-self
```

Configure the AC as Layer 2 transport to forward packets to the remote pseudowire.

```
Router# configure
Router(config)# interface TenGigE 0/1/0/8/2.1 l2tranport
Router(config-if)# encap dot1q 1
```

Configure L2VPN Xconnect with a neighbour which is a pseudowire.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group g1
Router(config-l2vpn-xc)# p2p g1
Router(config-l2vpn-xc-p2p)# interface TenGigE 0/1/0/2.1
Router(config-l2vpn-xc-p2p)# neighbor 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)#
```

Configure MACsec on CE device

```
Router# configure
Router(config)# key chain KC1 macsec
Router(config-kc1-MacSec)# key 5010
Router(config-kc1-MacSec-5010)# key-string password
```

```
04795B232C766A6C513A5C4E37582F220F087178116703312446552501 7A0C7101 cryptographic-algorithm
 aes-128-cmac
Router(config-kc1-MacSec-5010)# lifetime 11:08:00 Aug 08 2017 infinite
Router(config-kc1-MacSec-5010)# commit
!
Router# configure
Router(config)# interface HundredGigE 0/0/0/3
Router(config-if)# macsec psk-keychain KC1
Router(config-if)# commit
```

Running Configuration

This section shows BPDU Transparency with MACsec running configuration.

```
/* Configuring MPLS core.*/

/* Configure an IPv4 address on an interface that connects to the MPLS core. */

interface tengige 0/1/0/8/3
 no shut
 ipv4 address 192.0.2.0/24
!

/* Configure an IPv4 loopback interface. */

 interface loop 0
 ipv4 address 10.0.0.1/32

/* Configure OSPF as IGP. */

 router ospf 100 area 0
  interface TenGige 0/1/0/8/3
   interface loop 0
!

/* Configure MPLS LDP for the physical core interface. */

 mpls ldp
  interface TenGige 0/1/0/8/3
  !
!

/* Configuring L2VPN Xconnect. */

/* Configure an IPv4 address on an interface that connects to the MPLS core. */

router bgp 100
 bgp router-id 192.1.2.22
 address-family ipv4 unicast
 exit
 address-family l2vpn vpls-vpws
 neighbor 172.16.0.1
  remote-as 100
  update-source Loopback2
  address-family l2vpn vpls-vpws
   next-hop-self

/* Configure L2VPN Xconnect with a neighbour which is a pseudowire. */

l2vpn
 xconnect group g1
  p2p g1
  interface tengige 0/1/0/8/2.1
```

```
  neighbor 172.16.0.1 pw-id 1

/* Configure MACSec on CE device */
configure
 key chain KC1 macsec
  key 5010
   key-string password 04795B232C766A6C513A5C4E37582F220F087178116703312446552501 7A0C7101
cryptographic-algorithm aes-128-cmac
   lifetime 11:08:00 Aug 08 2017 infinite
commit
!
configure
 interface HundredGigE0/0/0/3
  macsec psk-keychain KC1
commit
end
```

Verification

The show outputs given in the following section display the details of the configuration of the BPDU transparency with MACsec feature, and the status of their configuration.

```
/* Verify if IGP on the core is up. */
Router# show ospf neighbor
Group Wed Aug 16 20:32:33.665 UTC
Indicates MADJ interface
# Indicates Neighbor awaiting BFD session up
Neighbors for OSPF 100
Neighbor ID     Pri   State           Dead Time   Address     Interface
172.16.0.1      1     FULL/DR         00:00:30    10.1.1.2    TenGigE0/1/0/8/0
 Neighbor is up for 06:05:27Total neighbor count: 1

/* Verify if the MPLS core is up. */
Router# show mpls ldp neighbor
Wed Aug 16 20:32:38.851 UTC

Peer LDP Identifier: 172.16.0.1:0
  TCP connection: 172.16.0.1:64932 - 172.31.255.254:646
  Graceful Restart: No
  Session Holdtime: 180 sec
  State: Oper; Msgs sent/rcvd: 487/523; Downstream-Unsolicited
  Up time: 06:05:24
  LDP Discovery Sources:
    IPv4: (2)
      TenGigE0/1/0/8/0
      Targeted Hello (172.31.255.254 -> 172.16.0.1, active)
    IPv6: (0)
  Addresses bound to this peer:
    IPv4: (8)
      10.0.0.1       10.0.0.2         10.0.0.200       172.16.0.1
      192.168.0.1    172.31.255.255   172.16.0.2       10.255.255.254
    IPv6: (0)

/* Verify if the BGP neighbor is up. */
Router# show bgp neighbor 10.10.10.1

Wed Aug 16 20:32:52.578 UTC

BGP neighbor is 10.10.10.1
 Remote AS 15169, local AS 15169, internal link
 Remote router ID 172.31.255.255
```

```
  BGP state = Established, up for 06:03:40
  NSR State: None
  Last read 00:00:34, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:34, attempted 19, written 19
  Second last write 00:01:34, attempted 19, written 19
  Last write before reset 00:00:00, attempted 0, written 0
   ****************
Connections established 1; dropped 0

/* Verify if the BGP neighbor's next-hop information is valid. */
Router# show cef 10.10.10.1
Wed Aug 16 20:33:18.949 UTC
10.10.10.1/32, version 16, internal 0x1000001 0x0 (ptr 0x8e0ef628) [1], 0x0 (0x8e287bc0),
 0xa20 (0x8e9253e0)
 Updated Aug 16 14:27:15.149
 local adjacency 172.16.0.1
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
   via 172.16.0.1/32, TenGigE0/1/0/8/0, 5 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8eb60568 0x8eb60e70]
    next hop 172.16.0.1/32
    local adjacency
     local label 64001      labels imposed {ImplNull}

/* Verify if L2VPN Xconnect is up. */
Router# show l2vpn xconnect

Wed Aug 16 20:47:01.053 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                   Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description            ST
------------------------   ------------------------------  ------------------------------
b1         b1         UP   BE100                  UP       10.10.10.1      1      UP
------------------------------------------------------------------------------------------

/* Note: If L2VPN is down even though the MPLS LDP neighbor is up, check if the AC is down.

To do this, use the show l2vpn xconnect detail command. */


/* Verify if L2VPN Xconnect is up */
Router# show l2vpn xconnect detail

!
 !

AC: Bundle-Ether100, state is up      <<<< This indicates that the AC is up.
    Type Ethernet
    MTU 1500; XC ID 0xa0000002; interworking none
    Statistics:
      packets: received 761470, sent 0
      bytes: received 94326034, sent 0
  PW: neighbor 10.10.10.1, PW ID 1, state is up ( established )
    PW class not set, XC ID 0xc0000001
    Encapsulation MPLS, protocol LDP
    Source address 172.16.0.2
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set

!
```

!