



L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 7.1.x

First Published: 2020-01-01

Last Modified: 2020-08-01

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 NON-INFRINGEMENT 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.

The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on standards documentation, or language that is used by a referenced third-party product.

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)

© 2020 Cisco Systems, Inc. All rights reserved.



CONTENTS

PREFACE

Preface xv

Changes to This Document xv

Obtaining Documentation and Submitting a Service Request xv

CHAPTER 1

New and Changed VPN Features 1

New and Changed VPN Features 1

CHAPTER 2

Configure Gigabit Ethernet for Layer 2 VPNs 3

Introduction to Layer 2 Virtual Private Networks 3

Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces 4

Configure Gigabit Ethernet Interfaces for Layer 2 Transport 5

Configure Link Loss Forwarding for Layer 2 Transport 6

Ethernet Data Plane Loopback 7

Configure Ethernet Data Plane Loopback 8

Running Configuration 9

Verification 10

Related Topics 11

Associated Commands 11

Ethernet Local Management Interface (E-LMI) 11

E-LMI Messaging 12

E-LMI Operation 13

Configure Ethernet Local Management Interface (E-LMI) 13

Running Configuration 15

Verify the Ethernet Local Management Interface (E-LMI) Configuration 16

CHAPTER 3

Configure Layer 2 Access Control Lists 19

- Layer 2 Access Control Lists 19
- Prerequisites for Configuring Layer 2 Access Control Lists 19
- Layer 2 Access Control Lists Feature Highlights 20
- Purpose of Layer 2 Access Control Lists 20
- How a Layer 2 Access Control List Works 20
- Layer 2 Access Control List Process and Rules 20
- Create Layer 2 Access Control List 21
- Restrictions for Configuring Layer 2 Access Control Lists 21
- Configuration 21
 - Running Configuration 22
 - Verification 22

CHAPTER 4 **Configure Virtual LANs in Layer 2 VPNs 23**

- Configure VLAN Subinterfaces 25
- Introduction to Ethernet Flow Point 27
 - Identify Frames of an EFP 28
 - Apply Features 29
 - Define Data-Forwarding Behavior 29
 - Ethernet Flow Points Visibility 30
 - Configuring EFP Visibility 30
- Configure VLAN Header Rewrite 32
 - Rewrite Encapsulation Combinations 35
- Rewrite of Priority Tag 39
 - Configure Rewrite of Priority Tag 40

CHAPTER 5 **L2CP Tunneling MEF 41**

- L2CP Tunneling 41
- L2CP Protocol Support on Cisco NCS 5500 Series Router 42

CHAPTER 6 **Configure Link Bundles for Layer 2 VPNs 45**

- Configure Gigabit Ethernet Link Bundle 45
- Configure VLAN Bundle 48
- References for Configuring Link Bundles 49
 - Characteristics of Link Bundles 50

Methods of Forming Bundles of Ethernet Interfaces 50

Link Aggregation Through LACP 51

CHAPTER 7

Configure Multipoint Layer 2 Services 53

Prerequisites for Implementing Multipoint Layer 2 Services 53

Information About Implementing Multipoint Layer 2 Services 53

Multipoint Layer 2 Services Overview 54

Bridge Domain 54

Bridge Domain and BVI Scale 55

Pseudowires 55

Access Pseudowire 55

Virtual Forwarding Instance 58

VPLS for an MPLS-based Provider Core 58

VPLS for Layer 2 Switching 59

Interoperability Between Cisco IOS XR and Cisco IOS on VPLS LDP Signaling 59

VPLS VFI with BVI as Routed Interface 59

Configure VPLS VFI with BVI as Routed Interface 60

MAC Address-related Parameters 63

MAC Address Flooding 63

MAC Address-based Forwarding 63

MAC Address Source-based Learning 63

MAC Address Aging 63

MAC Address Limit 64

MAC Address Withdrawal 66

How to Implement Services 66

Configuring a Bridge Domain 67

Creating a Bridge Domain 67

Associating Members with a Bridge Domain 68

Configuring Bridge Domain Parameters 69

Disabling a Bridge Domain 71

Flooding Disable 72

Configure Flooding Disable 73

Configuring a Layer 2 Virtual Forwarding Instance 74

Creating the Virtual Forwarding Instance 74

Associating Pseudowires with the Virtual Forwarding Instance	75
Associating a Virtual Forwarding Instance to a Bridge Domain	77
Attaching Pseudowire Classes to Pseudowires	79
Configuring Pseudowires Using Static Labels	80
Disabling a Virtual Forwarding Instance	82
Configuring the MAC Address-related Parameters	84
Configuring the MAC Address Source-based Learning	84
Configuring the MAC Address Limit	86
Configuring the MAC Address Aging	89
Disabling MAC Flush at the Bridge Port Level	91
MAC Address Withdrawal	92
Configure MAC Address Withdrawal	93
Configuration Examples for Multipoint Layer 2 Services	95
Multipoint Layer 2 Services Configuration for Provider Edge-to-Provider Edge: Example	95
Multipoint Layer 2 Services Configuration for Provider Edge-to-Customer Edge: Example	96
Displaying MAC Address Withdrawal Fields: Example	96
Bridging on IOS XR Trunk Interfaces: Example	98
Bridging on Ethernet Flow Points: Example	102
Flow Aware Transport Pseudowire (FAT PW)	106
LDP-Based VPLS and VPWS FAT Pseudowire	107
Configure LDP-Based VPLS and VPWS FAT Pseudowire	108

CHAPTER 8**Configure Point-to-Point Layer 2 Services 113**

Ethernet over MPLS	114
Ethernet Port Mode	114
VLAN Mode	115
Inter-AS Mode	115
QinQ Mode	116
QinAny Mode	117
Configure Local Switching Between Attachment Circuits	117
Configure Static Point-to-Point Connections Using Cross-Connect Circuits	121
Configure Dynamic Point-to-point Cross-Connects	123
Configure Inter-AS	124
Flexible Cross-Connect Service	124

Flexible Cross-Connect Service - Single-Homed	124
Flexible Cross-Connect Service - Multi-Homed	125
Flexible Cross-Connect Service Supported Modes	125
VLAN Unaware	125
Configure Single-Homed Flexible Cross-Connect Service using VLAN Unaware	126
Configure Multi-Homed Flexible Cross-Connect Service using VLAN Unaware	127
VLAN Aware	131
Configure Single-Homed Flexible Cross-Connect using VLAN Aware	131
Configure Multi-Homed Flexible Cross-Connect Service using VLAN Aware	133
Local Switching	137
Configure Multi-Homed Flexible Cross-Connect Service using Local Switching	137
AC-Aware VLAN Bundle	139
Configure Preferred Tunnel Path	140
Multisegment Pseudowire	141
Multisegment Pseudowire Redundancy	143
Configure Multisegment Pseudowire	144
Split Horizon Groups	147
Configure Split Horizon Group 2	149
G.8032 Ethernet Ring Protection	150
Configure G.8032 Ethernet Ring Protection	155
Configure ERP Profile	156
Configuring an ERP Instance	156
Configuring G.8032 Ethernet Ring Protection: Example	158
Configuring Interconnection Node: Example	159
Configuring the Node of an Open Ring: Example	160
Pseudowire Redundancy	161
Configure Pseudowire Redundancy	162
Running Configuration	162
Verification	163
Configure Pseudowire Redundancy	164
Access Pseudowire Redundancy	165
Configure Access Pseudowire Redundancy	165

- Multiple Spanning Tree Protocol 169
 - MSTP Supported Features 169
 - BPDU Guard 170
 - Flush Containment 170
 - Bringup Delay 170
 - Restrictions 171
 - Configure MSTP 171
 - Running Configuration 173
 - Verification 174
- Per-VLAN Rapid Spanning Tree 175
 - Configure PVRST 176
 - Running Configuration for PVRST 177
- Information About Multiple Spanning Tree Protocol 178
 - Spanning Tree Protocol Overview 178
 - STP Protocol Operation 179
 - Topology Changes 179
 - Variants of STP 180
 - MSTP Regions 180
 - MSTP Port Fast 181
 - MSTP Root Guard 182
 - MSTP Topology Change Guard 182

CHAPTER 10

- EVPN Features 183**
 - EVPN Overview 183
 - EVPN Concepts 184
 - EVPN Operation 185
 - EVPN Route Types 187
 - EVPN Timers 188
 - Configure EVPN L2 Bridging Service 190
 - Running Configuration 191
 - EVPN Modes 192
 - EVPN Port-Active Multihoming 192
 - Configure EVPN Port-Active Multihoming 193
 - EVPN MPLS Seamless Integration with VPLS 197

Migrate VPLS Network to EVPN Network through Seamless Integration	197
Configure EVPN on the Existing VPLS Network	198
Configure L2 EVPN Address-Family	199
Configure EVI and Corresponding BGP Route Target under EVPN Configuration Mode	199
Configure EVI under a Bridge Domain	200
EVI Configuration Under L2VPN Bridge-Domain	201
Verify EVPN Configuration	202
Clear Forwarding Table	205
EVPN Features	206
Configure EVPN MAC Address Limit	206
EVPN Software MAC Learning	208
Configure EVPN Software MAC Learning	209
Supported Modes for EVPN Software MAC Learning	210
Single Home Device or Single Home Network Mode	210
Dual Home Device—All-Active Load Balancing Mode	211
Verify EVPN Software MAC Learning	214
EVPN Out of Service	216
Configure EVPN Out of Service	217
EVPN Multiple Services per Ethernet Segment	220
Configure EVPN Multiple Services per Ethernet Segment	220
EVPN Core Isolation Protection	226
Configure EVPN Core Isolation Protection	226
Highest Random Weight Mode for EVPN DF Election	228
Configure Highest Random Weight Mode for EVPN DF Election	229
EVPN Bridging and VPWS Services over BGP-LU Underlay	230
Configure EVPN Bridging and VPWS Services over BGP-LU Underlay	232
EVPN Access-Driven DF Election	242
CFM Support for EVPN	248
CFM on EVPN ELAN	249
Configure CFM on EVPN ELAN	249
EVPN Routing Policy	252
EVPN Route Types	252
EVPN RPL Attribute	257
EVPN RPL Attribute Set	259

Configure EVPN RPL Feature	260
Running Configuration	261

CHAPTER 11

Configure EVPN IRB, Distributed Anycast Gateway and E-tree	269
EVPN IRB	269
EVPN Single-Homing Access Gateway	271
Configure GRT example	272
EVPN Multihoming All-Active	273
EVPN Single-Active Multihoming for Anycast Gateway IRB	273
Configure EVPN Single-Active Multihoming	274
Configure EVPN IRB with Host Routing	274
Configure EVPN Ethernet Segment	275
Configure EVPN Service Instance (EVI) Parameters	276
Configure Layer 2 Interface	276
Configure a Bridge Domain	276
Configure VRF	277
Enable Auto-BGP RT with Manual ESI Configuration	277
Supported EVPN IRB Scenarios	277
Distributed Anycast Gateway	278
EVPN IRB with All-Active Multi-Homing without Subnet Stretch or Host-Routing across the Fabric	278
EVPN IRB with All-Active Multihoming with Subnet Stretch or Host-Routing across the Fabric	279
MAC and IP Unicast Control Plane	280
Intra-subnet Unicast Data Plane	281
Inter-subnet Unicast Data Plane	281
VM Mobility Support	281
MAC and MAC-IP Sequence Numbers	281
Synchronized MAC and MAC-IP Sequence Numbers	281
Local Sequence Number Updates	282
Best Route Selection after Host Movement	282
Stale Route Deletion after a Host Movement	282
Host Movement Detection through GARP	282
Host Move Detection with Silent Host	282
Host Move Detection without GARP with Data Packet	282

Duplicate MAC Detection	282
Configuring EVPN IRB	283
Running Configuration for EVPN IRB	284
Verify EVPN IRB	285
EVPN IPv6 Hosts with Mobility	295
Configure EVPN IPv6 Hosts with Mobility	296
Duplicate IP Address Detection	306
Configure Duplicate IP Address Detection	307
Configuration Example	307
Running Configuration	308
Verification	308
EVPN Automatic Unfreezing of MAC and IP Addresses	309
EVPN E-Tree	309
Configure EVPN E-Tree	313
Configuration Example	313
Running Configuration	314
Verification	316
DHCPv4 Relay on IRB	318
Configure DHCPv4 Relay on IRB	323
Configuration Example	323
Running Configuration	324
DHCPv4 Relay Synchronization for All-Active Multihoming	325
DHCPv6 Relay IAPD on IRB	326
Configure DHCPv6 Relay IAPD on IRB	327
Configuration Example	327
Running Configuration	328
DHCPv6 PD Synchronization for All-Active Multihoming using Session Redundancy	329
Configure DHCPv6 PD Synchronization	330
Configuration Example	330
Running Configuration	330
IAPD Route Distribution and Withdrawal in DHCPv6 Relay	332
CHAPTER 12	EVPN Virtual Private Wire Service (VPWS) 333
	EVPN-VPWS Single Homed 333

- Configure EVPN-VPWS Single Homed 334
 - Running Configuration 334
- EVPN-VPWS Multi-Homed 335
 - Configure EVPN-VPWS Multi-Homed 335
 - Running Configuration 337
- Flow Label Support for EVPN VPWS 338
 - Configure Flow Label for EVPN VPWS 339

CHAPTER 13

L2VPN Services over Segment Routing for Traffic Engineering Policy 341

- L2VPN Preferred path 342
- EVPN VPWS Preferred Path over SR-TE Policy 342
 - Configure EVPN VPWS Preferred Path over SR-TE Policy 343
 - Configure Prefix-SID in ISIS 343
 - Configure Adjacency-SID in ISIS 345
 - Configure Segment-list 347
 - Configure SR-TE Policy 348
 - Configure EVPN VPWS over SR-TE Policy 349
 - Running Configuration 349
 - Verify EVPN VPWS Preferred Path over SR-TE Policy Configuration 354
 - Associated Commands 355
 - Related Topics 355
 - L2VPN VPWS Preferred Path over SR-TE Policy 355
 - Configure L2VPN VPWS Preferred Path over SR-TE Policy 355
 - Configure Prefix-SID in IS-IS 356
 - Configure Adjacency-SID in IS-IS 357
 - Configure Segment-list 359
 - Configure SR-TE Policy 360
 - Configure VPWS over SR-TE Policy 361
 - Running Configuration 362
 - Verify L2VPN VPWS Preferred Path over SR-TE Policy Configuration 365
 - Associated Commands 367
 - Related Topics 368
 - EVPN VPWS On-Demand Next Hop with SR-TE 368
 - Configure EVPN VPWS On Demand Next Hop with SR-TE 369

Topology	369
Configure Prefix-SID in ISIS	369
Configure SR-TE	371
Configure PCE and PCC	372
Configure SR Color	372
Configure EVPN Route Policy	373
Configure BGP	374
Configure EVPN VPWS	374
Configure Flexible Cross-connect Service (FXC) VLAN-unaware	375
Running Configuration	375
Related Topics	382
Overview of Segment Routing	382
How Segment Routing Works	383
Segment Routing Global Block	384

CHAPTER 14**Configure BPDU Transparency with MACsec 385**

Layer 2 Control Plane Tunneling in MACsec	385
MACsec and MKA Overview	385
L2CP Tunneling	386
L2CP Tunneling in MACsec	386
Configuration	386
Running Configuration	388
Verification	389

CHAPTER 15**References 393**

Gigabit Ethernet Protocol Standards	393
Carrier Ethernet Model References	393
Default Configuration Values for Gigabit Ethernet and 10-Gigabit Ethernet	395
References for Configuring Link Bundles	396
Characteristics of Link Bundles	396
Methods of Forming Bundles of Ethernet Interfaces	397
Link Aggregation Through LACP	397



Preface



Note This product has reached end-of-life status. For more information, see the [End-of-Life and End-of-Sale Notices](#).

This preface contains these sections:

- [Changes to This Document, on page xv](#)
- [Obtaining Documentation and Submitting a Service Request, on page xv](#)

Changes to This Document

This table lists the technical changes made to this document since it was first released.

Table 1: Changes to This Document

Date	Summary
January 2020	Initial release of this document.
August 2020	Republished for Release 7.1.2.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly What's New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:

<http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html>

Subscribe to the What's New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.



CHAPTER 1

New and Changed VPN Features

This table summarizes the new and changed feature information for the L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, and tells you where they are documented.

- [New and Changed VPN Features, on page 1](#)

New and Changed VPN Features

Table 2: VPN Features Added or Modified in IOS XR Release 7.1.x

Feature	Description	Changed in Release	Where Documented
EVPN Bridging and VPWS Services over BGP-LU Underlay	This feature was introduced.	Release 7.1.1	EVPN Bridging and VPWS Services over BGP-LU Underlay, on page 230
EVPN Single-Active Multi-Homing for Anycast-GW IRB	This feature was introduced.	Release 7.1.1	EVPN Single-Active Multihoming for Anycast Gateway IRB, on page 273
Flooding Disable	This feature was introduced.	Release 7.1.1	Flooding Disable, on page 72
Highest Random Weight Mode for EVPN DF Election	This feature was introduced.	Release 7.1.1	Highest Random Weight Mode for EVPN DF Election, on page 228
Multisegment Pseudowire	This feature was introduced.	Release 7.1.1	Multisegment Pseudowire, on page 141
Multiple Spanning Tree Protocol	This feature was introduced.	Release 7.1.1	Multiple Spanning Tree Protocol, on page 169
Per-VLAN Rapid Spanning Tree	This feature was introduced.	Release 7.1.1	Per-VLAN Rapid Spanning Tree, on page 175

Feature	Description	Changed in Release	Where Documented
Rewrite of Priority Tag	This feature was introduced.	Release 7.1.1	Rewrite of Priority Tag, on page 39
VPLS VFI with BVI as Routed Interface	This feature was introduced.	Release 7.1.1	VPLS VFI with BVI as Routed Interface, on page 59
Access Pseudowire Redundancy	This feature was introduced.	Release 7.1.2	Access Pseudowire Redundancy, on page 165
EVPN Port-Active Multihoming	This feature was introduced.	Release 7.1.2	EVPN Port-Active Multihoming, on page 192



CHAPTER 2

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 3](#)
- [Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces, on page 4](#)
- [Configure Gigabit Ethernet Interfaces for Layer 2 Transport, on page 5](#)
- [Configure Link Loss Forwarding for Layer 2 Transport, on page 6](#)
- [Ethernet Data Plane Loopback, on page 7](#)
- [Ethernet Local Management Interface \(E-LMI\), on page 11](#)
- [E-LMI Messaging, on page 12](#)
- [E-LMI Operation, on page 13](#)
- [Configure Ethernet Local Management Interface \(E-LMI\) , on page 13](#)

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.

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

```

/* Enter the interface configuration mode */
Router# configure
Router(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 */

Router(config-if)# dot1q tunneling ethertype 0x9100

/* Configure Layer 2 transport on the interface, and commit your configuration */
Router(config-if)# l2transport
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# commit

```

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 \(Ethernet\)](#)

Configure Link Loss Forwarding for Layer 2 Transport

Link Loss Forwarding (LLF) is supported on Cisco router. The LLF is used to avoid any packet loss and trigger the network convergence through alternate links.

LLF sends signals across the PW to the neighbouring device to bring the PW and far-end AC down if the local AC goes down. The LLF feature supports the **l2transport propagate remote-status** command used to propagate Layer 2 transport events.

LLF is supported for TenGigE and GigE interfaces and not supported on the Bundle interfaces.



Note

- Link Loss Forwarding (LLF) does not function on a 1GE copper SFP, irrespective of whether auto-negotiation is enabled or disabled.
- LLF does not function on a 1 GE fiber SFP, when auto-negotiation is enabled. LLF functions only when auto-negotiation is disabled on the 1 GE fiber SFP.
- Tx power level does not change to -40dBm, once the interface is in operational DOWN status due to LLF.

Running Configuration

```
/* Configuring propagation remote-status */
interface TenGigE 0/0/0/5
  l2transport
    propagate remote-status
  !
!
```

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:

- 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.
- 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 MAC address.
- External and internal Ethernet data plane loopback is not supported over 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.

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 7](#)

Associated Commands

- ethernet loopback
- show ethernet loopback

Related Topics

- [Ethernet Data Plane Loopback, on page 7](#)

Associated Commands

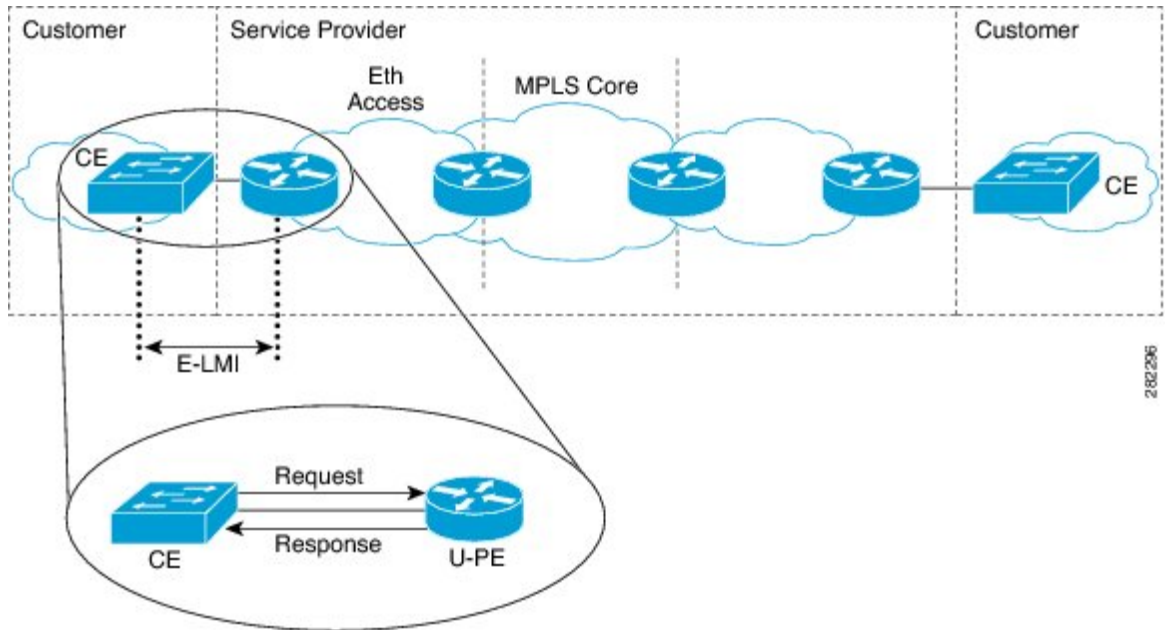
- ethernet loopback
- show ethernet loopback

Ethernet Local Management Interface (E-LMI)

The Cisco NCS 5500 Series Router supports the Ethernet Local Management Interface (E-LMI) protocol as defined by the *Metro Ethernet Forum, Technical Specification MEF 16, Ethernet Local Management Interface (E-LMI), January 2006* standard.

E-LMI runs on the link between the customer-edge (CE) device and the provider-edge (PE) device, or User Network Interface (UNI), and provides a way for the CE device to auto-configure or monitor the services offered by the PE device (see this figure).

Figure 1: E-LMI Communication on CE-to-PE Link



E-LMI is an asymmetric protocol whose basic operation involves the User-facing PE (uPE) device providing connectivity status and configuration parameters to the CE using STATUS messages in response to STATUS ENQUIRY messages sent by the CE to the uPE.

E-LMI Messaging

The E-LMI protocol as defined by the MEF 16 standard, defines the use of only two message types—STATUS ENQUIRY and STATUS.

These E-LMI messages consist of required and optional fields called information elements, and all information elements are associated with assigned identifiers. All messages contain the Protocol Version, Message Type, and Report Type information elements, followed by optional information elements and sub-information elements.

E-LMI messages are encapsulated in 46- to 1500-byte Ethernet frames, which are based on the IEEE 802.3 untagged MAC-frame format. E-LMI frames consist of the following fields:

- Destination address (6 bytes)—Uses a standard MAC address of 01:80:C2:00:00:07.
- Source address (6 bytes)—MAC address of the sending device or port.
- E-LMI Ethertype (2 bytes)—Uses 88-EE.
- E-LMI PDU (46–1500 bytes)—Data plus 0x00 padding as needed to fulfill minimum 46-byte length.
- CRC (4 bytes)—Cyclic Redundancy Check for error detection.

For more details about E-LMI messages and their supported information elements, refer to the Metro Ethernet Forum, Technical Specification MEF 16, Ethernet Local Management Interface (E-LMI), January 2006.

E-LMI Operation

The basic operation of E-LMI consists of a CE device sending periodic STATUS ENQUIRY messages to the PE device, followed by mandatory STATUS message responses by the PE device that contain the requested information. Sequence numbers are used to correlate STATUS ENQUIRY and STATUS messages between the CE and PE.

The CE sends the following two forms of STATUS ENQUIRY messages called Report Types:

- E-LMI Check—Verifies a Data Instance (DI) number with the PE to confirm that the CE has the latest E-LMI information.
- Full Status—Requests information from the PE about the UNI and all EVCs.

The CE device uses a polling timer to track sending of STATUS ENQUIRY messages, while the PE device can optionally use a Polling Verification Timer (PVT), which specifies the allowable time between transmission of the PE's STATUS message and receipt of a STATUS ENQUIRY from the CE device before recording an error.

In addition to the periodic STATUS ENQUIRY/STATUS message sequence for the exchange of E-LMI information, the PE device also can send asynchronous STATUS messages to the CE device to communicate changes in EVC status as soon as they occur and without any prompt by the CE device to send that information.

Both the CE and PE devices use a status counter (N393) to determine the local operational status of E-LMI by tracking consecutive errors received before declaring a change in E-LMI protocol status.

Configure Ethernet Local Management Interface (E-LMI)

Before you configure E-LMI on the router, be sure that you complete the following requirements:

- Identify the local and remote UNIs in your network where you want to run E-LMI, and define a naming convention for them.
- Enable E-LMI on the corresponding CE interface link on a device that supports E-LMI CE operation.

E-LMI is not supported on physical sub-interfaces and bundle main and sub- interfaces. E-LMI is configurable on Ethernet physical interfaces only.

In order to ensure the correct interaction between the CE and the PE, each device has two configurable parameters. The CE uses a Polling Timer (PT) and a Polling Counter; the PE uses a Polling Verification Timer (PVT) and a Status Counter.

To configure Ethernet LMI, complete the following tasks:

- Configure EVCs for E-LMI (required)
- Configure Ethernet CFM for E-LMI (required)
- Enable E-LMI on the Physical Interface (required)
- Configure the Polling Verification Timer (optional)
- Configure the Status Counter (optional)

```

/* Configure EVCs for E-LMI/

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router (config)# interface TenGigE0/3/0/9/1.1 l2transport
RP/0/RSP0/CPU0:router (config-subif)# encapsulation dot1q 1
RP/0/RSP0/CPU0:router (config-subif)# xconnect group evpn
RP/0/RSP0/CPU0:router (config)# l2vpn
RP/0/RSP0/CPU0:router (config-l2vpn)# xconnect group evpn
RP/0/RSP0/CPU0:router (config-l2vpn-xc)# p2p p1
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# interface TenGigE0/3/0/9/1.1
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 3001 source 1
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)#commit

/* Configure Ethernet CFM for E-LMI */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router (config)#interface TenGigE0/3/0/9/1.1 l2transport
RP/0/RSP0/CPU0:router (config-subif)# encapsulation dot1q 1
RP/0/RSP0/CPU0:router (config-subif)# ethernet cfm
RP/0/RSP0/CPU0:router (config-if-cfm)# mep domain irf_evpn_up service up_mep_evpn_1 mep-id
3001
RP/0/RSP0/CPU0:router (config-if-cfm-mep)#exit
RP/0/RSP0/CPU0:router (config)#ethernet cfm
RP/0/RSP0/CPU0:router (config-cfm)# domain irf_evpn_up level 3 id null
RP/0/RSP0/CPU0:router (config-cfm-dmn)#service up_mep_evpn_1 xconnect group evpn p2p p1 id
number 1
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)# mip auto-create all ccm-learning
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)# continuity-check interval 1m loss-threshold 3
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)#continuity-check archive hold-time 10
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)#mep crosscheck
RP/0/RSP0/CPU0:router (config-cfm-xcheck)# mep-id 1
RP/0/RSP0/CPU0:router (config-cfm-xcheck)#ais transmission interval 1m cos 6
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)#log ais
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)#log continuity-check errors
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)#log crosscheck errors
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)#log continuity-check mep changes
RP/0/RSP0/CPU0:router (config-cfm-dmn-svc)#commit

/* Enable E-LMI on the Physical Interface */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router (config)#interface TenGigE0/3/0/9/1
RP/0/RSP0/CPU0:router (config-if)# ethernet lmi
RP/0/RSP0/CPU0:router (config-if-elmi)#commit

```

```

/* Configure the Polling Verification Timer */

```

The MEF T392 Polling Verification Timer (PVT) specifies the allowable time between transmission of a STATUS message and receipt of a STATUS ENQUIRY from the UNI-C before recording an error. The default value is 15 seconds.

```

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router (config)#interface gigabitethernet 0/0/0/0
RP/0/RSP0/CPU0:router (config-if)# ethernet lmi
RP/0/RSP0/CPU0:router (config-if-elmi)#polling-verification-timer 30
RP/0/RSP0/CPU0:router (config-if-elmi)#commit

```

```

/* Configure the Status Counter */

```

The MEF N393 Status Counter value is used to determine E-LMI operational status by tracking receipt of consecutive good packets or successive expiration of the PVT on packets. The

default counter is four, which means that while the E-LMI protocol is in Down state, four good packets must be received consecutively to change the protocol state to Up, or while the E-LMI protocol is in Up state, four consecutive PVT expirations must occur before the state of the E-LMI protocol is changed to Down on the interface.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)#interface gigabitethernet 0/0/0/0
RP/0/RSP0/CPU0:router(config-if)# ethernet lmi
RP/0/RSP0/CPU0:router(config-if-elmi)#status-counter 5
RP/0/RSP0/CPU0:router(config-if-elmi)#commit
```

Running Configuration

This section shows E-LMI running configuration.

```
/* Configure EVCs for E-LMI */

configure
interface TenGigE0/3/0/9/1.1 l2transport
encapsulation dot1q 1

!

l2vpn
xconnect group evpn
p2p p1
interface TenGigE0/3/0/9/1.1
neighbor evpn evi 1 target 3001 source 1
commit

!

/* Configure Ethernet CFM for E-LMI */

configure
interface TenGigE0/3/0/9/1.1 l2transport
encapsulation dot1q 1
ethernet cfm
mep domain irf_evpn_up service up_mep_evpn_1 mep-id 3001

!

configure
ethernet cfm
domain irf_evpn_up level 3 id null
service up_mep_evpn_1 xconnect group evpn p2p p1 id number 1
mip auto-create all ccm-learning
continuity-check interval 1m loss-threshold 3
continuity-check archive hold-time 10
mep crosscheck
mep-id 1
!
ais transmission interval 1m cos 6
log ais
log continuity-check errors
log crosscheck errors
log continuity-check mep changes

!

/* Enable E-LMI on the Physical Interface */

configure
interface TenGigE0/3/0/9/1
```

```

    ethernet lmi
    !

/* Configure the Polling Verification Timer */

configure
interface gigabitethernet 0/0/0/0
    ethernet lmi
        polling-verification-timer 30
    !

/* Configure the Status Counter */

configure
interface gigabitethernet 0/0/0/0
    ethernet lmi
        status-counter 5
    !

```

Verify the Ethernet Local Management Interface (E-LMI) Configuration

Use the **show ethernet lmi interfaces detail** command to display the values for the Ethernet LMI configuration for a particular interface, or for all interfaces. The following example shows sample output for the command:

```

RP/0/RSP0/CPU0:router# show ethernet lmi interfaces detail

Interface: TenGigE0/3/0/9/1
Ether LMI Link Status: Up
Line Protocol State: Up
MTU: 1514 (1 PDU reqd. for full report)
CE-VLAN/EVC Map Type: Service Multiplexing with no bundling (1 EVC)
Configuration: Status counter 4, Polling Verification Timer 15 seconds
Last Data Instance Sent: 130
Last Sequence Numbers: Sent 179, Received 108

Reliability Errors:
  Status Enq Timeouts           0 Invalid Sequence Number      0
  Invalid Report Type           0

Protocol Errors:
  Malformed PDUs                0 Invalid Protocol Version     0
  Invalid Message Type          0 Out of Sequence IE           0
  Duplicated IE                 0 Mandatory IE Missing         0
  Invalid Mandatory IE          0 Invalid non-Mandatory IE     0
  Unrecognized IE               0 Unexpected IE                 0

Full Status Enq Received 00:03:17 ago  Full Status Sent      00:03:17 ago
PDU Received            00:00:07 ago  PDU Sent              00:00:07 ago
LMI Link Status Changed 01:59:54 ago  Last Protocol Error   never
Counters Cleared        never

Sub-interface: TenGigE0/3/0/9/1.1
VLANs: 1
EVC Status: Active
EVC Type: Point-to-Point
OAM Protocol: CFM

```

```

CFM Domain: irf_evpn_up (level 3)
CFM Service: up_mep_evpn_1

Remote UNI Count: Configured = 1, Active = 1
Remote UNI Id                                     Status
-----
<Remote UNI Reference Id: 1>                     Up

```

Make sure:

- The protocol (Ether LMI Link Status) is 'Up'.
- The output does not have "local UNI (UNI Id)" and also it is in provisioned state.
- The interface (Line Protocol State) is 'Up'.
- The CE-VLAN/EVC Map Type is as expected and shows the correct number of EVCs.
- The error counters are all 0.
- The LMI Link Status Changed timer shows the time since the protocol started.
- The sub-interface name(s) corresponds to the EFP(s) configured.
- The VLANs on each interface are as configured.
- The EVC Status is 'Active'.
- The CFM Domain and CFM Service match the provisioning.
- The Remote UNI Id is as provisioned.

Verify CFM (UP MEP)

```

RP/0/RSP0/CPU0:router# show ethernet cfm peer meps
Flags:
> - Ok                               I - Wrong interval
R - Remote Defect received           V - Wrong level
L - Loop (our MAC received)         T - Timed out
C - Config (our ID received)       M - Missing (cross-check)
X - Cross-connect (wrong MAID)     U - Unexpected (cross-check)
* - Multiple errors received       S - Standby

Domain irf_evpn_up (level 3), Service up_mep_evpn_1
Up MEP on TenGigE0/3/0/9/1.1 MEP-ID 3001
=====
St   ID MAC Address   Port   Up/Downtime   CcmRcvd SeqErr   RDI Error
-----
>   1 008a.964b.6410 Up     00:09:59     12      0      0      0
=====

```

Ensure St is >, which means it is OK(up)

Related Topics

- [Ethernet Local Management Interface \(E-LMI\), on page 11](#)
- [E-LMI Messaging, on page 12](#)

- [E-LMI Messaging, on page 12](#)

Associated Commands

- ethernet lmi
- show ethernet lmi interfaces
- show ethernet cfm peer meps



CHAPTER 3

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 19](#)
- [Prerequisites for Configuring Layer 2 Access Control Lists, on page 19](#)
- [Layer 2 Access Control Lists Feature Highlights, on page 20](#)
- [Purpose of Layer 2 Access Control Lists, on page 20](#)
- [How a Layer 2 Access Control List Works, on page 20](#)
- [Layer 2 Access Control List Process and Rules, on page 20](#)
- [Create Layer 2 Access Control List, on page 21](#)
- [Restrictions for Configuring Layer 2 Access Control Lists, on page 21](#)
- [Configuration, on page 21](#)

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 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 4

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.
- Q-in-Any Attachment Circuit—The Attachment Circuit covers all frames received and sent with a specific outer VLAN tag and any inner VLAN tag, as long as that inner VLAN tag is not Layer 3 terminated. Q-in-Any is an extension to QinQ that uses wildcarding to match any second tag.



Note The Q-in-Any mode is a variation of the basic Dot1Q mode. In Q-in-Any mode, the frames have a basic QinQ encapsulation; however, in Q-in-Any mode the inner tag is not relevant, except for the fact that a few specific inner VLAN tags are siphoned for specific services. For example, a tag may be used to provide L3 services for general internet access.

Each VLAN on a CE-to-PE link can be configured as a separate L2VPN connection (using either VC type 4 or VC type 5).

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 25](#)
- [Introduction to Ethernet Flow Point, on page 27](#)
- [Configure VLAN Header Rewrite, on page 32](#)

- [Rewrite of Priority Tag, on page 39](#)

Configure VLAN Subinterfaces

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*

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*

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.
- For double-tagged packet, the VLAN range is supported only on the inner tag.
- VLANs separated by comma are called a VLAN list. VLAN list isn't supported on the router.
- 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 Network Processor Unit (NPU) isn't supported. A subinterface with a single VLAN tag that falls into a range configured on another subinterface of the same NPU 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 32](#))
- 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
Double tagged frames Double tagged frames can be of the following types: <ul style="list-style-type: none"> • Single range • Range-in-Q • Q-in-Range 	802.1Q (ethertype 0x9100) double tagged frames 802.1ad (ethertype 0x9200) double tagged frames <ul style="list-style-type: none"> • 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.

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

Port                    State
----                    -
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

Port                    State
----                    -
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 symmetric
```

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 symmetric
!
!

/* 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 symmetric
!
!

/* 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](#)

Rewrite Encapsulation Combinations

The following table lists the supported and unsupported rewrite combinations:

Table 3: Rewrite Encapsulation Combinations

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
No rewrite	<ul style="list-style-type: none"> • untagged • default • dot1q range • dot1ad range • dot1q priority tagged • dot1ad priority tagged • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag 	<ul style="list-style-type: none"> • dot1q any • dot1ad any
Pop 1	<ul style="list-style-type: none"> • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag 	<ul style="list-style-type: none"> • untagged • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q priority tagged • dot1ad priority tagged

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
Pop 2	<ul style="list-style-type: none"> • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag 	<ul style="list-style-type: none"> • untagged • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q priority tagged • dot1ad priority tagged • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any
Push 1	<ul style="list-style-type: none"> • untagged • default • dot1q range • dot1ad range • dot1q priority tagged • dot1ad priority tagged • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag 	<ul style="list-style-type: none"> • dot1q any • dot1ad any

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
Push 2	<ul style="list-style-type: none"> • untagged • dot1q priority tagged • dot1ad priority tagged • dot1q • dot1ad 	<ul style="list-style-type: none"> • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag
Translate 1 to 1	<ul style="list-style-type: none"> • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag 	<ul style="list-style-type: none"> • untagged • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q priority tagged • dot1ad priority Tagged • custom 9100/9200 double tag
Translate 1 to 2	<ul style="list-style-type: none"> • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag 	<ul style="list-style-type: none"> untagged Default dot1q range dot1q any dot1ad any dot1ad range dot1q priority tagged dot1ad priority Tagged Custom 9100/9200 double tag

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
Translate 2 to 2	<ul style="list-style-type: none"> • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag 	<ul style="list-style-type: none"> • untagged • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q priority tagged • dot1ad priority Tagged • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any
translate 2-to-1	Not Supported	
dot1ad push 1	Not Supported	
dot1ad push 2	Not Supported	
dot1ad translate 1-to-1	Not Supported	
dot1ad translate 1-to-2	Not Supported	
dot1ad translate 2-to-2	Not Supported	
dot1ad translate 2-to-1	Not Supported	

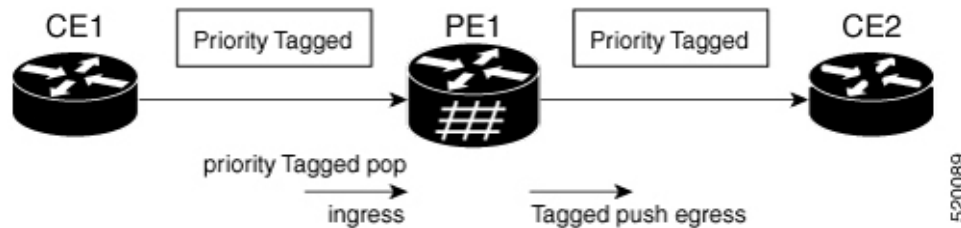
Rewrite of Priority Tag

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 2: 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 39](#)

Associated Commands

- `rewrite ingress tag pop 1 symmetric`



CHAPTER 5

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.

- [L2CP Tunneling, on page 41](#)
- [L2CP Protocol Support on Cisco NCS 5500 Series Router, on page 42](#)

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

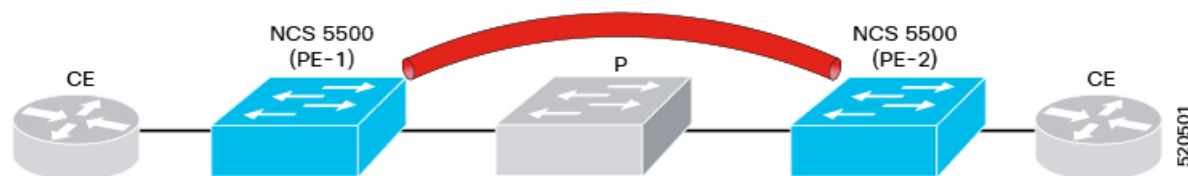
On a subinterface, when control packets such as LLDP and LACP are tunneled, the system tunnels the same control packets to the main interface.

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 3: L2CP Tunneling



L2CP packets are tunneled from NNI to NNI (depicted in red pipe). The Layer 2 traffic is sent through the Cisco NCS 5500 Series 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 5500 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.

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

Protocol	Packet Type	Action
STP	Tagged	Tunneled
VTP	Tagged	Tunneled
BPDU	Tagged	Tunneled
OAM	Tagged	Tunneled
ELMI	Tagged	Tunneled
UDLD	Tagged	Peer

Protocol	Services and Action on Cisco NCS 5700 Series Line Cards							
	EPL1	EPL2	ELAN	E-Tree	EVPL1	EVLAN	EVTREE	Enable on Interface
STP	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Not supported
RSTP	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Not supported
LACP /LAMP	Tunnel	Tunnel	Discard	Discard	Discard	Discard	Discard	Punt
LOAM	Tunnel	Tunnel	Tunnel	Tunnel	NA	NA	NA	Drop
E-LMI	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Punt
LLDP	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Punt
PTP	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Tunnel	Punt
ESMC /SynCE	Tunnel	Tunnel	Tunnel	Tunnel	NA	NA	NA	Not supported
CDP	Tunnel	Tunnel	Tunnel	Tunnel	NA	NA	NA	Punt
MACSEC	Tunnel	Tunnel	Tunnel	Tunnel	Not supported	Not supported	Not supported	Punt-not supported
UDLD	Drop	Drop	Drop	Drop	NA	NA	NA	Punt



Note L2CP protocols over BVI is not supported.



CHAPTER 6

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 45](#)
- [Configure VLAN Bundle, on page 48](#)
- [References for Configuring Link Bundles, on page 49](#)

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 5500 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 64 .
- The maximum number of supported ethernet link bundles is 128 .
- You observe a traffic drop for a few seconds for Layer 2, Layer 3, and BUM traffic when you add bundle members to the existing bundles on the NCS57 line card.

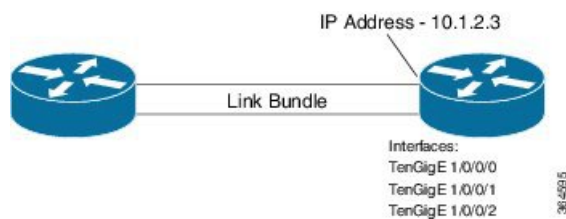
Configuration Example

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 4: 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
Tel1/0/0/0	Local	Active	0x8000, 0x0000	10000000
Link is Active				
Tel1/0/0/1	Local	Active	0x8000, 0x0000	10000000
Link is Active				
Tel1/0/0/2	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 45](#).

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 Etherchannel 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 7

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.

- [Prerequisites for Implementing Multipoint Layer 2 Services, on page 53](#)
- [Information About Implementing Multipoint Layer 2 Services, on page 53](#)
- [How to Implement Services, on page 66](#)
- [MAC Address Withdrawal, on page 92](#)
- [Configuration Examples for Multipoint Layer 2 Services, on page 95](#)
- [Flow Aware Transport Pseudowire \(FAT PW\) , on page 106](#)
- [LDP-Based VPLS and VPWS FAT Pseudowire, on page 107](#)

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.
-

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

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.

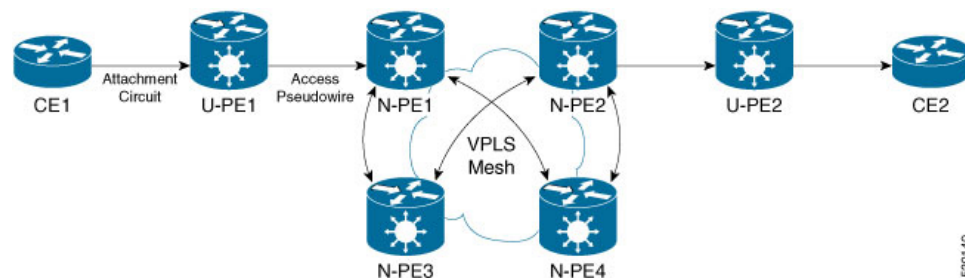
Access Pseudowire

The 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 access pseudowire (PW). This feature prevents signalling overhead and packet replication.

Unlike traditional VPLS where PWs terminate on a physical or logical port, an access PW terminates on N-PE devices. For each VPLS service, create an access PW between U-PEs and N-PEs.

VPLS requires a full mesh of pseudowire (PWs) between L2VPN PEs that participate in the VPLS service. For each VPLS service, PWs must be set up between the PEs. In a full mesh of PWs, the number of PWs increases as the number of PEs increases causing scalability issues. You can decrease the number of PWs with a hierarchy of PEs.

Figure 5: Access Pseudowire



In this topology, a user Provider Edge (U-PE) device has ACs to the CEs. The U-PE device transports the CE traffic over an access PW to a network Provider Edge (N-PE) device. The N-PE is a core VPLS PE connected with other N-PEs in a VPLS mesh. On the N-PE, the access PW coming from the U-PE is much like an AC. The U-PE is not part of the mesh with the other N-PEs. So the N-PE considers the access PW as an AC. The N-PE forwards traffic from that access PW to the core PWs that are part of the VPLS full mesh. Configure the core PWs between N-PEs under a VFI. Apply the split horizon rule to all the core PWs configured under the VFI. Access PWs from U-PEs are not configured under a VFI, so they do not belong to the same Split Horizon Groups (SHGs) as the VFI PWs. Traffic is forwarded from an access PW to a VFI PW and conversely.

You must configure the access pseudowire in a split-horizon group.

Configure Access Pseudowire

Perform this task to configure Access Pseudowire feature.

```
/* Configure U-PE1 */
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
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

/* Configure N-PE1 */
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
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface GigabitEthernet0/1/0/3.2
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 2
Router(config-l2vpn-bg-bd-vfi-pw)#pw-class class1
Router(config-l2vpn-bg-bd-vfi-pw-pw)#commit
```

Running Configuration

This sections shows Access Pseudowire running configuration.

```
/* On U-PE1 */
configure
 interface TenGigE0/1/0/5.2
   encapsulation dot1q 2
   rewrite ingress tag pop 1 symmetric
 !
 l2vpn
  xconnect group XCON1
```

```

p2p xc1
 interface TenGigE0/1/0/5.2
 neighbor 172.16.0.1 pw-id 1
 !
!
-----
/* On N-PE1 */
l2vpn
 router-id 172.16.0.1
 pw-class class1
 encapsulation mpls
 transport-mode ethernet
 !
!
l2vpn
 bridge group bg1
 bridge-domain bd1
 interface GigabitEthernet0/1/0/3.2
 split-horizon group
 !
!
!
vfi vf1
 neighbor 10.0.0.1 pw-id 2
 pw-class class1
 !
!

```

Verification

Verify Access Pseudowire configuration.

```
Router:U-PE1#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 Group	Name	ST	Segment 1 Description	ST	Segment 2 Description	ST
XCON_1	xc1	UP	Te0/1/0/5.2	UP	172.16.0.1 1	UP

```
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)

Related Topics

- [Access Pseudowire, on page 55](#)

Associated Commands

- show l2vpn xconnect group
- show l2vpn bridge-domain

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.

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

- The storm control configuration is supported only on one sub-interface under a main interface, though the system allows you to configure storm control on more than one sub-interface. However, only the first storm control configuration under a main interface takes effect, though the running configuration shows all the storm control configurations that are committed. After reload, any of the storm control configurations may take effect irrespective of the order of configuration.
- The storm control configuration under a bridge domain is not supported.
- 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

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

The Cisco IOS Software encodes the NLRI length in the first 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
```

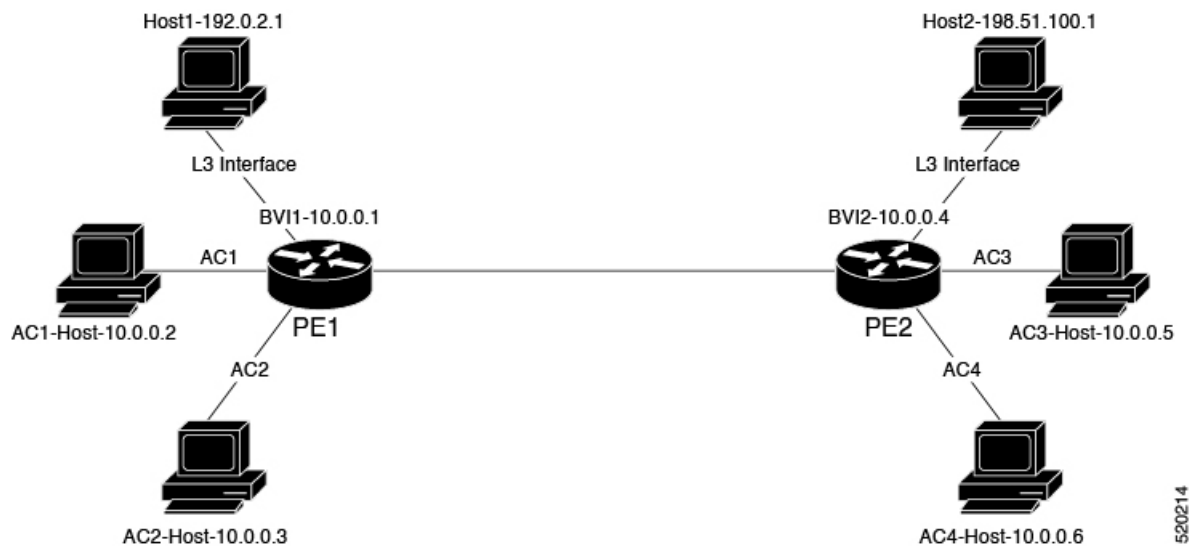
VPLS VFI with BVI as Routed Interface

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 6: 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 routes 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. PE1 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 BFD over BVI.

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 59](#)

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

The MAC address limit is used to limit the number of learned MAC addresses. The default value for the MAC address limit is 64000 for Cisco NCS 5501 and Cisco NCS 5502.

Restrictions

- You can configure up to a maximum of six different mac-limit values under a bridge domain for the following routers and line cards:
 - NCS-55A1-24H
 - NCS-55A1-48Q6H
 - NCS-55A1-36H
 - NCS-55A1-36H-SE
 - NCS-55A2-MOD-HD-S
 - NCS-55A2-MOD-S
 - NCS-5502
 - NCS-5502-SE
 - NCS55-36x100G-S
 - NC55-24H12F-SE
 - NCS55-36x100G-A-SS
- You can configure up to a maximum of 30 different mac-limit values under a bridge domain on routers that have the Cisco NC57 line cards installed.
- For NCS55xx routers and NCS57 line cards, the mac-limit value programmed in the hardware depends on the:
 - Static MAC address configured under the AC for a bridge domain.
 - BVI configured under a bridge domain.

Depending on the BVI or static MAC address configured, new mac-limit profiles are required. The following example shows the different bridge domains with default mac-limit with static MAC address and BVI.

Example 1

In this example, the bridge domain requires a default mac-limit profile. For instance, default mac-limit = X.

```
bridge-domain 1
 interface HundredGigE 0/0/0/10
```

Example 2

In this example, the bridge domain requires a new mac-limit profile with mac-limit = X+1 to accommodate the static BVI MAC address.

```
bridge-domain 2
 interface HundredGigE 0/0/0/11
   routed interface bvi
```

Example 3

In this example, the bridge domain requires a new mac-limit profile with mac-limit = X+2 to accommodate two static MAC addresses configured under the AC.

```
bridge-domain 3
 interface HundredGigE 0/0/0/12
   static-mac-address 0000.1111.2222
   static-mac-address 0000.2222.1111
```

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 destination 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
```

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 .

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **interface** *type interface-path-id*
6. (Optional) **static-mac-address** { *MAC-address* }
7. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **flooding disable**
6. **mtu** *bytes*
7. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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 VFI's that are associated with the bridge domain are disabled. You are still able to attach or detach members to the bridge domain and the VFI's that are associated with the bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **shutdown**
6. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi-name*}
6. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**

3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **vfi** { *vfi name* }
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** { *vfi name* }
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. **static-mac-address** { *MAC-address* }
8. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** { *vfi-name* }
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. **pw-class** { *class-name* }
8. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **vfi** { *vfi-name* }
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. **mpls static label** { **local** *value* } { **remote** *value* }
8. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** { *vfi-name* }
6. **shutdown**
7. Use the **commit** or **end** command.

8. show l2vpn bridge-domain [detail]

DETAILED STEPS

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 cisco  
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domainname*
5. **mac**

6. **learning disable**
7. Use the **commit** or **end** command.
8. **show l2vpn bridge-domain [detail]**

DETAILED STEPS

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 cisco  
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 Limit

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



Note MAC Address Limit action is supported only on the ACs and not on the core pseudowires.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. (Optional) **interface type** *interface_id*
6. **mac**
7. **limit**
8. **maximum** { *value* }
9. **action** { **flood** | **no-flood** | **shutdown** }
10. **notification** { **both** | **none** | **trap** }
11. **mac limit threshold** *80*
12. Use the **commit** or **end** command.
13. **show l2vpn bridge-domain [detail]**

DETAILED STEPS

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 cisco
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 *(Optional)* **interface type interface_id****Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# interface gigabitEthernet 0/2/0/1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Enters the interface configuration mode of the specified interface and adds this interface as the bridge domain member interface.

Note Run this step if you want to configure the MAC address limit only for a specific interface. The further steps show the router prompt displayed when you have skipped this step to configure the MAC address limit at the bridge domain level.

Step 6 **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 7 **limit****Example:**

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

Sets the MAC address limit for action, maximum, and notification and enters L2VPN bridge group bridge domain MAC limit configuration mode.

Step 8 **maximum { value }****Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# maximum 5000
```

Configures the specified action when the number of MAC addresses learned on a bridge is reached.

Step 9 **action { flood | no-flood | shutdown }****Example:**

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

Configures the bridge behavior when the number of learned MAC addresses exceed the MAC limit configured.

Step 10 **notification { both | none | trap }****Example:**

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

Specifies the type of notification that is sent when the number of learned MAC addresses exceeds the configured limit.

Step 11 **mac limit threshold 80****Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn)# mac limit threshold 80
```

Configures the MAC limit threshold. The default is 75% of MAC address limit configured in step 8.

Step 12 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 13 **show l2vpn bridge-domain [detail]**

Example:

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

Displays the details about the MAC address limit.

Configuring the MAC Address Aging

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **mac**
6. **aging**
7. **time** { *seconds* }
8. Use the **commit** or **end** command.
9. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

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 cisco
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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **mac**
6. **port-down flush disable**
7. Use the **commit** or **end** command.

DETAILED STEPS

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 cisco
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

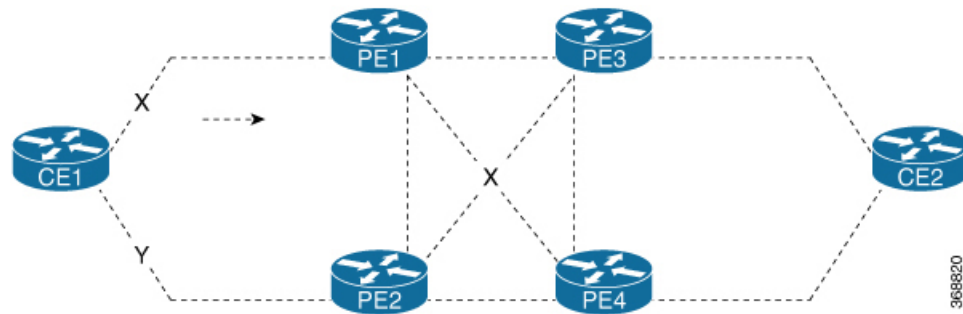
Consider the following topology in which CE1 is dual-homed to PE1 and PE2. The link X actively participates in VPLS while Y is a redundant link. Initially PE1, PE2, PE3, and PE4 learn their MAC forwarding tables that are based on the traffic profile and traffic becomes a known unicast. When the MAC address 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 7: MAC Address Withdrawal



Restrictions

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

- This feature is not supported on Access PW.
- 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 withdrawal on PE2 */
l2vpn
bridge group bg1
  bridge-domain bd1
  mac

```

```

    withdraw disable
  !

```

Verification

Verify MAC address withdrawal configuration.

```

/* Verify if 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

/* 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

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 Cisco NCS 5500 Series Routers 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 Cisco NCS 5500 Series Routers 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:
    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:

SUMMARY STEPS

1. **configure**
2. **interface Bundle-ether10**
3. **l2transport**
4. **interface GigabitEthernet0/2/0/5**
5. **bundle id 10 mode active**
6. **interface GigabitEthernet0/2/0/6**
7. **bundle id 10 mode active**
8. **interface GigabitEthernet0/2/0/0**
9. **l2transport**
10. **interface GigabitEthernet0/2/0/1**
11. **l2transport**
12. **interface TenGigE0/1/0/2**
13. **l2transport**
14. **l2vpn**
15. **bridge group examples**
16. **bridge-domain test-switch**
17. **interface Bundle-ether10**
18. **exit**
19. **interface GigabitEthernet0/2/0/0**
20. **exit**
21. **interface GigabitEthernet0/2/0/1**
22. **exit**
23. **interface TenGigE0/1/0/2**
24. Use the **commit** or **end** command.

DETAILED STEPS

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 Cisco NCS 5500 Series Router 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:

SUMMARY STEPS

1. **configure**
2. **interface Bundle-ether10**
3. **interface Bundle-ether10.999 l2transport**
4. **encapsulation dot1q 999**
5. **interface GigabitEthernet0/6/0/5**
6. **bundle id 10 mode active**
7. **interface GigabitEthernet0/6/0/6**

8. **bundle id 10 mode active**
9. **interface GigabitEthernet0/6/0/7.999 l2transport**
10. **encapsulation dot1q 999**
11. **interface TenGigE0/1/0/2.999 l2transport**
12. **encapsulation dot1q 999**
13. **l2vpn**
14. **bridge group examples**
15. **bridge-domain test-efp**
16. **interface Bundle-ether10.999**
17. **exit**
18. **interface GigabitEthernet0/6/0/7.999**
19. **exit**
20. **interface TenGigE0/1/0/2.999**
21. Use the **commit** or **end** command.

DETAILED STEPS

- 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.
-

Flow Aware Transport Pseudowire (FAT PW)

Routers typically loadbalance traffic based on the lower most label in the label stack which is the same label for all flows on a given pseudowire. This can lead to asymmetric loadbalancing. The flow, in this context, refers to a sequence of packets that have the same source and destination pair. The packets are transported from a source provider edge (PE) to a destination PE.

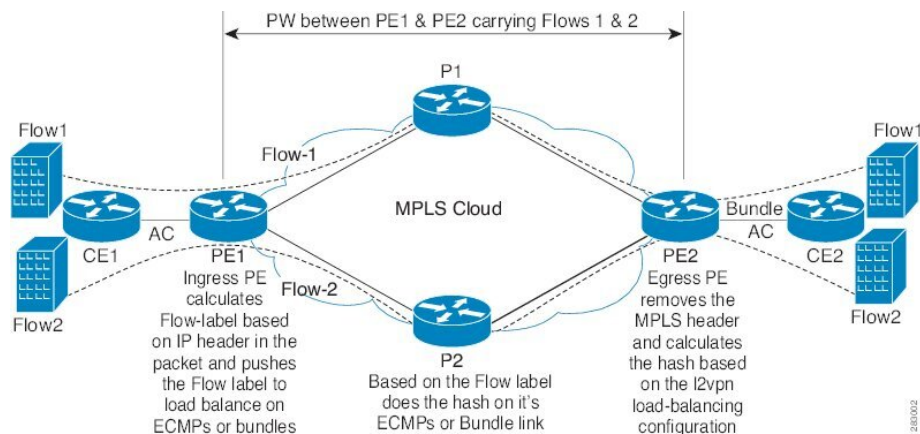
Flow-Aware Transport Pseudowires (FAT PW) provide the capability to identify individual flows within a pseudowire and provide routers the ability to use these flows to loadbalance traffic. FAT PWs are used to loadbalance traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on indivisible packet flows entering a pseudowire; and is inserted as the lower most label in the packet. Routers can use the flow label for loadbalancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core.



Note Based on the number of traffic flows, a SMAC based hashing tuple creates the same flow label or less number of flow labels.

The following figure shows a FAT PW with two flows distributing over ECMPs and bundle links.

Figure 8: FAT PW with two flows distributing over ECMPs and Bundle-Links



An additional label is added to the stack, called the flow label, which contains the flow information of a virtual circuit (VC). 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 and 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.

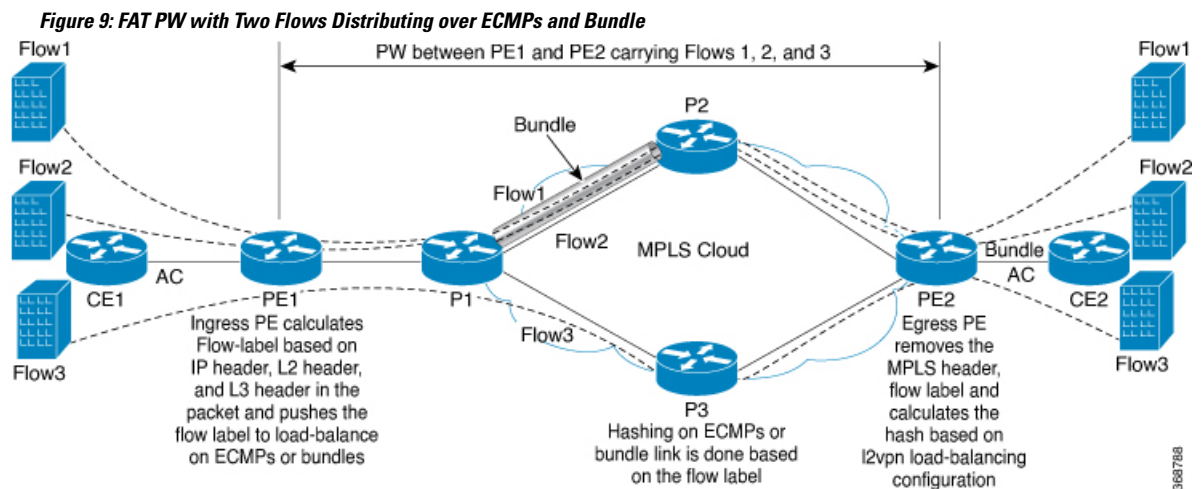
You cannot send MPLS OAM ping traffic over a FAT PW, since there is no flow label support for MPLS OAM.

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.



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)# interface TenGigE0/0/0/5.1
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)# interface TenGigE0/0/0/5.1001
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
  interface TenGigE0/0/0/5.1
  !
  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
  interface interface TenGigE0/0/0/5.1001
  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: TenGigE0/0/0/5.1001, 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      TenGigE0/0/0/5.1001                   HundredGigE0/0/1/0.1001
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 107](#)

Associated Commands

- `show l2vpn xconnect detail`



CHAPTER 8

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 114
- [Configure Local Switching Between Attachment Circuits](#), on page 117
- [Configure Static Point-to-Point Connections Using Cross-Connect Circuits](#), on page 121
- [Configure Dynamic Point-to-point Cross-Connects](#), on page 123
- [Configure Inter-AS](#), on page 124
- [Flexible Cross-Connect Service](#), on page 124
- [Flexible Cross-Connect Service Supported Modes](#), on page 125
- [AC-Aware VLAN Bundle](#), on page 139
- [Configure Preferred Tunnel Path](#), on page 140
- [Multisegment Pseudowire](#), on page 141
- [Configure Multisegment Pseudowire](#), on page 144
- [Split Horizon Groups](#), on page 147
- [G.8032 Ethernet Ring Protection](#), on page 150
- [Configuring G.8032 Ethernet Ring Protection: Example](#), on page 158
- [Pseudowire Redundancy](#) , on page 161
- [Configure Pseudowire Redundancy](#), on page 164
- [Access Pseudowire Redundancy](#), on page 165

Ethernet over MPLS

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, 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.

Table 4: Load Balancing Parameters for Ethernet Frames

Ethernet Frame Type	Parameters for Load Balancing Through LAG Hashing
Ethernet Frame with non-IP payload	<ul style="list-style-type: none"> • Router ID • Input Port • Source Ethernet MAC • Destination Ethernet MAC
Ethernet Frame with IP payload	<ul style="list-style-type: none"> • Router ID • Input Port • Source Ethernet MAC • Destination Ethernet MAC • Source IP Address • Destination IP Address • IP Protocol

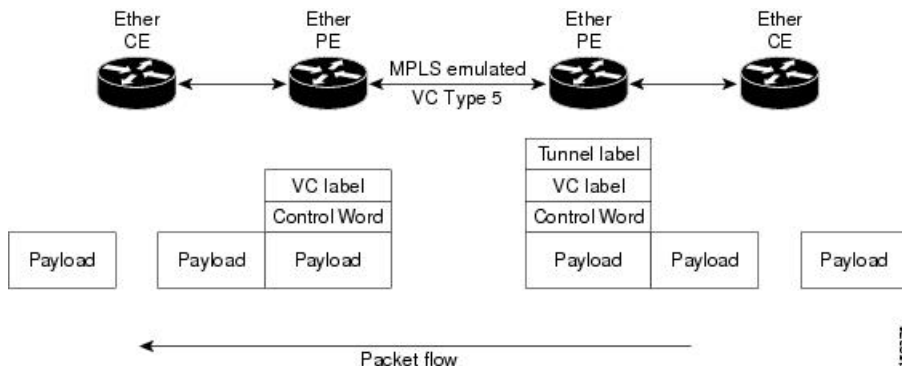
The following sections describe the different modes of implementing EoMPLS.

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 10: Ethernet Port Mode Packet Flow

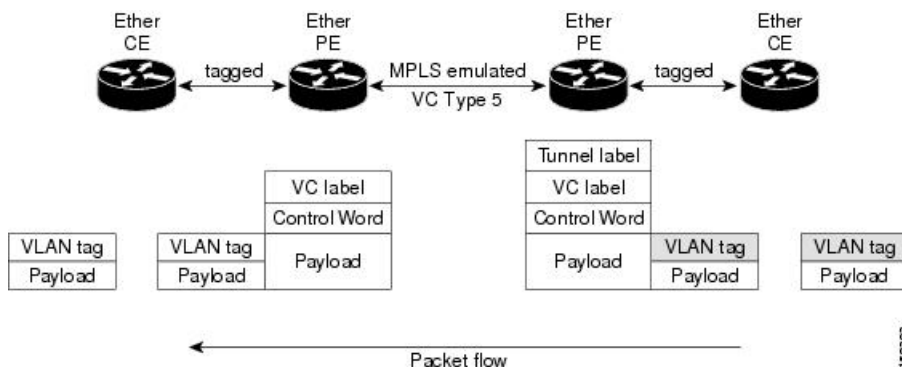


VLAN Mode

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 11: 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.

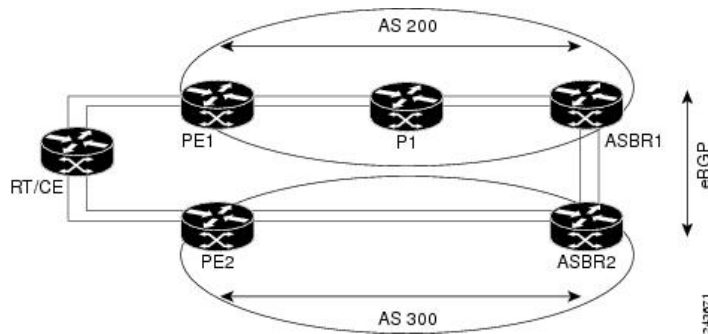
Inter-AS Mode

Inter-AS is a peer-to-peer type model that allows extension of VPNs through multiple provider or multi-domain networks. This lets service providers peer up with one another to offer end-to-end VPN connectivity over extended geographical locations.

EoMPLS support can assume a single AS topology where the pseudowire connecting the PE routers at the two ends of the point-to-point EoMPLS cross-connects resides in the same autonomous system; or multiple AS topologies in which PE routers can reside on two different ASs using iBGP and eBGP peering.

The following figure illustrates MPLS over Inter-AS with a basic double AS topology with iBGP/LDP in each AS.

Figure 12: EoMPLS over Inter-AS: Basic Double AS Topology



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 Router 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.

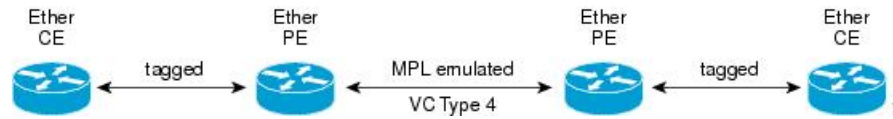
Layer 3 services over QinQ include:

- 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 13: EoMPLS over QinQ Mode



Note EoMPLS does not support pseudowire stitching or multi segments.

QinAny Mode

In the QinAny mode, the service provider VLAN tag is configured on both the ingress and the egress nodes of the provider edge VLAN. QinAny mode is similar to QinQ mode using a Type 5 VC, except that the customer edge VLAN tag is carried in the packet over the pseudowire, as the customer edge VLAN tag is unknown.

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

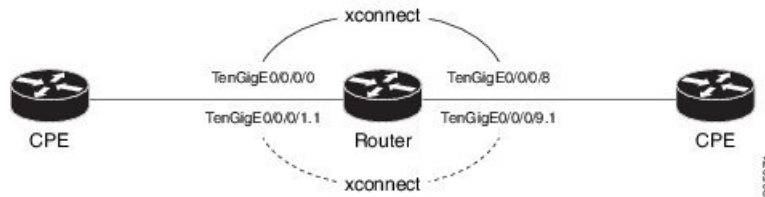
- All sub-interfaces under the given physical port support only two Tag Protocol Identifiers (TPIDs), such as:
 - 0x88a8, 0x8100
 - 0x9100, 0x8100
 - 0x9200, 0x8100
- VLAN and TPID-based ingress packet filtering is not supported.
- Egress TPID rewrite is not supported.

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 14: 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
Group      Name      ST      Segment 1
          Description      ST      Segment 2
          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.

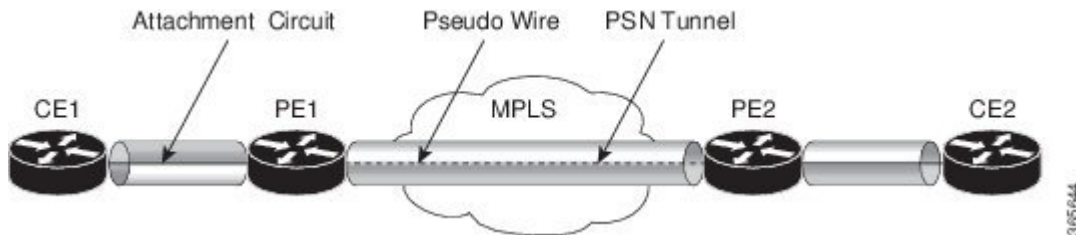


Note Static pseudowire connections do not use LDP for signaling.

Topology

The following topology is used to configure static cross-connect circuits in a Layer 2 VPN.

Figure 15: 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
XCON1	xc1	UP	Gi0/1/0/0.1	UP	10.165.100.151 100

```
/* 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
XCON2	xc1	UP	Gi0/2/0/0.4	UP	10.165.200.254 100

Configure Dynamic Point-to-point Cross-Connects

Perform this task to configure dynamic point-to-point cross-connects.



Note For dynamic cross-connects, LDP must be up and running.

Configuration

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group vlan_grp_1
Router(config-l2vpn-xc)# p2p vlan1
Router(config-l2vpn-xc-p2p)# interface TenGigE 0/0/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor 2.2.1.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
configure
l2vpn
xconnect group vlan_grp_1
p2p vlan1
interface TenGigE 0/0/0/0.1
neighbor 2.2.1.1 pw-id 1
!
```

Configure Inter-AS

The Inter-AS configuration procedure is identical to the L2VPN cross-connect configuration tasks (see [Configure Static Point-to-Point Connections Using Cross-Connect Circuits](#), on page 121 section and [Configure Dynamic Point-to-point Cross-Connects](#), on page 123 section), except that the remote PE IP address used by the cross-connect configuration is now reachable through iBGP peering.



Note You must be knowledgeable about iBGP, eBGP, and ASBR terminology and configurations to complete this configuration.

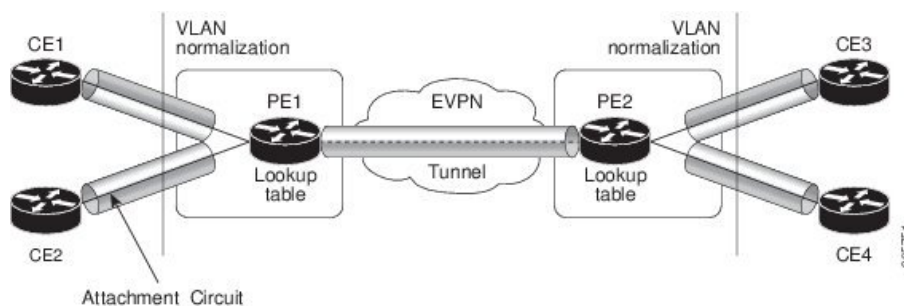
Flexible Cross-Connect Service

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.

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 16: Flexible Cross-Connect Service

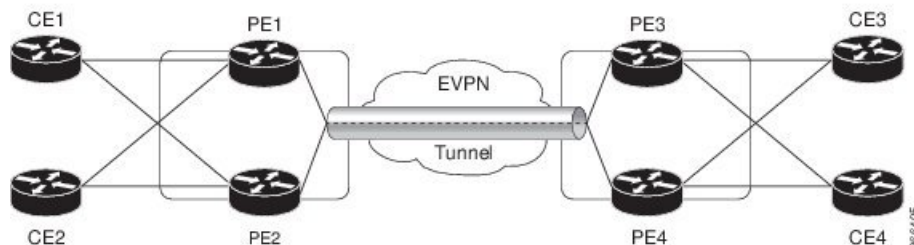


Flexible Cross-Connect Service - Multi-Homed

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 17: 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 symmetric
!

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 symmetric
!

Configure
interface GigabitEthernet 0/0/0/0.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symmetric
!

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-l2vpn-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 symmetric
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 symmetric
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 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

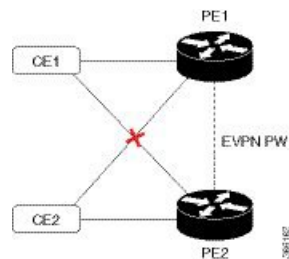
Traffic between the two ACs is locally switched within the PE when two ACs belonging to different Ethernet Segment have the same normalization VLANs. Local switching is supported only on FXC VLAN-aware.

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.

Figure 18: 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
!

```

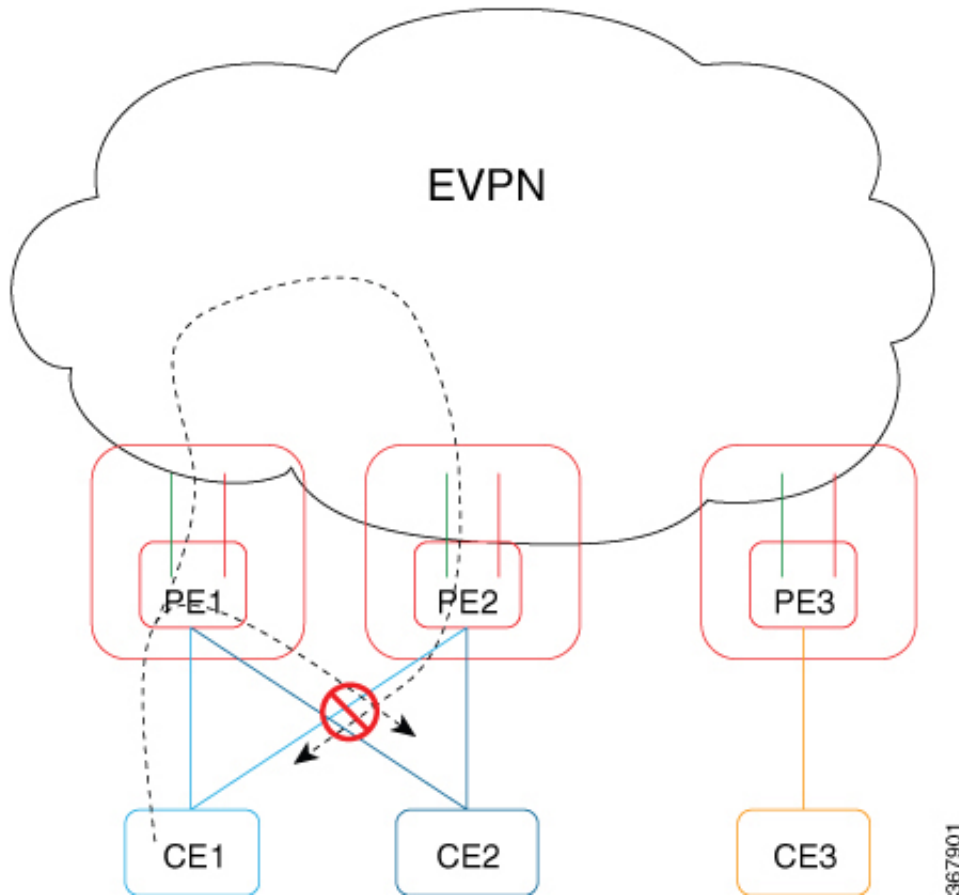
AC-Aware VLAN Bundle

The AC-Aware VLAN Bundle feature allows you to configure more than one subinterface on the same main port in an EVPN enabled bridge domain.

Without this feature, MAC routes identify originating interface using only ESI. When there are multiple subinterfaces with the same ESI, there is no way to distinguish one from the other. Bridge Port (BP) stamping is done with only the EVI and ESI.

With this feature a peering node hosting the advertised ESI performs BP-stamping to a proper local subinterface.

Figure 19: Topology



In this topology, when the traffic from CE1 flows to PE1, PE1 floods the message to the other PEs. As PE2 is directly connected to CE1, a loop is formed between these PEs. To avoid the loop, the traffic from local CE1 subinterface on PE1 to remote CE1 subinterface on PE2 is prevented using ESI filtering.

The AC-Aware VLAN Bundle feature is enabled by default which allows you to configure more than one subinterface on the same main port in an EVPN enabled bridge domain. This feature conforms to *draft-sajassi-bess-evpn-ac-aware-bundling*. Here, the Attachment Circuit ID (AC-ID) is signaled using new EVPN BGP Extended Community.

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.

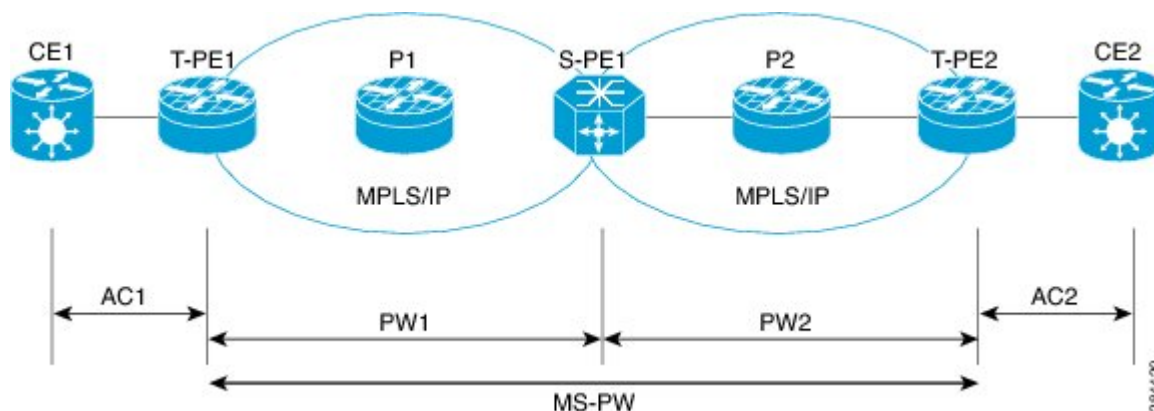
A multisegment pseudowire dynamically builds two discrete pseudowire segments and performs a pseudowire switching to establish a PW control channel between the source and destination PE nodes.

- 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 20: 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

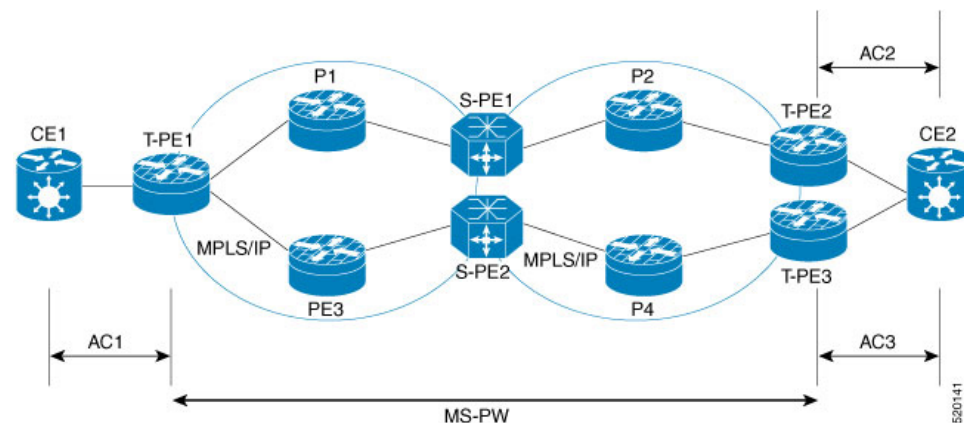
You must consider the following restrictions while configuring the Multisegment Pseudowire feature:

- 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 21: 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

Perform 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)#control-word disable
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-pw)#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)#control-word disable
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)#control-word disable
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
control-word disable
!
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
control-word disable
!
!

/* T-PE2 Configuration */
Configure
l2vpn
pw-class dynamic_mpls
encapsulation mpls
protocol ldp
control-word disable
!

```

```

xconnect group XCON1
  p2p xcl
  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 xconnect
```

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 Description	ST	Segment 2 Description	ST
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 (LSP ping verification)	0x2 (LSP ping verification)
VCCV CC type	0x5 (control word) (TTL expiry)	0x7 (control word) (router alert label) (TTL expiry)

Incoming PW Switching TLV:

IP Address: 70.70.70.70, PW ID: 100

Description: T-PE1 MS-PW to 172.16.0.1 via 192.168.0.1

Outgoing PW Switching TLV:

IP Address: 90.90.90.70, PW ID: 300

Description: T-PE2 MS-PW to 10.0.0.1 via 192.168.0.1

IP Address: 192.168.0.1, PW ID: 100

Description: S-PE1 MS-PW between 10.0.0.1 and 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 141](#)
- [Multisegment Pseudowire Redundancy, on page 143](#)

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 5: 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.

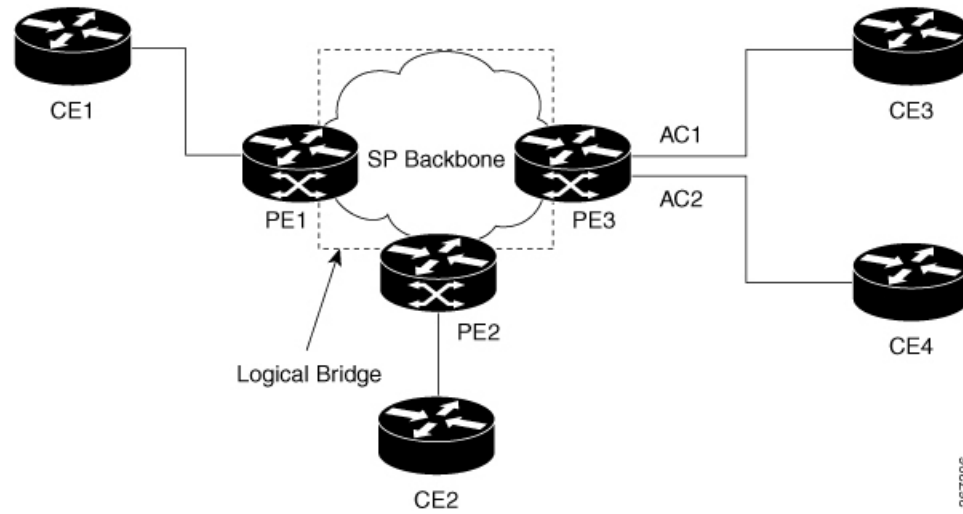
Split Horizon Group 2

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 pseudowire 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 22: 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 group 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 TenGigE0/7/0/18/1
Router(config-l2vpn-bg-bd-ac)#split-horizon group
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#interface TenGigE0/7/0/18/2

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 class1
encapsulation mpls
protocol ldp
ipv4 source 192.168.0.1
!
!
bridge group bg1
bridge-domain bd
!
bridge-domain bd1
interface TenGigE0/7/0/18/1
split-horizon group
!
interface TenGigE0/7/0/18/2
split-horizon group
!
vfi vfil
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:
  Te0/7/0/18/1
, stage: up, Static MAC addresses: 0
  Te0/7/0/18/2, stage: up, Static MAC addresses: 0
List of Access PWs:
List of VFIs:
  VFI vfil (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:

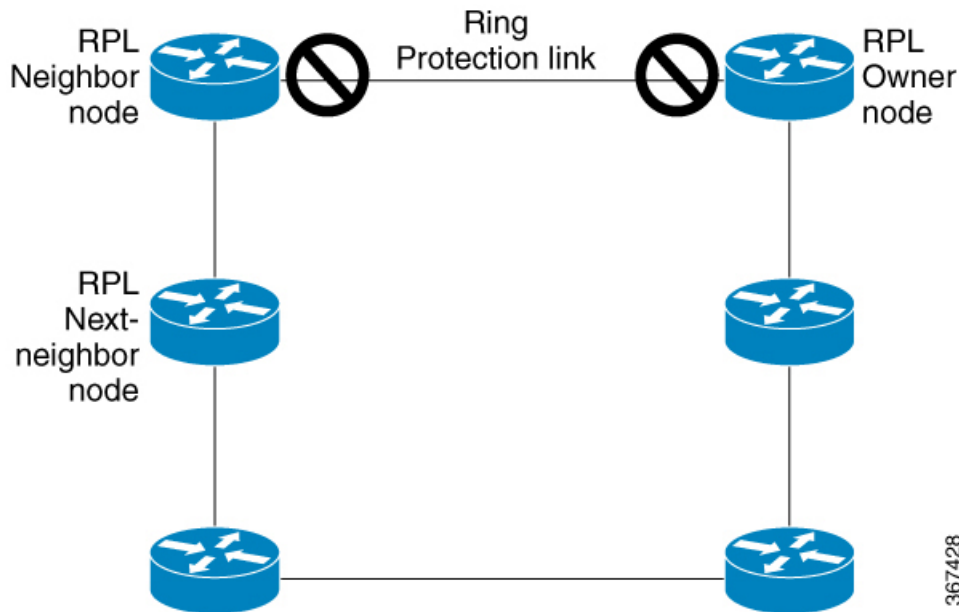
- The principle of loop avoidance.
- The utilization of learning, forwarding, and Filtering Database (FDB) mechanisms.

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 23: 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



Note ERPS Ring Link failures/Force switch/Manual switch events during MAC flush events results into unpredictable convergence.

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.



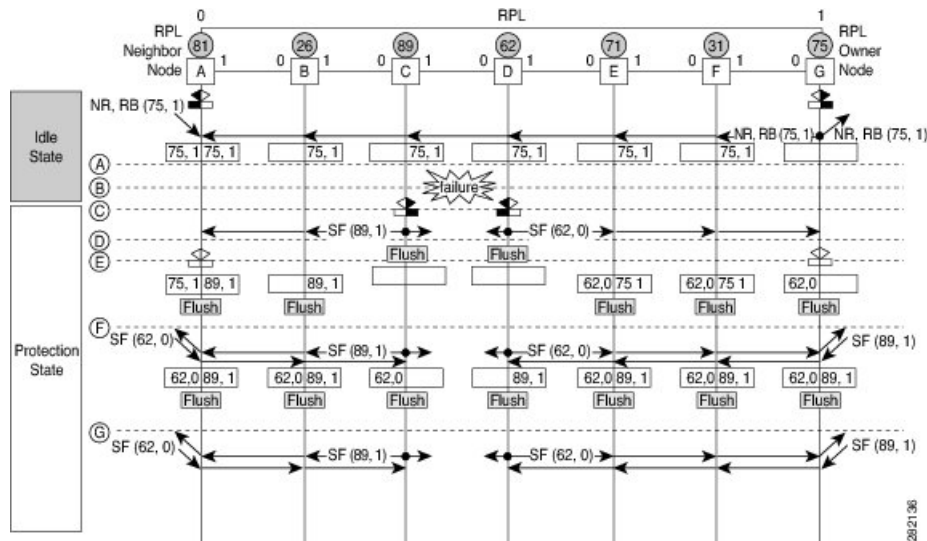
Note Wait-to-Block timer may be shorter than the Wait-to-Restore timer

- 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 24: 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:

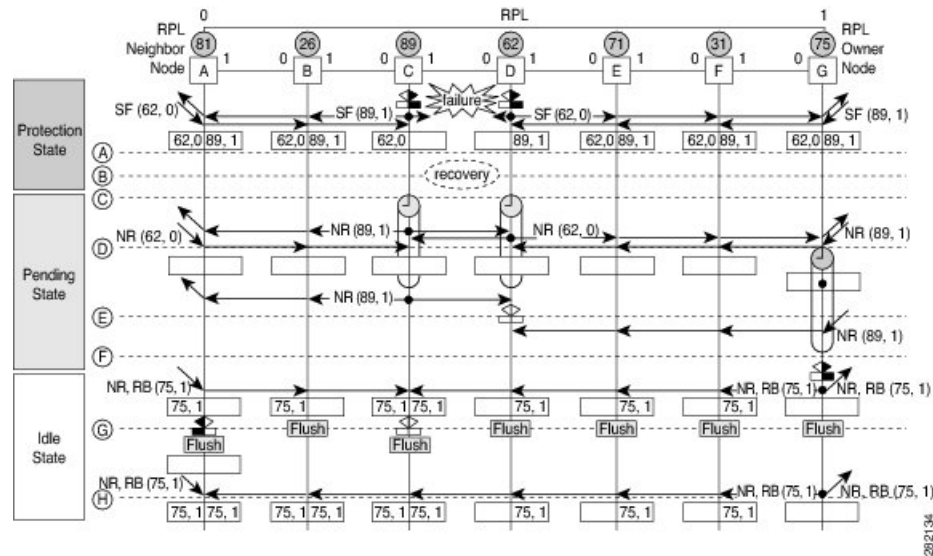
- Message source
- ▶ R-APS channel blocking
- Client channel blocking
- Ⓝ Node ID

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 its 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 25: 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:

- Configure 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 Example

```
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 Configuration

```
configure
Ethernet ring g8032 profile p1
timer wtr 5
non-revertive
!
!
```

Configuring an ERP Instance

Perform this task to configure an ERP instance.

Configuration Example

```
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 TenGigE0/0/0/0.1
Router(config-l2vpn-erp-instance-aps)#port1 interface TenGigE0/0/0/8.1
/* 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 Configuration

```

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 TenGigE0/0/0/0.1
    port1 interface TenGigE0/0/0/8.1
    !
    !
  !

```

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: pl
    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 0/1/0/0
    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 g0/1/0/0
        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 g0/1/0/0.1

# Set up the bridge domains

```

```

bridge group ABC
  bridge-domain BD2
    interface Gig 0/0/0/0.2

    interface Gig 0/1/0/0.2
    interface Gig 0/2/0/0.2

  bridge-domain BD2-APS
    interface Gig 0/0/0/0.1
    interface Gig 0/1/0/0.1

# EFPs configuration
interface Gig 0/0/0/0.1 l2transport
  encapsulation dot1q 5

interface Gig 0/0/0/0.1 l2transport
  encapsulation dot1q 5

interface g0/0/0/0.2 l2transport
  encapsulation dot1q 10-100

interface g 0/1/0/0.2 l2transport
  encapsulation dot1q 10-100

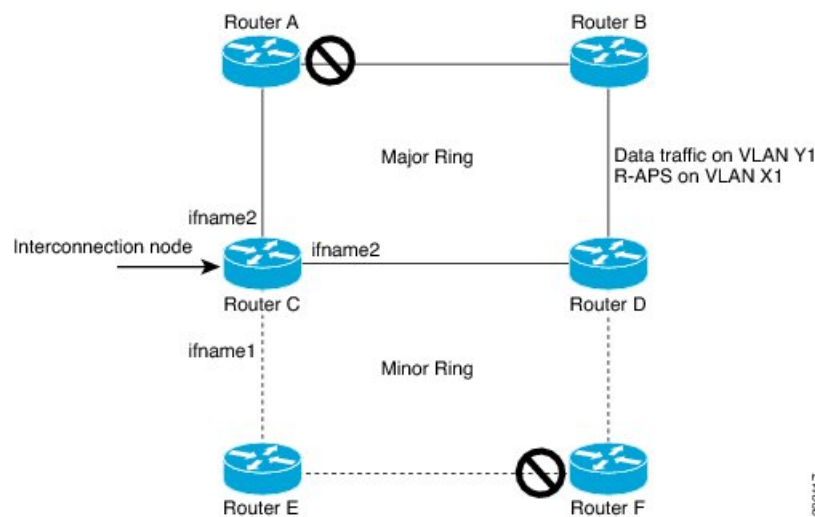
interface g 0/2/0/0.2 l2transport
  encapsulation dot1q 10-100

```

Configuring Interconnection Node: Example

This example shows you how to configure an interconnection node. The following figure illustrates an open ring scenario.

Figure 26: 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

```

```

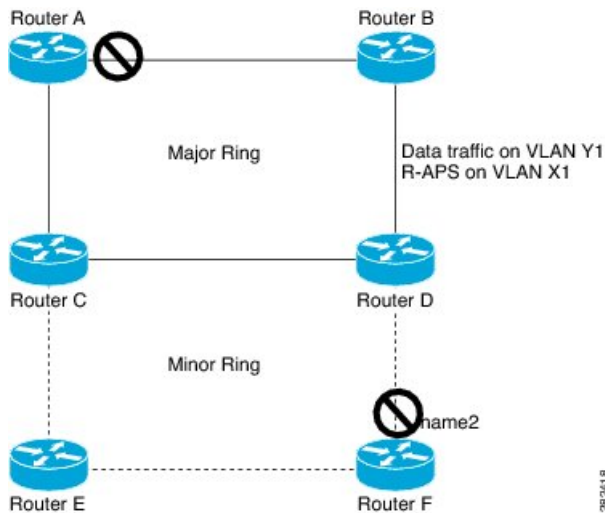
interface Gig 0/0/0/3.10 l2transport
 encapsulation dot1q 6
 l2vpn
 ethernet ring g8032 ring8
   port0 interface Gig 0/0/0/1
   port1 none /* This router is connected to an interconnection node. */
 open-ring
 !
 instance 1
 inclusion-list vlan-ids 1,7-150
 aps-channel
   port0 interface Gig 0/0/0/1.1
   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 27: 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

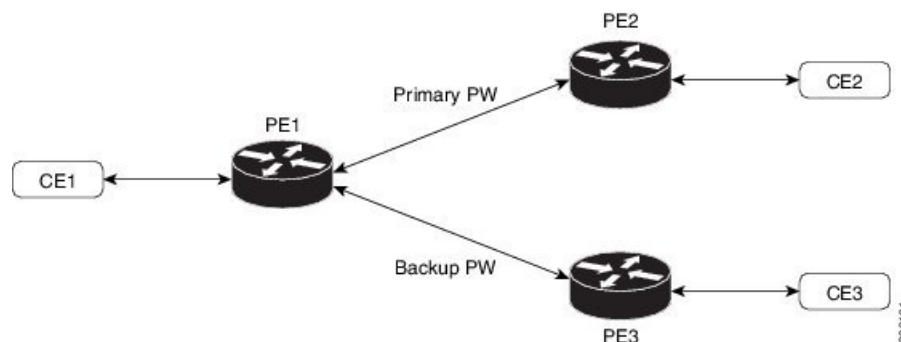
/* Set up the bridge domain
bridge group bg1
  bridge-domain BD2
  bridge-domain BD2-APS /* APS-channel has its own bridge domain */
  interface Gig 0/0/0/1.1
  interface Gig 0/0/0/2.1
  !
/* Data traffic has its own bridge domain */
bridge-domain BD2
  interface Gig 0/0/0/1.10
  interface Gig 0/0/0/2.10

```

Pseudowire Redundancy

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 28: 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 xcl
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/1/0/0.1
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 xcl
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/1/0/0.1
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 xcl
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/1/0/0.1
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 0/1/0/0.1
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 0/1/0/0.1
neighbor ipv4 10.0.0.1 pw-id 1
!

/* On PE3 */
!
l2vpn

```

```
xconnect group XCON1
 p2p XCON1_P2P2
  interface GigabitEthernet 0/1/0/0.1
  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 Group	Name	ST	Segment 1 Description	ST	Segment 2 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 Group	Name	ST	Segment 1 Description	ST	Segment 2 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 Group	Name	ST	Segment 1 Description	ST	Segment 2 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

```

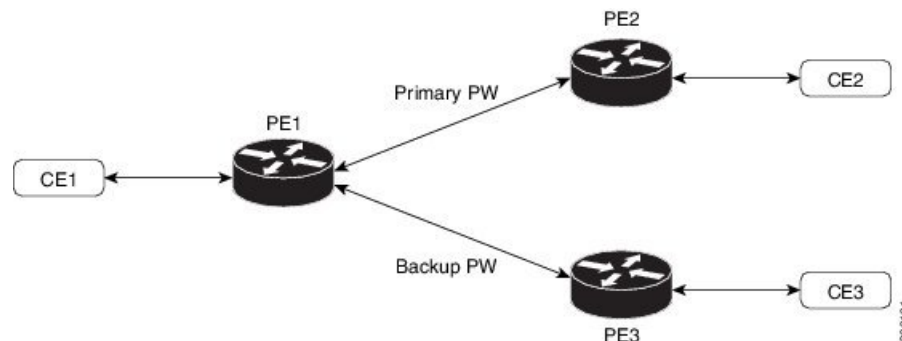
Router# show-running configuration
...
l2vpn
encapsulation mpls
!
xconnect group A
p2p xc1
interface tengige 0/0/0/0.2
neighbor ipv4 10.1.1.2 pw-id 2
pw-class path1
backup neighbor 10.2.2.2 pw-id 5
!
!
...

```

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 29: 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 165](#)

Associated Commands

- show l2vpn bridge-domain



CHAPTER 9

Multiple Spanning Tree Protocol

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
MSTP and PVRST	Release 7.6.1	

- [Multiple Spanning Tree Protocol, on page 169](#)
- [Per-VLAN Rapid Spanning Tree, on page 175](#)
- [Information About Multiple Spanning Tree Protocol, on page 178](#)

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 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-bpdu-guard** command on MST default instance, and if one of the bridge ports receives legacy BPDU, the port enters **error-disable** state.



Note MSTP supports interoperability with RSTP as described in the 802.1Q standard. However, these features do not support interoperability with legacy STP.

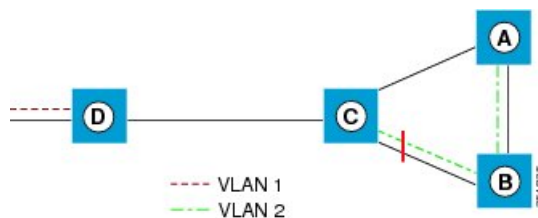
BPDU Guard

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 30: 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 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-bpdu-guard** command on MST default instance, and if one of the bridge ports receives legacy BPDU, the port enters **error-disable** state.

Configure MSTP

By default, STP is disabled on all interfaces. You must enable MSTP on each physical or Ethernet Bundle interface. When you configure MSTP on an interface, all the subinterfaces of that interface are automatically MSTP-enabled.

Perform these tasks to configure MSTP:

- Configure VLAN interfaces
- Configure L2VPN bridge-domains
- Configure MSTP

Configuration Example

```
/* Configure VLAN interfaces */  
Router# configure
```

```

Router(config)# interface TenGigE0/0/0/2.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface TenGigE0/0/0/3.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface TenGigE0/0/0/14.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface TenGigE0/0/0/2.1021 l2transport
Router(config-subif)# encapsulation dot1q 1021
Router(config)# interface TenGigE0/0/0/3.1021 l2transport
Router(config-subif)# encapsulation dot1q 1021
Router(config)# interface TenGigE0/0/0/14.1021 l2transport
Router(config-subif)# encapsulation dot1q 1021
Router(config-subif)# commit

/* Configure L2VPN bridge-domains */
Router# configure
Router(config)# l2vpn bridge group mstp
Router(config-l2vpn-bg)# bridge-domain mstp1001
Router(config-l2vpn-bg-bd)# int TenGigE 0/0/0/2.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int TenGigE 0/0/0/3.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int TenGigE 0/0/0/14.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn-bg)# bridge-domain mstp1021
Router(config-l2vpn-bg-bd)# int TenGigE 0/0/0/2.1021
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int TenGigE 0/0/0/3.1021
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int TenGigE 0/0/0/14.1021
Router(config-l2vpn-bg-bd-ac)# commit

/* Configure MSTP */

Router# configure
Router(config)# spanning-tree mst abc
Router(config-mstp)# name mstp1
Router(config-mstp)# instance 1001
Router(config-mstp-inst)# vlan-ids 1001-1020
Router(config-mstp-inst)# exit
Router(config-mstp)# instance 1021
Router(config-mstp-inst)# vlan-ids 1021-1040
Router(config-mstp-inst)# exit
Router(config-mstp)# int tenGigE 0/0/0/2
Router(config-mstp-if)# exit
Router(config-mstp)# int tenGigE 0/0/0/3
Router(config-mstp-if)# exit
Router(config-mstp)# int tenGigE 0/0/0/14
Router(config-mstp-if)# commit

/* Configure MSTP Parameters */
Router#configure
Router(config)#spanning-tree mst a
Router(config-mstp)#bringup delay for 10 minutes
Router(config-mstp)#flush containment disable
Router(config-mstp)#name m1
Router(config-mstp)#revision 10
Router(config-mstp)#forward-delay 20
Router(config-mstp)#max age 40
Router(config-mstp)#transmit hold-count 8

```



```

Router(config-mstp)#provider-bridge
Router(config-mstp)#instance 101
Router(config-mstp-inst)#priority 8192
Router(config-mstp-inst)#vlan-id 2-1005
Router(config-mstp)#interface FastEthernet 0/0/0/1
Router(config-mstp-if)#instance 101 port-priority 160
Router(config-mstp-if)#portfast bpduguard
Router(config-mstp-if)#commit

```

Running Configuration

This section show MSTP running configuration.

```

!
Configure
/* Configure VLAN interfaces */
interface TenGigE0/0/0/2.1001 l2transport
 encapsulation dot1q 1001
!
interface TenGigE0/0/0/3.1001 l2transport
 encapsulation dot1q 1001
!
interface TenGigE0/0/0/14.1001 l2transport
 encapsulation dot1q 1001

interface TenGigE0/0/0/2.1021 l2transport
 encapsulation dot1q 1021
!
interface TenGigE0/0/0/3.1021
 l2transport
 encapsulation dot1q 1021
!
interface TenGigE0/0/0/14.1021 l2transport
 encapsulation dot1q 1021
!
/* Configure L2VPN Bridge-domains */
l2vpn
bridge group mstp
 bridge-domain mstp1001
   interface TenGigE0/0/0/2.1001
   !
   interface TenGigE0/0/0/3.1001
   !
   interface TenGigE0/0/0/14.1001
   !
bridge-domain mstp1021
   interface TenGigE0/0/0/2.1021
   !
   interface TenGigE0/0/0/3.1021
   !
   interface TenGigE0/0/0/14.1021
!
/* Configure MSTP */
spanning-tree mst abc
 name mstp1
 instance 1001
   vlan-ids 1001-1020
!
 instance 1021
   vlan-ids 1021-1040
!
interface TenGigE0/0/0/2
!

```

```

interface TenGigE0/0/0/3
!
interface TenGigE0/0/0/14

/* Configure MSTP Parameters */
spanning-tree mst a
bringup delay for 10 minutes
flush containment disable
name m1
revision 10
forward-delay 20
max hops 30
transmit hold-count 8
provider-bridge
instance 101
priority 8192
vlan-id 2-1005
interface FastEthernet 0/0/0/1
instance 101 port-priority 160
portfast bpduguard
!
!

```

Verification

Verify the MSTP configuration using the **show spanning-tree mst** command.

```

/* Verify the MSTP configuration */
Router# show spanning-tree mst abc instance 121
Mon Jan 23 12:11:48.591 UTC
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 121:

VLANS Mapped: 121-130

```

Root ID    Priority    32768
Address    dceb.9456.b9d4
This bridge is the root
Int Cost    0
Max Age 20 sec, Forward Delay 15 sec

```

```

Bridge ID  Priority    32768 (priority 32768 sys-id-ext 0)
Address    dceb.9456.b9d4
Max Age 20 sec, Forward Delay 15 sec
Max Hops 20, Transmit Hold count 6

```

Interface	Port ID	Port ID	Role	State	Designated	Port ID
	Pri.Nbr	Cost			Bridge ID	Pri.Nbr
BE1	128.1	10000	DSGN	FWD	32768 dceb.9456.b9d4	128.1
Te0/0/0/1	128.2	2000	DSGN	FWD	32768 dceb.9456.b9d4	128.2
Te0/0/0/16	128.3	2000	DSGN	FWD	32768 dceb.9456.b9d4	128.3
Te0/0/0/17	128.4	2000	DSGN	FWD	32768 dceb.9456.b9d4	128.4

Related Topics

- [Information About Multiple Spanning Tree Protocol, on page 178](#)
- [Multiple Spanning Tree Protocol, on page 169](#)

Associated Commands

- spanning-tree mst
- show spanning-tree mst

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. When you are running Rapid PVST+ you must configure the feature on all VLANs for a particular port.

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.

PVRST has the following characteristics:

- You can configuration Forward Delay and Max Age timers globally and not per VLAN.
- You can configure Hello timer on per port basis and not per VLAN basis. 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.
- PVRST supports 64 VLANs and 512 EFP's per router.

Configure PVRST

Perform this task to configure PVRST per VLAN.

Prerequisites

- Define L2 transport subinterfaces with VLAN encapsulation. PVRST does not support dot1ad encapsulation.
- Configure L2VPN bridge domains under bridge group for every VLAN running spanning tree.
- Configure corresponding l2transport subinterfaces in the bridge-domain.

Configuration Example

```

/* Configure PVRST */
Router# configure
Router(config)# spanning-tree pvrst stp
Router(config-pvrst)# forward-delay 10
Router(config-pvrst)# maximum age 10
Router(config-pvrst)# transmit hold-count 4
Router(config-pvrst)# vlan 200
Router(config-pvrst-vlan)# exit
Router(config-pvrst)# vlan 300
Router(config-pvrst-vlan)# exit
Router(config-pvrst)# 400
Router(config-pvrst-vlan)# exit
Router(config-pvrst)# interface Bundle-Ether1
Router(config-pvrst-if)# exit
Router(config-pvrst)# interface Bundle-Ether2
Router(config-if)# exit
Router(config-pvrst)# interface TenGigE0/0/0/21

/* Configure bridge domain */
Router(config)# l2vpn bridge group pvrst
Router(config-l2vpn-bg)# bridge-domain pvrst1001
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface Bundle-Ether2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/21
Router(config-l2vpn-bg-bd-ac)# commit

```

Running Configuration for PVRST

This section shows PVRST running configuration.

```

/* PVRST Configuration */
configure
spanning-tree pvrst stp
  forward-delay 10
  maximum age 10
  transmit hold-count 4
  vlan 200
  !
  vlan 300
  !
  vlan 400
  !
  interface Bundle-Ether1
  !
  interface Bundle-Ether2
  !
  interface TenGigE0/0/0/21
  !
/* Configure bridge-domain */
l2vpn
bridge group pvrst
  bridge-domain pvrst1001
  interface Bundle-Ether1
  !
  !
  interface Bundle-Ether2
  !
  interface TenGigE0/0/0/14.1001
  !
  interface TenGigE0/0/0/21

```

Verification

Verify the PVRST configuration using the **show spanning-tree pvrst** command.

```

Router# show spanning-tree pvrst stp
Mon Jan 20 22:56:16.242 UTC
Role:  ROOT=Root, DSGN=Designated, ALT=Alternate, BKP=Backup
State: FWD=Forwarding, LRN=Learning, BLK=Blocked
VLAN 200:
  Root ID      Priority      32768
              Address      008a.9610.08d8
              Max Age 20 sec, Forward Delay 15 sec
  Bridge ID    Priority      32768 (priority 32768 sys-id-ext 0)
              Address      00bc.6021.a4d8
              Max Age 10 sec, Forward Delay 10 sec
              Transmit Hold count 4
Interface     Port ID      Role State Designated          Port ID
              Pri.Nbr Cost          Bridge ID            Pri.Nbr
-----
Te0/0/0/14   128.2    2000    DSGN FWD    32768 00bc.6021.a4d8    128.2
Te0/0/0/20   128.3    2000    DSGN FWD    32768 00bc.6021.a4d8    128.3
Te0/0/0/26   128.4    2000    DSGN FWD    32768 00bc.6021.a4d8    128.4
Te0/0/0/27   128.5    2000    DSGN FWD    32768 00bc.6021.a4d8    128.5
Te0/0/0/38   128.6    2000    ALT  BLK    32768 008a.9610.08d8    128.4
Te0/0/0/6    128.1    2000    ROOT FWD    32768 008a.9610.08d8    128.2
-----
VLAN 300:

```

```

Root ID      Priority      32768
            Address      008a.9610.08d8
            Max Age 20 sec, Forward Delay 15 sec
Bridge ID    Priority      32768 (priority 32768 sys-id-ext 0)
            Address      00bc.6021.a4d8
            Max Age 10 sec, Forward Delay 10 sec
            Transmit Hold count 4
Interface    Port ID      Role State Designated      Port ID
            Pri.Nbr Cost          Bridge ID          Pri.Nbr
-----
Te0/0/0/14   128.2    2000    DSGN FWD   32768 00bc.6021.a4d8 128.2
Te0/0/0/20   128.3    2000    DSGN FWD   32768 00bc.6021.a4d8 128.3
Te0/0/0/26   128.4    2000    DSGN FWD   32768 00bc.6021.a4d8 128.4
Te0/0/0/27   128.5    2000    DSGN FWD   32768 00bc.6021.a4d8 128.5
Te0/0/0/38   128.6    2000    ALT BLK    32768 008a.9610.08d8 128.4
Te0/0/0/6    128.1    2000    ROOT FWD   32768 008a.9610.08d8 128.2
-----
VLAN 400:
Root ID      Priority      32768
            Address      008a.9610.08d8
            Max Age 20 sec, Forward Delay 15 sec
Bridge ID    Priority      32768 (priority 32768 sys-id-ext 0)
            Address      00bc.6021.a4d8
            Max Age 10 sec, Forward Delay 10 sec
            Transmit Hold count 4
Interface    Port ID      Role State Designated      Port ID
            Pri.Nbr Cost          Bridge ID          Pri.Nbr
-----
Te0/0/0/14   128.2    2000    DSGN FWD   32768 00bc.6021.a4d8 128.2
Te0/0/0/20   128.3    2000    DSGN FWD   32768 00bc.6021.a4d8 128.3
Te0/0/0/26   128.4    2000    DSGN FWD   32768 00bc.6021.a4d8 128.4
Te0/0/0/27   128.5    2000    DSGN FWD   32768 00bc.6021.a4d8 128.5
Te0/0/0/38   128.6    2000    ALT BLK    32768 008a.9610.08d8 128.4
Te0/0/0/6    128.1    2000    ROOT FWD   32768 008a.9610.08d8 128.2
-----

```

Related Topics

- [Information About Multiple Spanning Tree Protocol, on page 178](#)
- [Multiple Spanning Tree Protocol, on page 169](#)

Associated Commands

- `spanning-tree pvrst`
- `show spanning-tree pvrst`

Information About Multiple Spanning Tree Protocol

To configure Multiple Spanning Tree Protocol, you must understand these concepts:

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+.

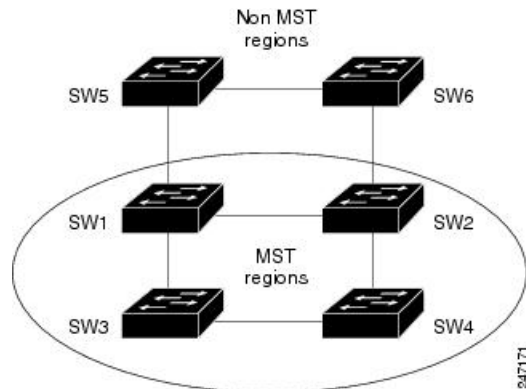
PVRST uses point-to-point wiring to provide rapid convergence of the spanning tree. The spanning tree reconfiguration can occur in less than one second with PVRST (in contrast to 50 seconds with the default settings in the 802.1D STP).
- REP (Cisco-proprietary ring-redundancy protocol)— This is a Cisco-proprietary 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.

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.

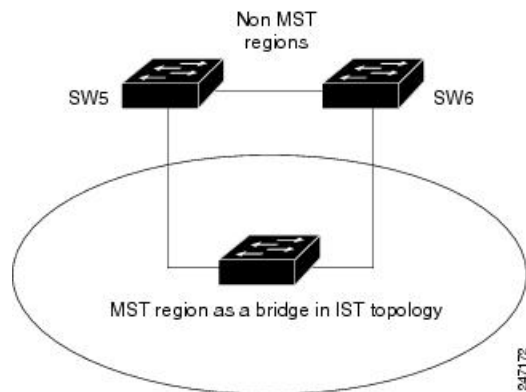
The following figure shows the operation of MST regions when bridges running MSTP are connected to bridges running legacy STP or RSTP. In this example, switches SW1, SW2, SW3, SW4 support MSTP, while switches SW5 and SW6 do not.

Figure 31: 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 32: 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.



CHAPTER 10

EVPN Features

This chapter describes how to configure Layer 2 Ethernet VPN (EVPN) features on the router.

- [EVPN Overview, on page 183](#)
- [EVPN Concepts, on page 184](#)
- [EVPN Operation, on page 185](#)
- [EVPN Route Types, on page 187](#)
- [EVPN Timers, on page 188](#)
- [Configure EVPN L2 Bridging Service, on page 190](#)
- [EVPN Modes, on page 192](#)
- [EVPN MPLS Seamless Integration with VPLS, on page 197](#)
- [Configure EVPN on the Existing VPLS Network, on page 198](#)
- [EVPN Features, on page 206](#)
- [CFM Support for EVPN, on page 248](#)
- [CFM on EVPN ELAN, on page 249](#)
- [EVPN Routing Policy, on page 252](#)

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 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.

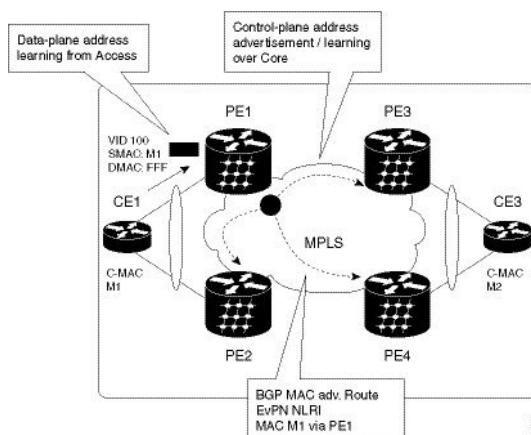
- **EAD/ES:** Ethernet Auto Discovery Route per ES is also referred to as Route Type 1. This route is used to converge the traffic faster during access failure scenarios. This route has Ethernet Tag of 0xFFFFFFFF.
- **EAD/EVI:** Ethernet Auto Discovery Route per EVI is also referred to as Route Type 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 33: 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 7: 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 NLRI. 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:

Table 8: EVPN Timers

Timer	Range	Default Value	Trigger	Applicability	Action	Sequence
startup-cost-in	30-86400s	disabled	node recovered*	Single-Homed, All-Active, Single-Active	Postpone EVPN startup procedure and Hold AC link(s) down to prevent CE to PE forwarding. Startup-cost-in timer allows PE to set core protocols first.	1
peering	0-3600s	3s	node recovered, interface recovered	All-Active, Single-Active	Starts after sending EVPN RT4 to postpone rest of EVPN startup procedure. Peering timer allows remote PE (multihoming AC with same ESI) to process RT4 before DF election will happen.	3

Timer	Range	Default Value	Trigger	Applicability	Action	Sequence
global mac evpn timer	0-300s	300s	when BGP is fired	Single-Flow-Active and Multi homed all active	Delay the time and effort required to delete the remote portion to save programming cycles working for forwarding path first.	4



- Note**
- The timers are available in EVPN global configuration mode and in EVPN interface sub-configuration mode.
 - Startup-cost-in is available in EVPN global configuration mode only.
 - Timers are triggered in sequence (if applicable).
 - Cost-out in EVPN global configuration mode brings down AC link(s) to prepare node for reload or software upgrade.

* indicates all required software components are loaded.

** indicates link status is up.

*** you can change the recovery timer on Single-Homed AC if you do not expect any STP protocol convergence on connected CE.

Global MAC EVPN Timer

Global mac evpn timer is configurable under **evpn timers mac-postpone** timer. Global MAC EVPN timer is relevant for SYNC routes only in the following scenarios:

- FRR (fast re-route) is configured: MAC and MAC+IP deletes are postponed to help with convergence.
- All-active: MAC+IPs deletes are postponed to allow time for ARP to converge.
- Single-flow-active: MAC+IP deletes are postponed to allow speculative (Address Resolution Protocol) ARP to point to local adjacency.

Typically, a route that is deleted is always quickly learned locally. Using this knowledge, we can delay the time and effort required to delete the remote portion to save programming cycles working for forwarding path first.



- Note** The timer of 5-minutes start when EVPN receives a delete from BGP. The timer doesn't start at the exact time of AC shut or mass-withdraw.

The benefit of this speculative behavior is that we can reduce MAC-IP delete/re-create churn in forwarding and BGP.

Triggers of Global Mac EVPN Timer:

- Shut / No shut on IRB/BVI Interfaces.
- Removing and adding AC Interface Configuration.
- Removing and adding BVI Interface Configuration.
- Removing and adding BVI Interface from Bridge Domains.
- Shut / No shut on AC/Main-port Interface Configuration.

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 0/0/0/1.1
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 0/0/0/1.1

  evi 6005
!
```

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 this 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 this mode, all PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.
 - Port-Active - In this mode, only the PE which is in the active mode sends and receives the traffic. This mode supports single-active redundancy load balancing at the port-level or the interface-level.

EVPN Port-Active Multihoming

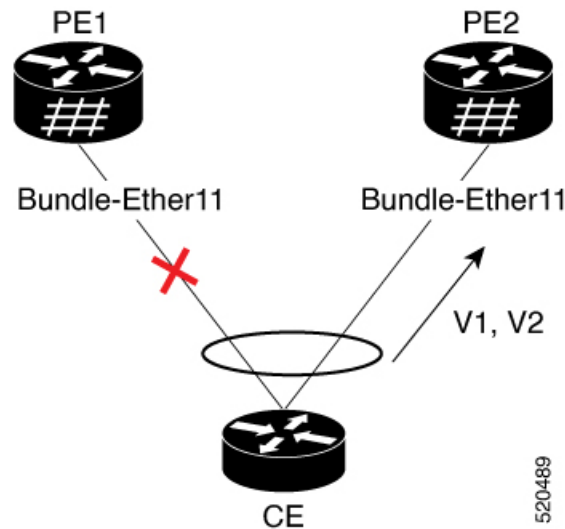
The EVPN Port-Active Multihoming feature supports single-active redundancy load balancing at the port-level or the interface-level. You can use this feature when you want to forward the traffic to a specific interface, rather than have a per-flow load balancing across multiple PE routers. This feature provides a faster convergence during a link failure. This feature enables protocol simplification as only one of the physical ports is active at a given time. You can enable this feature only on bundle interfaces.

EVPN port-active provides protocol simplification compared to Inter-Chassis Communication Protocol (ICCP), which runs on top of Label Distribution Protocol (LDP). You can use this feature as an alternative to multi-chassis link aggregation group (MC-LAG) with ICCP.

Also, you can use this feature when you want certain QoS features to work.

This feature allows one of the PEs to be in active mode and another in the standby mode at the port-level. Only the PE which is in the active mode sends and receives the traffic. The other PE remains in the standby mode. The PEs use the Designated Forwarder (DF) election mechanism to determine which PE must be in the active mode and which must be in the standby mode. You can use either modulo or Highest Random Weight (HRW) algorithm for per port DF election. By default, the modulo algorithm is used for per port DF election.

Figure 34: EVPN Port-Active Multihoming



Consider a topology where the customer edge device (CE) is multihomed to provider edge devices, PE1 and PE2. Use single link aggregation at the CE. Only one of the two interfaces is in the forwarding state, and the other interface is in the standby state. In this topology, PE2 is in the active mode and PE1 is in the standby mode. Hence, PE2 carries traffic from the CE. All services on the PE2 interface operate in the active mode. All services on the PE1 operate in the standby mode.

If the interface is running LACP, then the standby sets the LACP state to Out-of-Service (OOS) instead of bringing the interface state down. This state enables better convergence on standby to active transition.

If you remove the port-active configuration on both PE1 and PE2 and then add back the port-active configuration on both the PEs, PE2 is chosen as an active interface again.

EVPN port-active is compatible with the following services:

- L2 bridging
- L3 gateway
- L2VPN VPLS
- EVPN ELAN
- EVPN IRB
- L2VPN VPWS
- EVPN VPWS
- FXC

This feature supports both L2 and L3 port-active functionality. L2 and L3 port-active can coexist on the same bundle. For example, if you configure port-active on a bundle, the bundle can have a mix of both L3 subinterfaces and L2 subinterfaces participating in the services mentioned above.

Configure EVPN Port-Active Multihoming

Perform this task to configure EVPN port-active multihoming.

Configure the same ESI on both the routers. Configure Ethernet-Segment in port-active load-balancing mode on peering PEs for a specific interface.

Configuration Example

```
/* PE1 and PE2 Configuration */

Router#configure
Router(config)#interface Bundle-Ether11
Router(config-if)#lACP system mac 3637.3637.3637
Router(config-if)#exit

Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether11
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 11.11.11.11.11.00.11.11.11
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#commit

/* If you want enable L3 port-active, configure the IP address */
Router#configure
Router(config)#interface Bundle-Ether11
Router(config-if)#ipv4 address 10.0.0.1 255.0.0.0
Router(config-if)#ipv6 address 10::1/64
Router(config-if)#commit
```

Running Configuration

This section shows port-active running configuration.

```
configure
interface Bundle-Ether11
  lACP system mac 3637.3637.3637
  !

evpn
interface Bundle-Ether11
  ethernet-segment
  identifier type 0 11.11.11.11.11.00.11.11.11
  load-balancing-mode port-active
  !
!
interface Bundle-Ether11
  ipv4 address 10.0.0.1 255.0.0.0
  ipv6 address 10::1/64
  !
!
```

Verification

Verify that you have configured the Port-Active Multihoming feature successfully.

```
Router:PE2#show bundle bundle-ether 11

Bundle-Ether11
  Status:                               Up
  Local links <active/standby/configured>: 1 / 0 / 1
  Local bandwidth <effective/available>: 1000000 (1000000) kbps
  MAC address (source):                   02b4.3cb4.a004 (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:
  Flap suppression timer:          Off
  Cisco extensions:                 Disabled
  Non-revertive:                   Disabled
mLACP:
  IPv4 BFD:                         Not configured
  IPv6 BFD:                         Not configured

```

Port	Device	State	Port ID	B/W, kbps
Gi0/2/0/8	Local	Active	0x8000, 0x0006	1000000

Link is Active

/* PE2 is in the active mode, hence the status shows as Up and the Link as Active. */

Router:PE1#show bundle bundle-ether 11

```

Bundle-Ether11
Status:                LACP OOS (out of service)
Local links <active/standby/configured>:  0 / 1 / 1
Local bandwidth <effective/available>:    0 (0) kbps
MAC address (source):  02cf.94c1.0a04 (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:
  Flap suppression timer: Off
  Cisco extensions:      Disabled
  Non-revertive:        Disabled
mLACP:
  IPv4 BFD:              Not configured
  IPv6 BFD:              Not configured

```

Port	Device	State	Port ID	B/W, kbps
Gi0/2/0/7	Local	Standby	0x8000, 0x0006	1000000

Link is in standby due to bundle out of service state

/* PE1 is in the standby mode, hence the status shows as LACP OOS (out of service) and the Link is in standby due to bundle out of service state. */

Router:CE#show bundle bundle-ether 11

```

Bundle-Ether11
Status:                Up
Local links <active/standby/configured>:  1 / 0 / 2
Local bandwidth <effective/available>:    1000000 (1000000) kbps
MAC address (source):  02ff.566c.be04 (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
Gi0/0/0/8	Local	Active	0x8000, 0x0006	1000000
Link is Active				
Gi0/0/0/16	Local	Negotiating	0x8000, 0x000b	1000000
Partner is not Synchronized (Waiting, Standby, or LAG ID mismatch)				

```
Router:PE2#show evpn ethernet-segment interface BE11 detail
```

```
/* The following output shows that the port-active mode is configured and the port is in
the UP state. */
```

```

Ethernet Segment Id          Interface          Nexthops
-----
0011.1111.1111.0011.1111 BE11
                                192.168.0.2
                                192.168.0.3

ES to BGP Gates      : Ready
ES to L2FIB Gates   : Ready
Main port           :
  Interface name    : Bundle-Ether11
  Interface MAC     : 02b4.3cb4.a004
  IfHandle          : 0x00004170
  State             : Up
  Redundancy        : Not Defined
ESI type            : 0
  Value             : 11.1111.1111.0011.1111
ES Import RT        : 1111.1111.1100 (from ESI)
Source MAC           : 0000.0000.0000 (N/A)
Topology            :
  Operational       : MH
  Configured        : Port-Active
Service Carving     : Auto-selection
  Multicast         : Disabled
Convergence          :
  Mobility-Flush    : Count 0, Skip 0, Last n/a
Peering Details     : 2 Nexthops
  192.168.0.2 [MOD:P:7fff]
  192.168.0.3 [MOD:P:00]
Service Carving Results:
  Forwarders       : 0
  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       : 20 sec [not running]
Carving timer        : 0 sec [not running]
Local SHG label     : None

```



```
Remote SHG labels : 0
Access signal mode: Bundle OOS (Default)
```

```
Router:PE1#show evpn ethernet-segment interface BE11 detail
```

```
/* The following output shows that the por-active mode is configured and the port is in the Standby state. */
```

```

Ethernet Segment Id          Interface          Nexthops
-----
0011.1111.1111.0011.1111 BE11              192.168.0.2
                               192.168.0.3

ES to BGP Gates      : Ready
ES to L2FIB Gates   : Ready
Main port           :
  Interface name    : Bundle-Ether11
  Interface MAC     : 02cf.941c.0a04
  IfHandle          : 0x00004170
  State             : Standby
  Redundancy        : Not Defined
ESI type            : 0
  Value             : 11.1111.1111.0011.1111
ES Import RT        : 1111.1111.1100 (from ESI)
Source MAC          : 0000.0000.0000 (N/A)
Topology            :
  Operational       : MH
  Configured        : Port-Active
Service Carving     : Auto-selection
  Multicast         : Disabled
Convergence         :
  Mobility-Flush    : Count 0, Skip 0, Last n/a
Peering Details     : 2 Nexthops
  192.168.0.2 [MOD:P:00]
  192.168.0.3 [MOD:P:7fff]
Service Carving Results:
  Forwarders       : 0
  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     : 20 sec [not running]
Carving timer      : 0 sec [not running]
Local SHG label    : None
Remote SHG labels  : 0
Access signal mode: Bundle OOS (Default)

```

EVPN MPLS Seamless Integration with VPLS

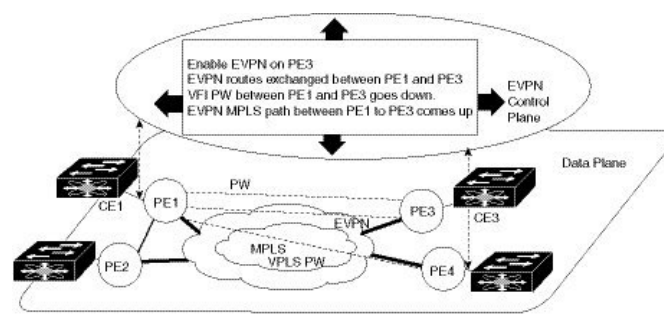
Migrate VPLS Network to EVPN Network through Seamless Integration

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 35: 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 201](#) 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 GigabitEthernet0/2/0/0.1
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)#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 GigabitEthernet0/2/0/0.1
  !
  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:

MPLS Static Labels Based VPLS

```
l2vpn
bridge group bg1
bridge-domain bd-1-1
  interface GigabitEthernet0/2/0/0.1
  !
  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 GigabitEthernet0/2/0/0.2
  !
  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 GigabitEthernet0/2/0/0.4
  !
  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

```
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

```

Locally learnt MAC routes can be viewed by forwarding table
show l2vpn forwarding bridge-domain mac-address location 0/0/cpu0
To Resynchronize MAC table from the Network Processors, use the command...
l2vpn resynchronize forwarding mac-address-table location <r/s/i>

Mac Address	Type	Learned from/Filtered on	LC learned	Resync Age/Last Change	Mapped to
0033.0000.0001	dynamic	Gi0/2/0/0.1	N/A	20 Feb 21:06:59	N/A
0033.0000.0002	dynamic	Gi0/2/0/0.2	N/A	20 Feb 21:06:59	N/A
0033.0000.0003	dynamic	Gi0/2/0/0.3	N/A	20 Feb 21:04:29	N/A
0033.0000.0004	dynamic	Gi0/2/0/0.4	N/A	20 Feb 21:06:59	N/A

The remote routes learned via evpn enabled BD
show l2vpn forwarding bridge-domain mac-address location 0/0/\$
To Resynchronize MAC table from the Network Processors, use the command...
l2vpn resynchronize forwarding mac-address-table location <r/s/i>

Mac Address	Type	Learned from/Filtered on	LC learned	Resync Age/Last Change	Mapped to
0033.0000.0001	EVPN	BD id: 0	N/A	N/A	N/A
0033.0000.0002	EVPN	BD id: 1	N/A	N/A	N/A
0033.0000.0003	EVPN	BD id: 2	N/A	N/A	N/A
0033.0000.0004	EVPN	BD id: 3	N/A	N/A	N/A

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
```

Verify 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.

EVPN Features

This section lists the supported EVPN features and how to configure them:

Configure EVPN MAC Address Limit

To configure EVPN MAC address limit, the following restrictions are applicable:

- Remote MAC addresses are programmed in the hardware irrespective of whether the MAC address limit is configured or not.
- MAC address limit can be modified correctly only when the device is not actively learning any MAC addresses. This is an expected behavior.
- When the MAC learning is enabled, you can configure the MAC address limit up to a maximum of six. However, when the MAC learning is disabled, you can configure the MAC address limit up to a maximum of five.
- The **clear l2vpn mac address table** command is not supported. The MAC address table is cleared when **shut** or **no shutdown** is performed on an attachment circuit interface or sub interface, or when the MAC aging timer expires.
- You can configure both MAC limit Action and MAC notification. However, the configuration does not take into effect as the functionality is not supported.

Configuration Example

Perform this task to configure EVPN MAC address limit.

This table lists the MAC address limit parameters and values that are configured:

Parameter	Value
MAC address limit	50
MAC limit threshold	80%

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN-BG-SH
Router(config-l2vpn-bg)# bridge-domain EVPN_2701
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# limit
Router(config-l2vpn-bg-bd-mac-limit)# maximum 50
```

```

Router(config-l2vpn-bg-bd-mac-limit)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# mac limit threshold 80
Router(config-l2vpn)# commit

```

Running Configuration

```

l2vpn
bridge group EVPN-BG-SH
  bridge-domain EVPN_2701
    mac
      limit
        maximum 50
    !
  !
!
mac limit threshold 80
commit

```

Verification

Verify the EVPN MAC address limit parameters are set as described in above table:

```

Router# show l2vpn bridge-domain bd-name EVPN_2701 detail
Legend: pp = Partially Programmed.
Bridge group: EVPN-BG-SH, bridge-domain: EVPN_2701, 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: 50, Action: none, Notification: syslog
MAC limit reached: no, threshold: 80%
  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
  DHCPv4 Snooping profile: none
  IGMP Snooping: disabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  Storm Control: disabled
  Bridge MTU: 1500
  MIB cvplsConfigIndex: 26
  Filter MAC addresses:
  P2MP PW: disabled
  Create time: 21/04/2019 16:28:05 (2d23h ago)
  No status change since creation
  ACs: 1 (1 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
      evi: 6101

```

```

XC ID 0x8000040c
Statistics:
  packets: received 0 (unicast 0), sent 0
  bytes: received 0 (unicast 0), sent 0
  MAC move: 0
List of ACs:
AC: Bundle-Ether101.2701, state is up, active in RG-ID 101
  Type VLAN; Num Ranges: 1
  Rewrite Tags: [1000, 2000]
  VLAN ranges: [2701, 2701]
  MTU 9112; XC ID 0xa000060b; interworking none; MSTi 6
  MAC learning: enabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
MAC limit: 50, Action: none, Notification: syslog
MAC limit reached: no, threshold: 80%
  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
  DHCPv4 Snooping profile: none
  IGMP Snooping: disabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  Storm Control:
    Broadcast: enabled(160000 pps)
    Multicast: enabled(160000 pps)
    Unknown unicast: enabled(160000 pps)
  Static MAC addresses:
  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
  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 Access PWs:
List of VFIs:
List of Access VFIs:

```

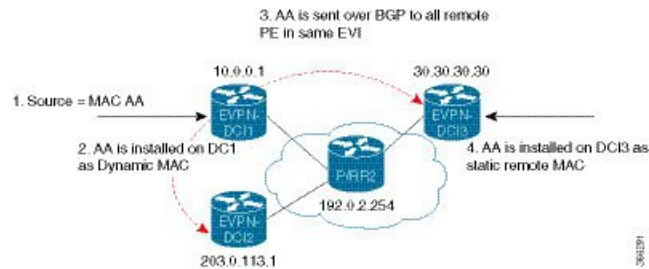
EVPN Software MAC Learning

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 36: 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 On EVPN bridge domain, the Cisco NCS 5500 router does not support control word and does not enable control word by default.



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 TenGigE0/4/0/10.2001
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

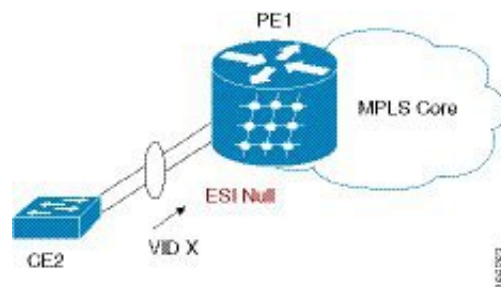
The following are the modes in which EVPN Software MAC Learning is supported:

- 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 37: 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.

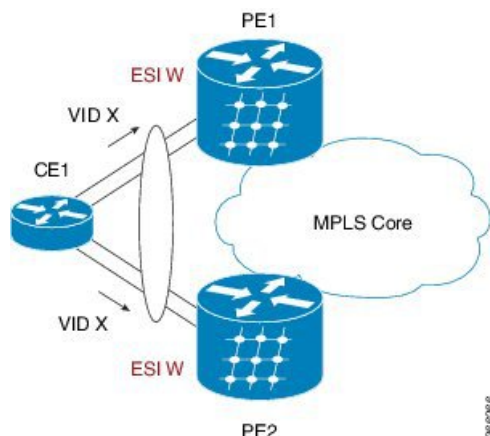
```
RP/0/RSP0/CPU0:router# show evpn ethernet-segment interface Te0/4/0/10 detail

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 38: 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 01.11.00.00.00.00.00.01

/* 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 01.11.00.00.00.00.00.01
  !
  !
  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 dot1q 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: TenGigE0/4/0/10.2001, 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 0/6/0/1.36001 00a3.0001.0001

```

Verify the summary information for the MAC address.

```
RP/0/RSP0/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
0000.2001.5555   dynamic      Te0/0/0/2/0.2001        N/A         11 Jan 14:37:22
N/A <-- local dynamic
00bb.2001.0001   dynamic      Te0/0/0/2/0.2001        N/A         11 Jan 14:37:22
N/A
0000.2001.1111   EVPN         BD id: 1110              N/A         N/A
N/A <-- remote static
00a9.2002.0001   EVPN         BD id: 1110              N/A         N/A
N/A

```



Note NCS platforms do not distinguish between MACs learned on a different LC and MACs learned on a remote device. In both cases, the MACs are handled as "EVPN", because the EVPN functionality is a superset of all other modes and is sufficient to program the hardware.

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
```

EVI	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	0000.2001.5555	20.1.5.55	TenGigE0/0/0/2/0.2001	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).

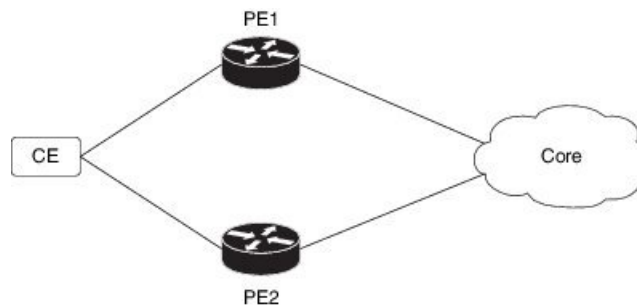
Use the **cost-out** command to bring down all the bundle interfaces belonging to an Ethernet VPN (EVPN) Ethernet segment on a node. The Ethernet A-D Ethernet Segment (ES-EAD) routes are withdrawn before shutting down the bundles. The PE signals to the connected customer edge (CE) device to bring down 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 39: 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

```

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 5500 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 5500 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 5500 Series Routers*.

- Native EVPN

For more information see, *EVPN Features* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 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
Router(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.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.00
  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 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 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

Group	Name	ST	Segment 1 Description	ST	Segment 2 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

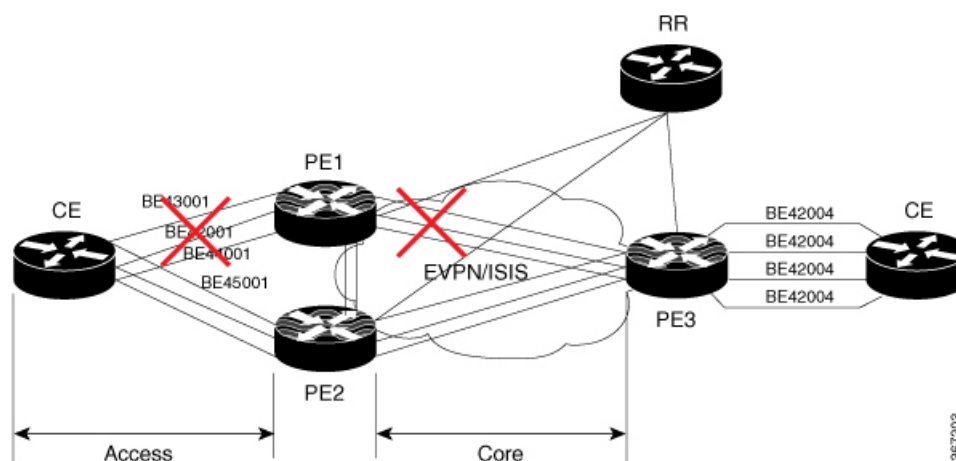
EVPN Core Isolation Protection

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 40: 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.

Starting from Cisco IOS-XR software version 7.1.2, you can configure a sub-interface as an EVPN Core. With this enhancement, when using IOS-XR software versions 7.1.2 and above, EVPN core facing interfaces can be physical, bundle main, or sub-interfaces. For all Cisco IOS-XR software versions lower than 7.1.2, EVPN core facing interfaces must be physical or bundle main. Sub-interfaces are not supported.

EVPN core facing interfaces can be physical main interface or subinterface, or bundle main interface or subinterface.

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.

```

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
```

Highest Random Weight Mode for EVPN DF Election

The Highest Random Weight (HRW) Mode for EVPN DF Election feature provides optimal load distribution of Designated Forwarder (DF) election, redundancy, and fast access. It ensures a nondisruptive service for an ES irrespective of the state of a peer DF.

The DF election is calculated based on the weight. The highest weight becomes the DF and the subsequent weight becomes a backup DF (BDF). The weight is determined by the mathematical function of EVI, ESI, and the IP address of the server.

DF weight calculation is based on the weight vector:

$$W_{rand}(v, S_i) = (1103515245((1103515245 \cdot S_i + 12345) \text{ XOR } D(v)) + 12345) \pmod{2^{31}}$$

where:

- S_i: IP Address of the server i
- v: EVI
- D(v): 31 bit digest [CRC-32 of v]

The existing DF election algorithm is based on ordinal value of a modulus calculation, and it comprises of number of peers and EVI. The DF is determined by the mathematical function of ESI and EVI, which is called “service carving”. This mode of DF election is described in RFC 7432.

In modulus calculation mode, the algorithm does not perform well when the Ethernet tags are all even or all odd. When the Ethernet Segment (ES) is multihomed to two PEs, all the VLANs pick only one of the PEs as the DF; one of the PEs does not get elected at all as the DF. The DF election is not optimal in this mode of operation.

The HRW mode of DF election has the following advantages over modulus mode of DF election:

- The DF election for the respective VLANs is equally distributed among the PEs.
- When a PE which is neither a DF nor a BDF hosts some VLANs on a given ES, and if the PE goes down, or its connection to the ES goes down, it does not result in a DF and BDF reassignment to the other PEs. This eliminates computation during the connection flaps.
- It avoids the service disruption that are inherent in the existing modulus based algorithm.
- The BDF provides redundant connectivity. The BDF ensures that there is no traffic disruption when a DF fails. When a DF fails, the BDF becomes the DF.

Configure Highest Random Weight Mode for EVPN DF Election

Perform this task to configure Highest Random Weight Mode for EVPN DF Election feature.

Configuration Example

```
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether 23
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# service-carving hrw
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
configure
 evpn
  interface Bundle-Ether 23
    ethernet-segment
      service-carving hrw
  !
!
!
```

Verification

Verify that you have configured HRW mode of DF election.

```
Router#show evpn ethernet-segment interface bundleEther 23 carving detail
Ethernet Segment Id      Interface      Nexthops
-----
0011.1111.1111.1111.1111 Gi0/2/0/0    192.168.0.2
                               192.168.0.3

ES to BGP Gates      : Ready
ES to L2FIB Gates   : Ready
Main port            :
  Interface name     : GigabitEthernet0/2/0/0
  Interface MAC      : 02db.c740.ca4e
  IfHandle           : 0x01000060
  State              : Up
  Redundancy         : Not Defined
ESI type             : 0
  Value              : 11.1111.1111.1111.1111
ES Import RT        : 0011.0011.0011 (Local)
```

```

Source MAC      : 0000.0000.0000 (N/A)
Topology       :
  Operational   : MH, Single-active
  Configured    : Single-active (AApS) (default)
Service Carving : HRW    -> Operation mode of carving
Peering Details : 192.168.0.2[HRW:P:00] 192.168.0.3[HRW:P:00] -> Carving capability as
advertised by peers
Service Carving Results:
  Forwarders    : 1
  Permanent     : 0
  Elected      : 0
  Not Elected  : 1
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 : 28109
Remote SHG labels : 1
                  24016 : nexthop 192.168.0.3

```

Associated Commands

- service-carving
- show evpn ethernet-segment

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.

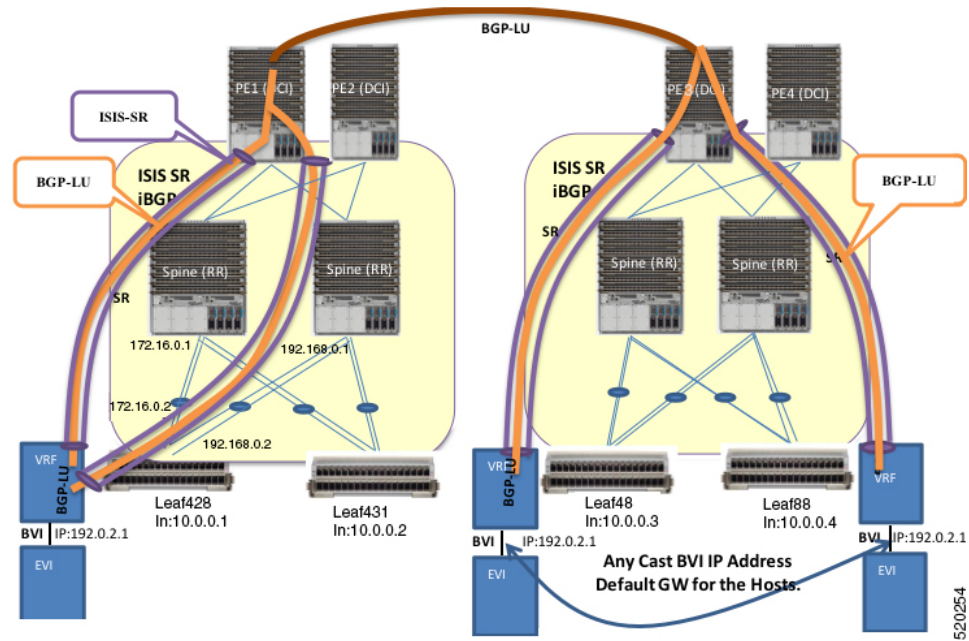
This feature supports the following services:

- 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 5500 series routers and NCS57 line cards.

Figure 41: 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 routes 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.

Limitations

The following EVPN services are not supported over BGP-LU over IGP with L2 unicast and BUM traffic on the Cisco NCS 5500 series routers and NCS57 line cards:

- EVPN-ELAN and ELINE for EVPN Multi-Homing Single-Active
- EVPN IRB with VRF (intra-subnet)

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 */
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 */
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

/* 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 bdl
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 */
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 vpv4 unicast re-originated
advertise vpv6 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 vpv4 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 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
```

Related Topics

[EVPN Bridging and VPWS Services over BGP-LU Underlay, on page 230](#)

Associated Commands

- show l2vpn bridge-domain
- show bgp l2vpn evpn neighbors
- show cef vrf

EVPN Access-Driven DF Election

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

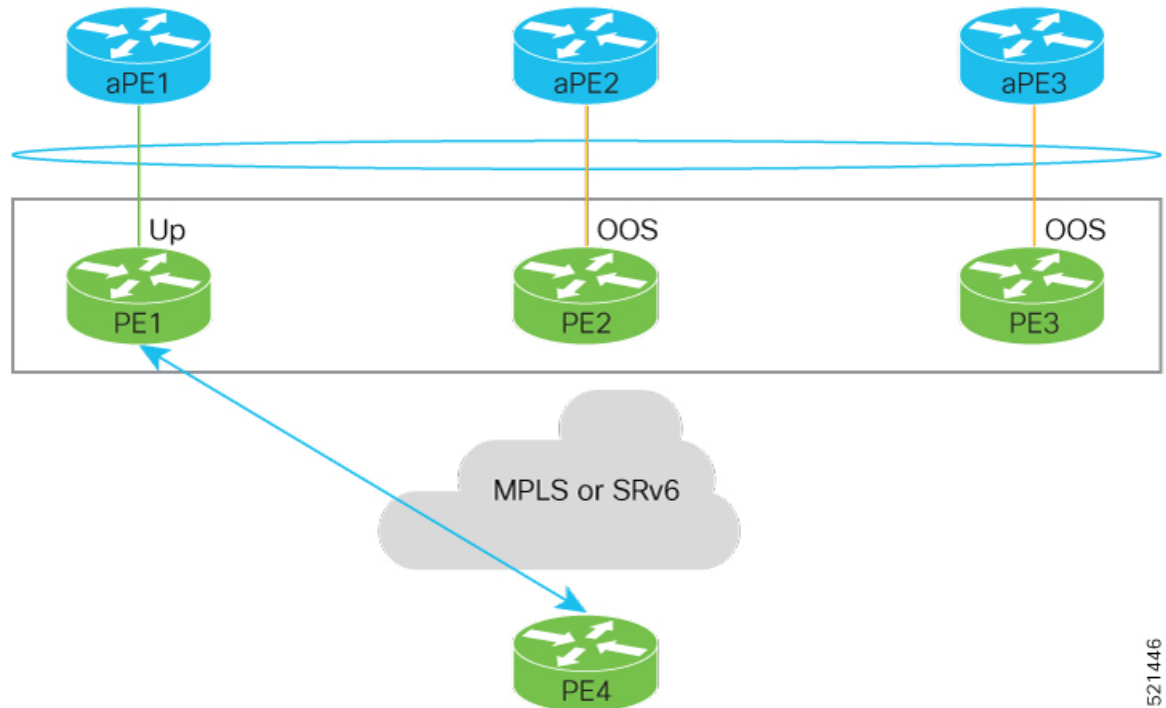
- The feature is supported only in an EVPN-VPWS scenario where EVPN PEs are in the port-active mode.
- The bundle attached to the ethernet segment must be configured with **lACP mode active**.

LACP mode on is not supported.

Topology

Let's understand the feature on how the backup path is precomputed with the following topology.

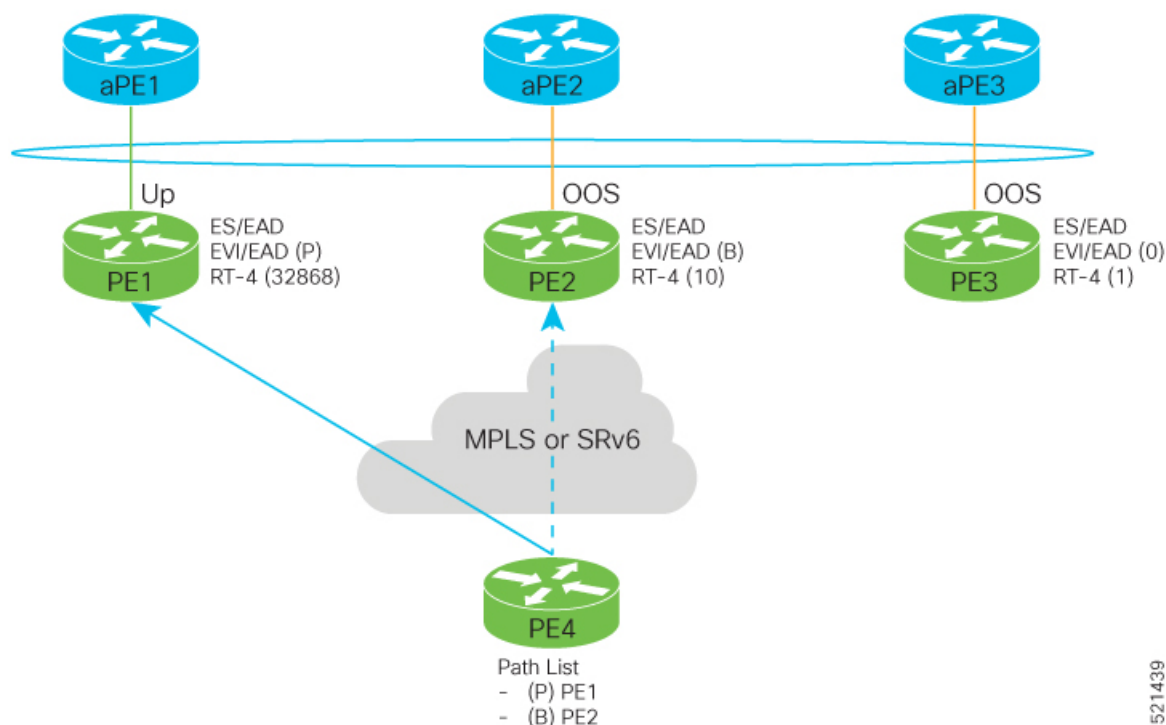
Figure 42: EVPN Access-Driven DF Election



- 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.



521439

- 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 devices 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.

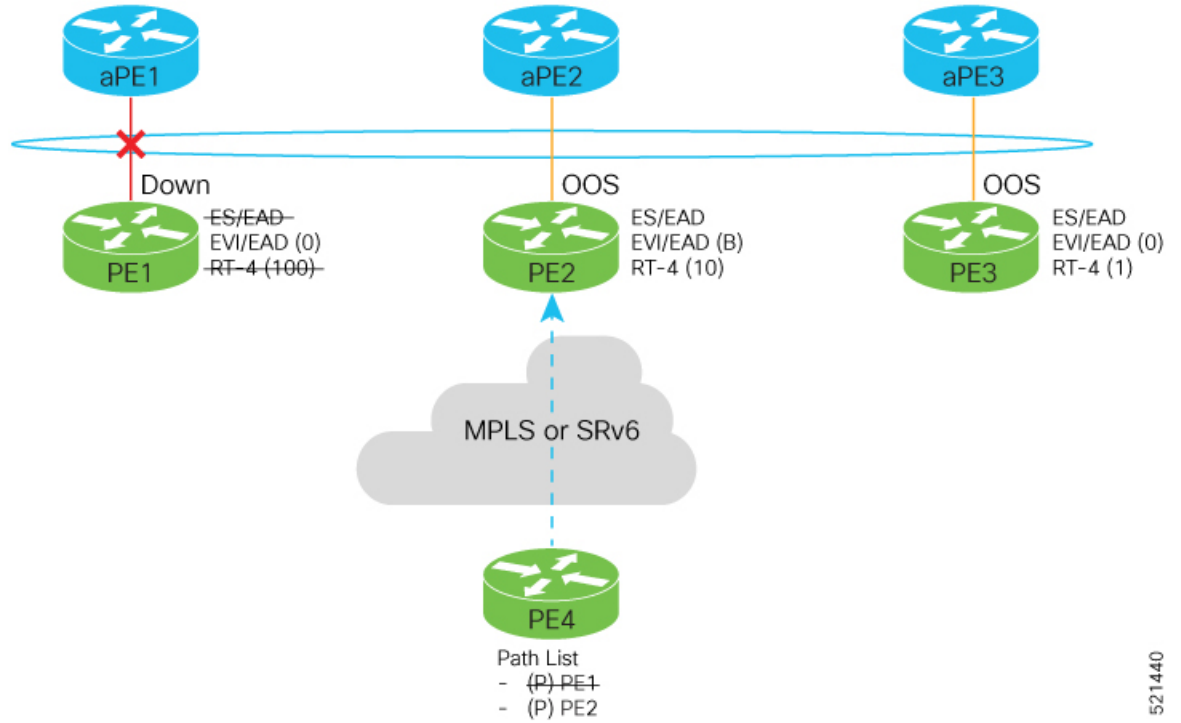
Failure and Recovery Scenarios

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

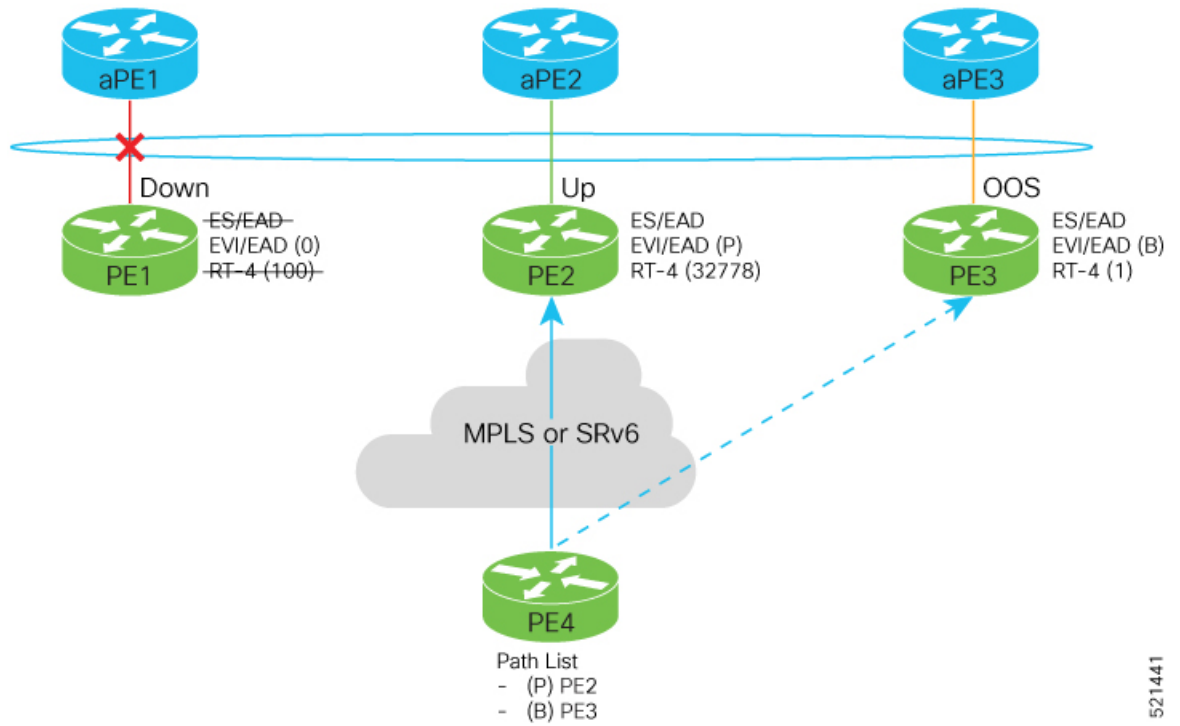
Consider a scenario where the aPE1-PE1 interface is down.



521440

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.

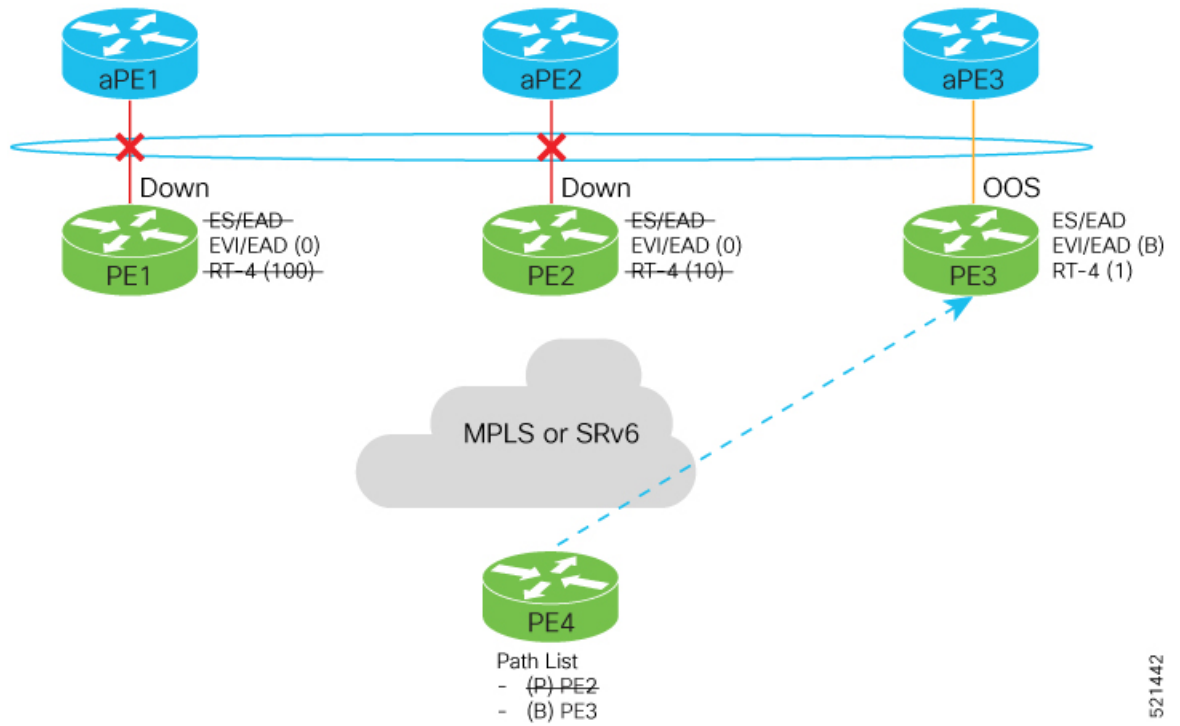
The aPE2-PE2 becomes the primary with a weight of 32778, and aPE3-PE3 becomes the backup. The aPE2-PE2 advertises P-bit to PE4. aPE3-PE3 advertises the B-bit to PE4.



521441

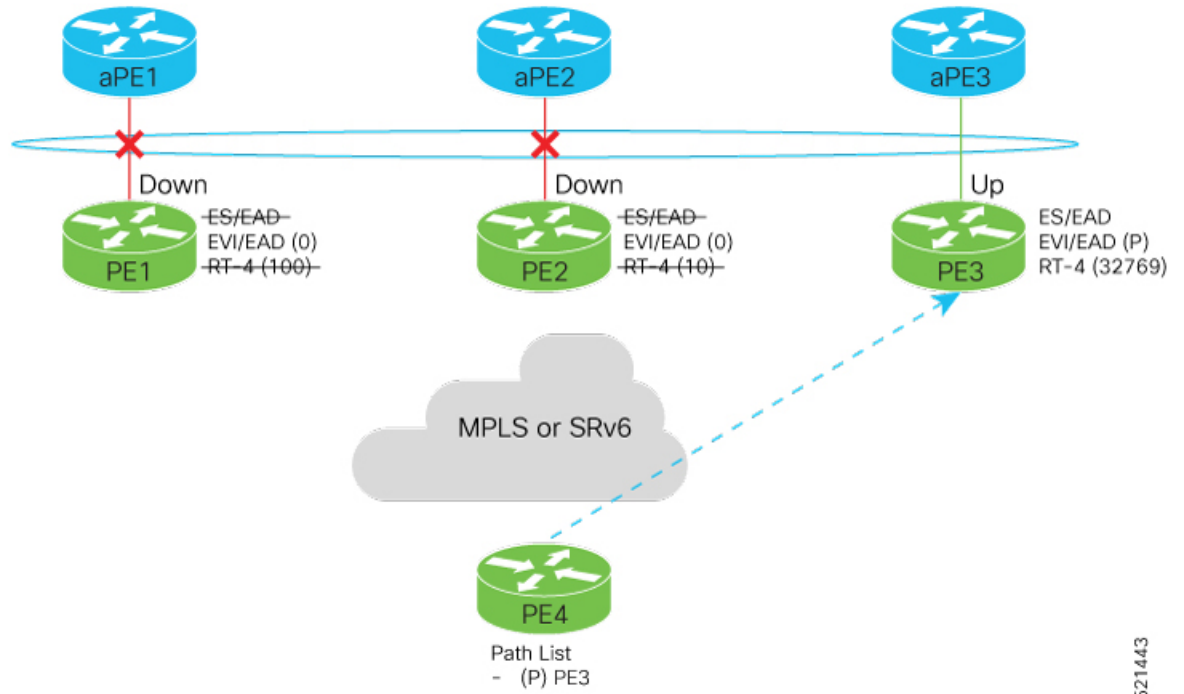
Scenario - 2

Consider a scenario where aPE2-PE2 interface is also down.



521442

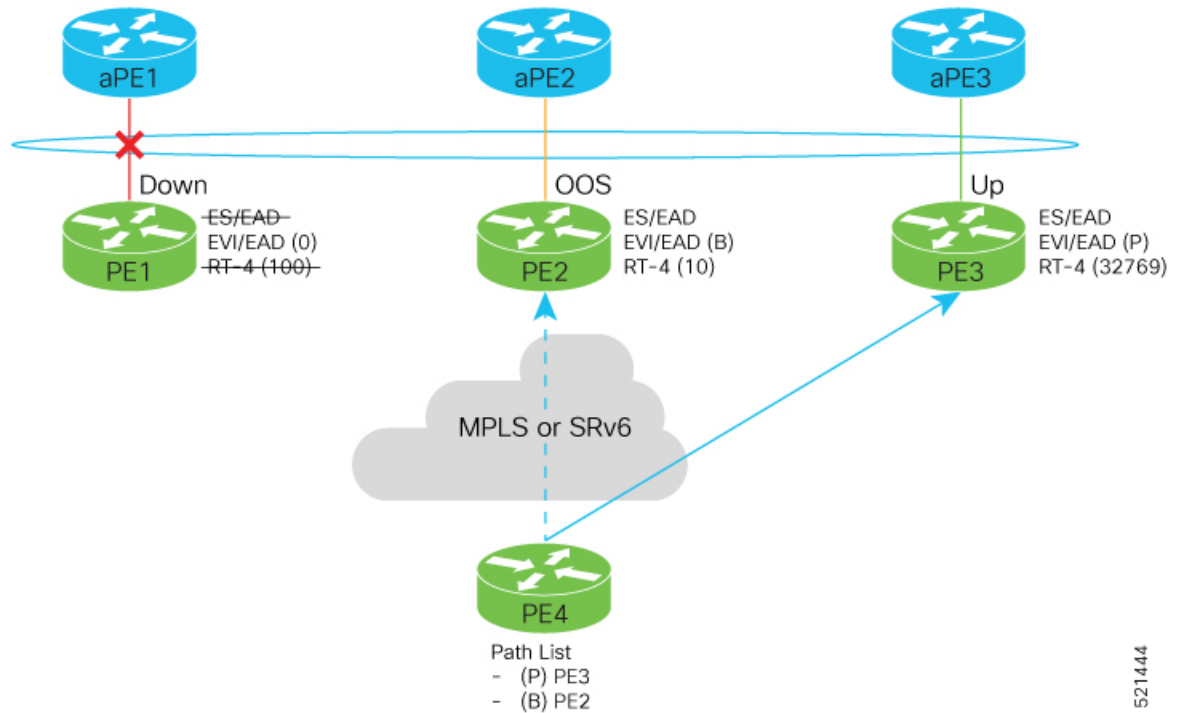
When the aPE2-PE2 interface is also down, the traffic is sent through aPE3-PE3 link. aPE3-PE3 becomes the primary path with a weight of 32769.



521443

Scenario - 3

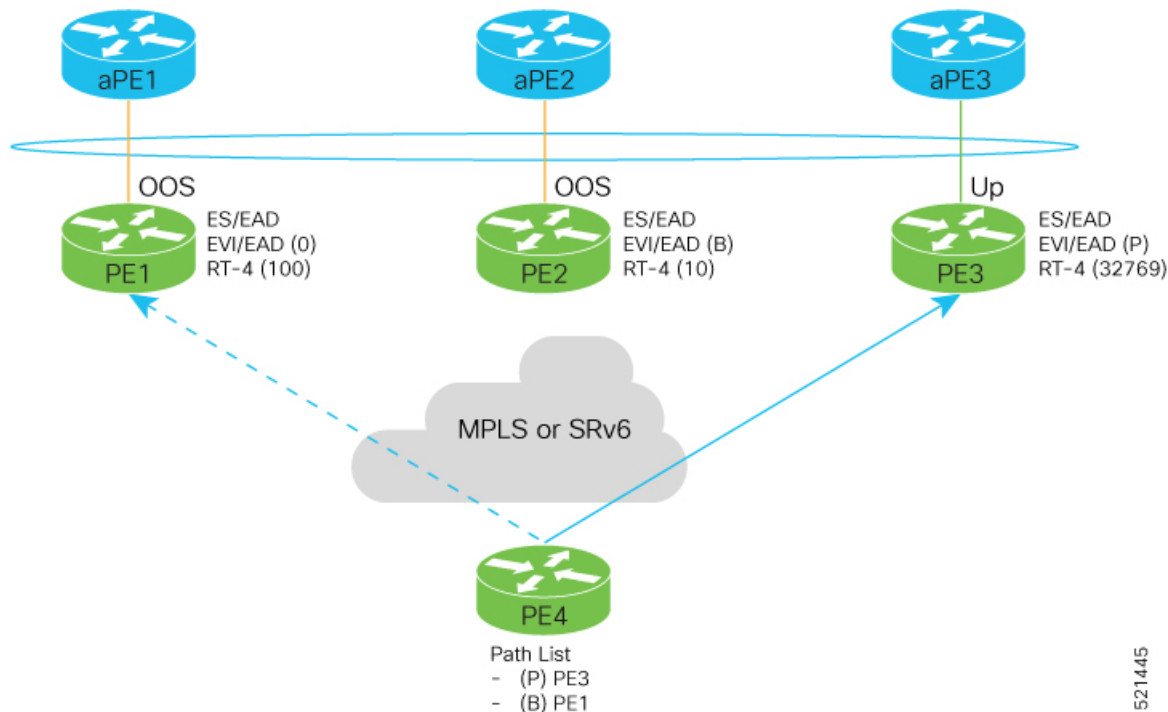
When the aPE2-PE2 interface comes up, the aPE3-PE3 link still remains the primary path. aPE2-PE2 interface becomes the backup path with a weight of 10.



521444

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.



521445

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.

For more information about Ethernet Connectivity Fault Management (CFM), refer to the *Configuring Ethernet OAM* chapter in the *Interface and Hardware Component Configuration Guide for Cisco NCS 5500 Series Routers*.

CFM on EVPN ELAN

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.

Cisco IOS XR Software Release 6.6.1 introduces CFM support for single-homed EVPN Emulated Local Area Network (ELAN) services. This functionality helps you to monitor the ELAN services of users against their contractual service-level agreements (SLAs), thereby providing high speed Layer 2 and Layer 3 services with high resiliency and less operational complexity to different market segments.

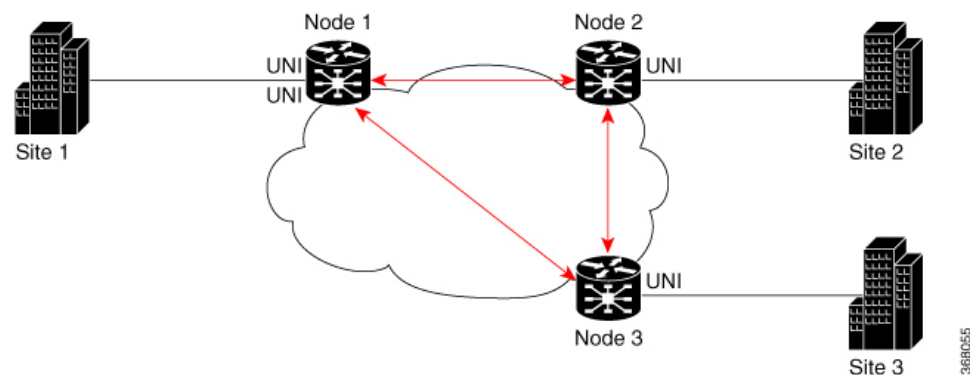
Restrictions for CFM on EVPN ELAN

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.

Configure CFM on EVPN ELAN

Figure 43: 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 0/0/0/2.100 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.

```

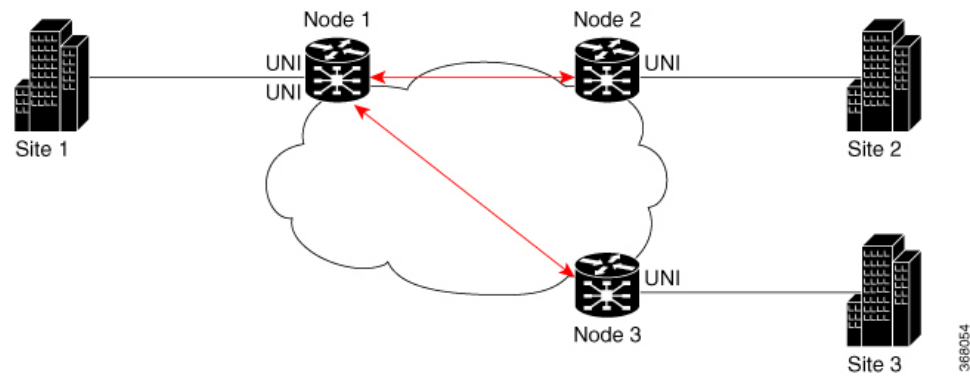
ethernet cfm
 domain bd-domain level 1 id null
  service bd-domain bridge group bg-elan bridge-domain bd-elan id icc-based MC MCMC
  continuity-check interval 1m
  mep crosscheck
  mep-id 1112
  mep-id 1113
  !
!
!

interface GigabitEthernet0/0/0/2.100 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 44: 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 0/0/0/2.100 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 GigabitEthernet0/0/0/2.100 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
!
```

Related Topics

[CFM on EVPN ELAN, on page 249](#)

Associated Commands

- continuity-check

- ethernet cfm
- mep crosscheck
- mep domain
- sla operation

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:

Route Type (1 octet)	*
Length (1 octet)	
Route Distinguisher (RD) (8 octets)	*
Ethernet Segment Identifier (10 octets)	*
Ethernet Tag ID (4 octets)	*
MPLS Label (3 octets)	

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:

Route Type (1 octet)	*
Length (1 octet)	
RD (8 octets)	*
Ethernet Segment Identifier (10 octets)	
Ethernet Tag ID (4 octets)	*
MAC Address Length (1 octet)	*
MAC Address (6 octets)	*
IP Address Length (1 octet)	*
IP Address (0, 4, or 16 octets)	*
MPLS Label1 (3 octets)	
MPLS Label2 (0 or 3 octets)	

3063156

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)] [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:

Route Type (1 octet)	*
Length (1 octet)	
RD (8 octets)	*
Ethernet Tag ID (4 octets)	*
IP Address Length (1 octet)	*
Originating Router's IP Address (4 or 16 octets)	*

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:

```

+-----+
|Route Type (1 octet)          |*
+-----+
|Length (1 octet)             |
+-----+
|RD (8 octets)                |*
+-----+
|Ethernet Segment Identifier (10 octets)|*
+-----+
|IP Address Length (1 octet)   |*
+-----+
|Originating Router's IP Address |*
|(4 or 16 octets)             |
+-----+

```

3/6/2016

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:

```

+-----+
|Route Type (1 octet)                |*
+-----+
|Length (1 octet)                    |
+-----+
|RD   (8 octets)                     |*
+-----+
|Ethernet Segment Identifier (10 octets)|
+-----+
|Ethernet Tag ID (4 octets)          |*
+-----+
|IP Address Length (1 octet)         |*
+-----+
|IP Address (4 or 16 octets)         |*
+-----+
|GW IP Address (4 or 16 octets)      |
+-----+
|MPLS Label (3 octets)               |
+-----+

```

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 referring 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 referring 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 referring 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 referring it in a route-policy (Attach point - neighbor-in) */
Wed Oct 26 11:52:23.720 IST
esi-set demo_es1
  ad34.1233.1222.ffff.44ff,
  ad34.1233.1222.ffff.6666
end-set
!
route-policy use_es1
  if esi in demo_es1 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
```

```
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
!
```




CHAPTER 11

Configure EVPN IRB, Distributed Anycast Gateway and E-tree

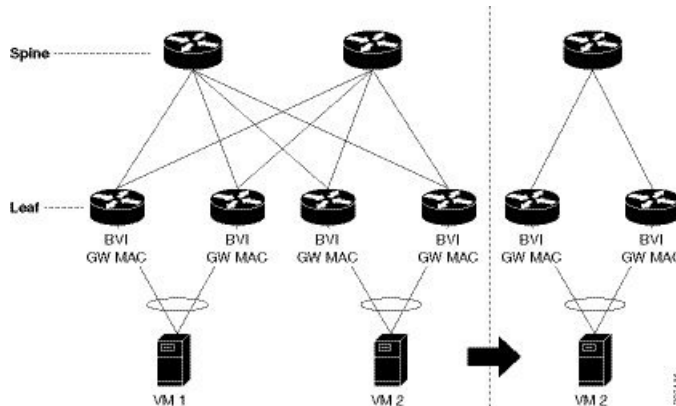
This chapter introduces you to Ethernet VPN (EVPN) Integrated Routing and Bridging (IRB), Distributed Anycast Gateway, and E-Tree features and their description.

- [EVPN IRB](#) , on page 269
- [EVPN Single-Homing Access Gateway](#) , on page 271
- [EVPN Multihoming All-Active](#), on page 273
- [EVPN Single-Active Multihoming for Anycast Gateway IRB](#), on page 273
- [Enable Auto-BGP RT with Manual ESI Configuration](#), on page 277
- [Supported EVPN IRB Scenarios](#), on page 277
- [Distributed Anycast Gateway](#), on page 278
- [VM Mobility Support](#), on page 281
- [Duplicate IP Address Detection](#), on page 306
- [EVPN Automatic Unfreezing of MAC and IP Addresses](#), on page 309
- [EVPN E-Tree](#), on page 309
- [DHCPv4 Relay on IRB](#), on page 318
- [DHCPv4 Relay Synchronization for All-Active Multihoming](#), on page 325
- [DHCPv6 Relay IAPD on IRB](#), on page 326
- [DHCPv6 PD Synchronization for All-Active Multihoming using Session Redundancy](#) , on page 329
- [IAPD Route Distribution and Withdrawal in DHCPv6 Relay](#), on page 332

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 45: 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 overlaid 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 come from the MAC address configured on the bundle. To check the dynamic pool of MAC addresses included, use the `show ethernet mac-allocation detail` command.

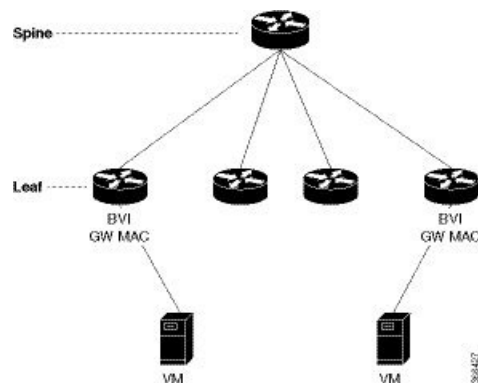
For example, if the dynamic MAC address was 008A.9624.48D8 and the configured static MAC address is 0011.2222.ABCD. Then, the source MAC for transit (forwarded) traffic will be 008A.9624.4BCD.

For more information on limitations, refer *Limitations and Compatible Characteristics of Ethernet Link Bundles* in *Interface and Hardware Component Configuration Guide for Cisco NCS 5500 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 46: 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.

Configure GRT example

Running Configuration

This section shows the GRT running configuration.

```

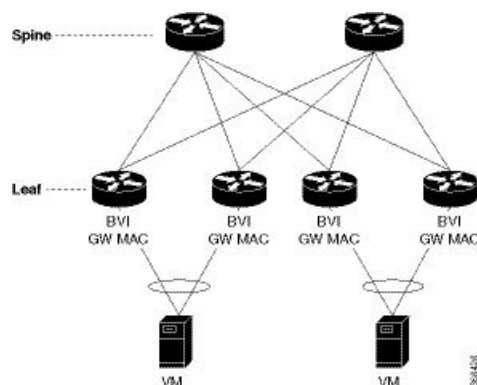
cef adjacency route override rib
evpn
  evi 41020
  advertise-mac
  !
  !
  !
interface TenGigE0/0/0/5.417 l2transport
  encapsulation dot1q 417
  rewrite ingress tag pop 1 symmetric
  !
interface BVI417
  host-routing
  ipv4 address 192.168.0.65 255.255.255.248
  mac-address abf.4065.417
  !
router bgp 65000
  bgp router-id 1.1.1.1
  address-family ipv4 unicast
  redistribute connected
  !
  address-family l2vpn evpn
  bgp implicit-import
  !
  neighbor 2.2.2.2
  remote-as 65000
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family l2vpn evpn
  !
  !
  !
l2vpn
  bridge group test
  bridge-domain test
  interface TenGigE0/0/0/5.417
  !
  routed interface BVI417
  !
  evi 41020
  !
  !
  !
  !
  !

```

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 47: EVPN Multi-Homing All-Active



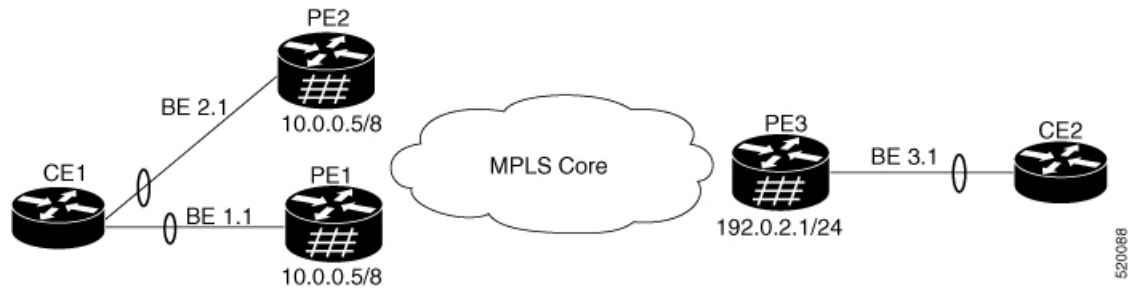
The above topology describes how EVPN Multi-homing access gateway enables redundant network connectivity by allowing a CE device to connect to more than one PE device. Disruptions to the network connectivity are prevented by allowing a CE device to be connected to a PE device or several PE devices through 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 48: EVPN: Single-Active Multihoming for Anycast Gateway IRB

Different bundles on CE1



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 40.00.00.00.00.00.00.01
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 40.00.00.00.00.00.00.01
      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 be synced across the EVPN PE to which the servers are multi-homed.

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 exchanged 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 exchanged 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 than 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 packets 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.

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

```

/* Configure CEF to prefer RIB prefixes over adjacency prefixes.*/

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

RP/0/RSP0/CPU0:router(config)# cef adjacency route override rib

/* 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 CEF adjacency overwrite. */
cef adjacency route override rib

/* 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) 1000000100010010000100010800
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 non-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-1010 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-1010 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 56.56.56.56.56.56.56.01
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 56.56.56.56.56.56.56.02
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
!

```



```

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

Perform 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 306](#)

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 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:

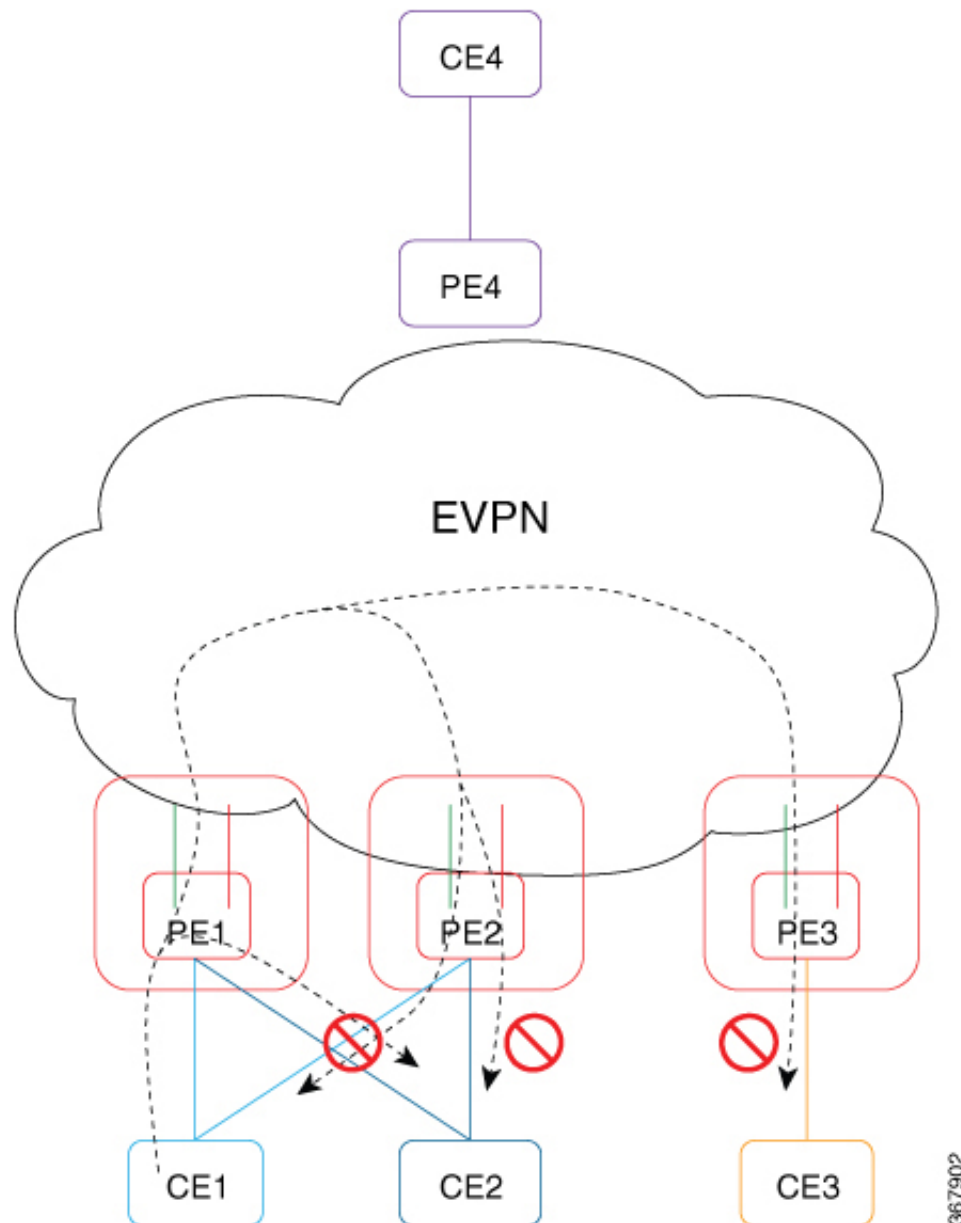
- **clear l2route evpn mac** { mac-address } | **all** [evi evi] **frozen-flag**
- **clear l2route evpn ipv4** { ipv4-address } | **all** [evi evi] **frozen-flag**
- **clear l2route evpn ipv6** { ipv6-address } | **all** [evi evi] **frozen-flag**

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 49: 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.

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 synchronization, 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 synchronization, 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.

Communication 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.

```
/* 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 MLAG on PE1 for dual-home all-active EVPN */

Router# configure
Router(config)# redundancy
Router(config-redundancy)# ICCP group 1
Router(config-iccp-group)# mlacp node 1
Router(config-iccp-group)# mlacp system mac 000d.0002.0011
Router(config-iccp-group)# mlacp system priority 1
Router(config-iccp-group)# mode singleton
Router(config-iccp-group)# backbone
Router(config-iccp-group-backbone)# interface Bundle-Ether110
!

Router# configure
Router(config)# interface Bundle-Ether1121
Router(config-if)# description DH-F2-1
Router(config-if)# lacp switchover supress-flaps 300
Router(config-if)# mlacp iccp-group 1
Router(config-if)# bundle wait-while 100
Router(config-if)# load-inerval 30

/* Configure MLAG on PE2 for dual-home all-active EVPN */
```

```

Router# configure
Router(config)# redundancy
Router(config-redundancy)# ICCP group 1
Router(config-iccp-group)# mlacp node 2
Router(config-iccp-group)# mlacp system mac 000d.0002.0011
Router(config-iccp-group)# mlacp system priority 1
Router(config-iccp-group)# mode singleton
Router(config-iccp-group)# backbone
Router(config-iccp-group-backbone)# interface Bundle-Ether120
!
Router# configure
Router(config)# interface Bundle-Ether1121
Router(config-if)# description DH-F2-1
Router(config-if)# lacp switchover supress-flaps 300
Router(config-if)# mlacp iccp-group 1
Router(config-if)# bundle wait-while 100
Router(config-if)# load-inerval 30

/* 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 */
redundancy
iccp
group 1
mlacp node 1
mlacp system mac 000d.0002.0011
mlacp system priority 1

```

```

        mode singleton
        backbone
        interface Bundle-Ether110
    !
interface Bundle-Ether1121
description DH-F2-1
lACP switchover suppress-flaps 300
mlACP iccp-group 1
bundle wait-while 100
load-interval 30

!

evpn
evi 1
    etree leaf
    !

l2vpn
bridge group bg1
    bridge-domain bd1
        interface Bundle-Ether1121.1
            routed interface BVI1
        !
    evi 1

interface Bundle-Ether1121.1
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 */
redundancy
iccp
    group 1
        mlACP node 2
        mlACP system mac 000d.0002.0011
        mlACP system priority 1
        mode singleton
        backbone
        interface Bundle-Ether120
    !
!
interface Bundle-Ether1121
description DH-F2-1
lACP switchover suppress-flaps 300
mlACP iccp-group 1
bundle wait-while 100
load-interval 30

```

```

evpn
 evi 1
  etree leaf
  !
  !

l2vpn
 bridge group bg1
  bridge-domain bd1
  interface Bundle-Ether1121.1
  routed interface BV11
  !
  evi
  !
interface Bundle-Ether1121.1
l2transport
 encapsulation dot1q 1
 rewrite ingress tag pop 1 symmetric
 !
 !
interface BV11
 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 ESO 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 309](#)

Associated Commands

- etree leaf
- show bgp l2vpn evpn rd

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.

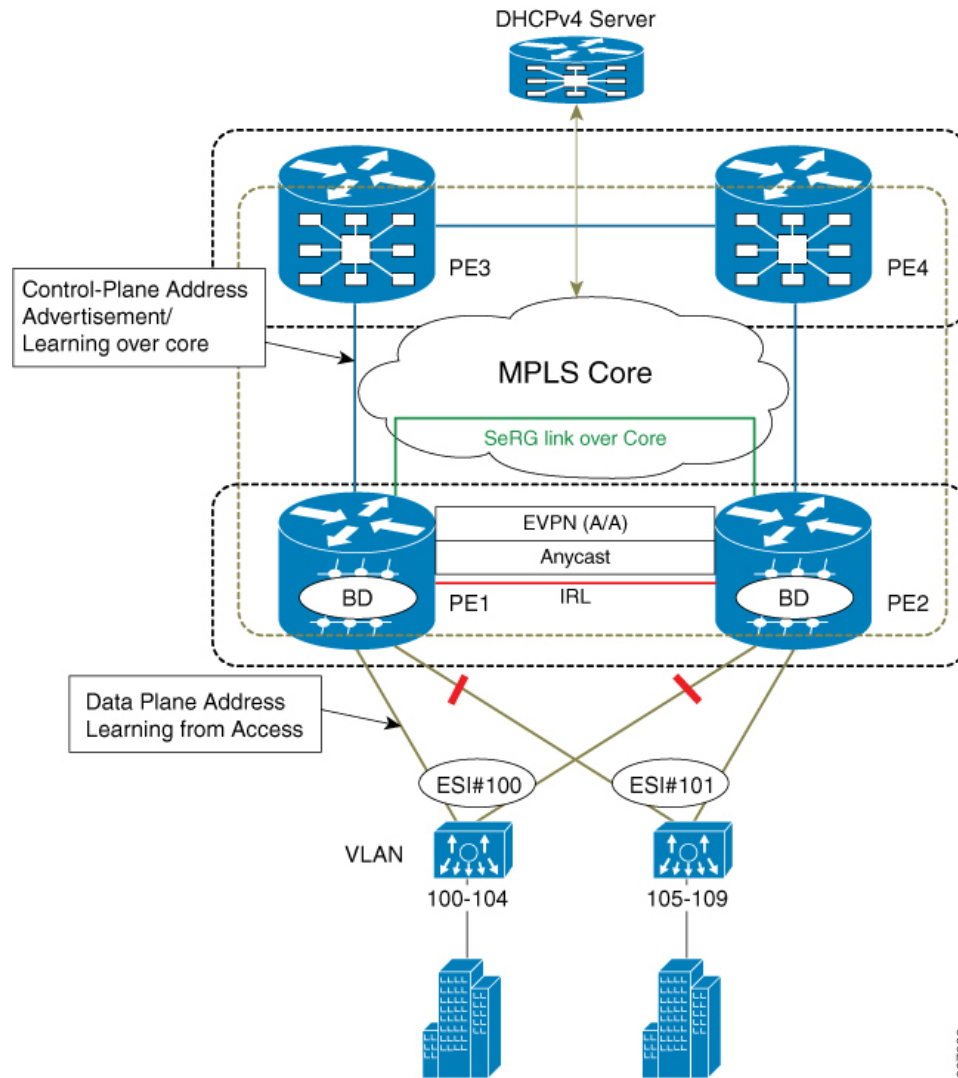
Relay-Option Policy	DHCPv4 Access Side Packet	Local Configuration	DHCPv4 Relay Packet Decision
Encapsulate	No Relay-Information	DHCPv4-Profile with Remote-ID L2Transport AC with Circuit-ID	Relay-Agent with Remote-ID and Circuit-ID
Encapsulate	Relay-Information (Remote-ID and Circuit-ID)	DHCPv4-Profile with Remote-ID L2Transport 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
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

Relay-Option Policy	DHCPv4 Access Side Packet	Local Configuration	DHCPv4 Relay Packet Decision
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 50: PON behavior in handling DHCPv4 Server for EVPN All-Active Multihoming



PON behavior in handling DHCPv4 Server for EVPN All-Active Multihoming

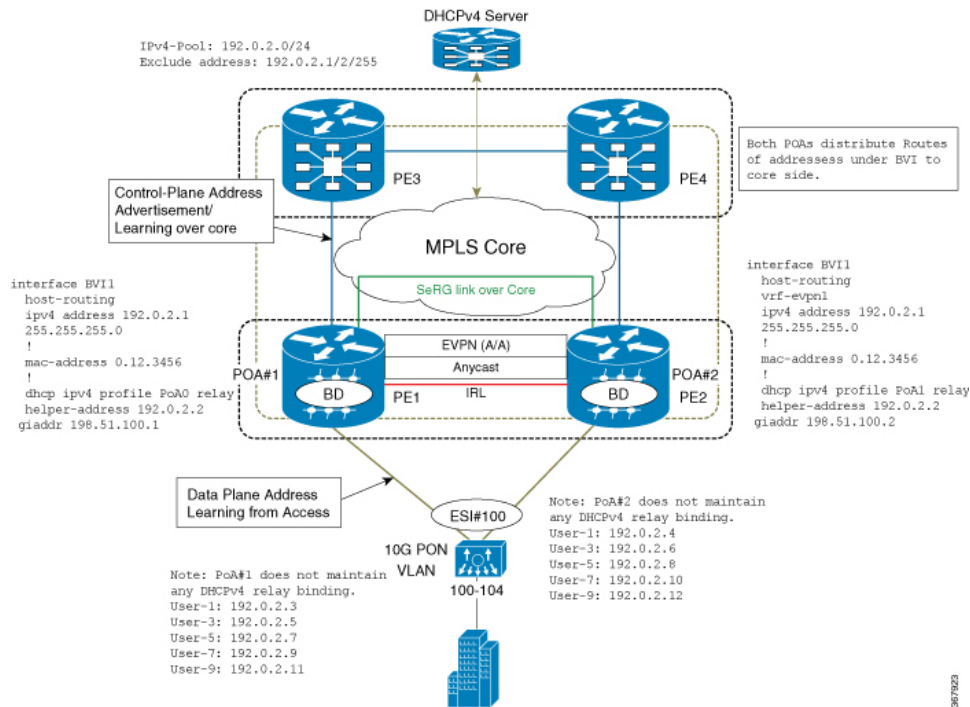
In this topology, PE1 and PE2 are edge routers for access side, which serve CEs (10G-OLT) over BVI interfaces by associating routing and bridging domains to process DHCPv4 packets. CEs (L2 OLT, PONs, any L2 domain switches) hashes the incoming control packets (DHCPv4 packets) towards port channels that 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.

DHCPv4 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 51: DHCPv4 Relay Handling for EVPN and DHCPv4 Server in Default VRF

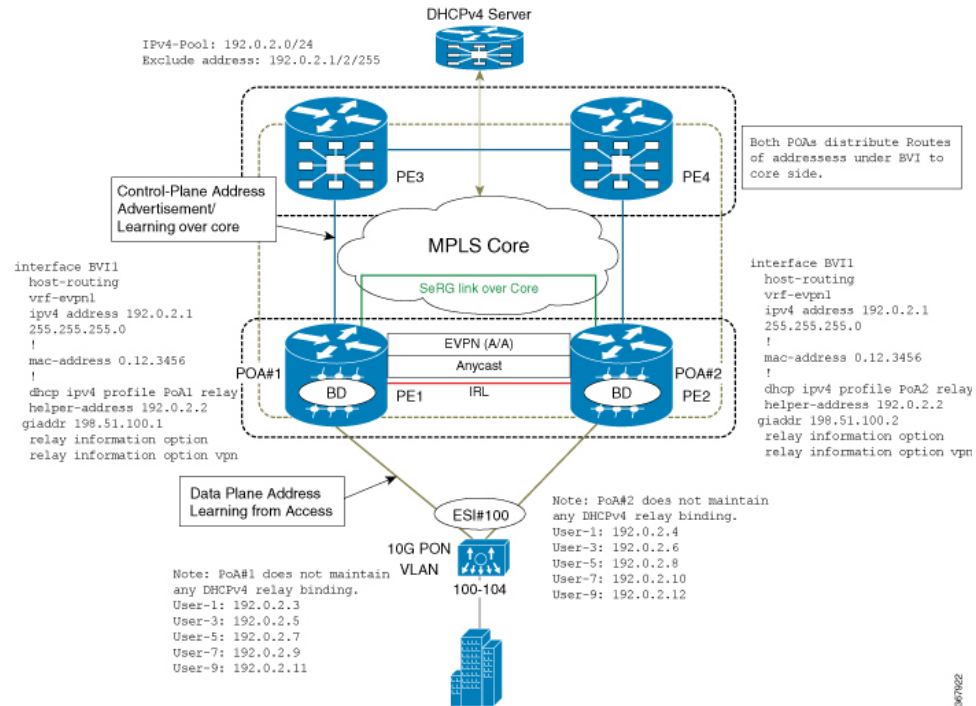


DHCPv4 Relay Handling for EVPN and DHCPv4 Server in Different VRF

DHCPv4 relay over EVPN IRB and DHCPv4 servers reside in different VRFs or DHCPv4 server has a 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 52: DHCPv4 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 ascii 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 BV11
 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 BV11
 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 ipv4 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 318](#)

Associated Commands

- show dhcp vrf default ipv4 relay statistics
- show dhcp ipv4 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.

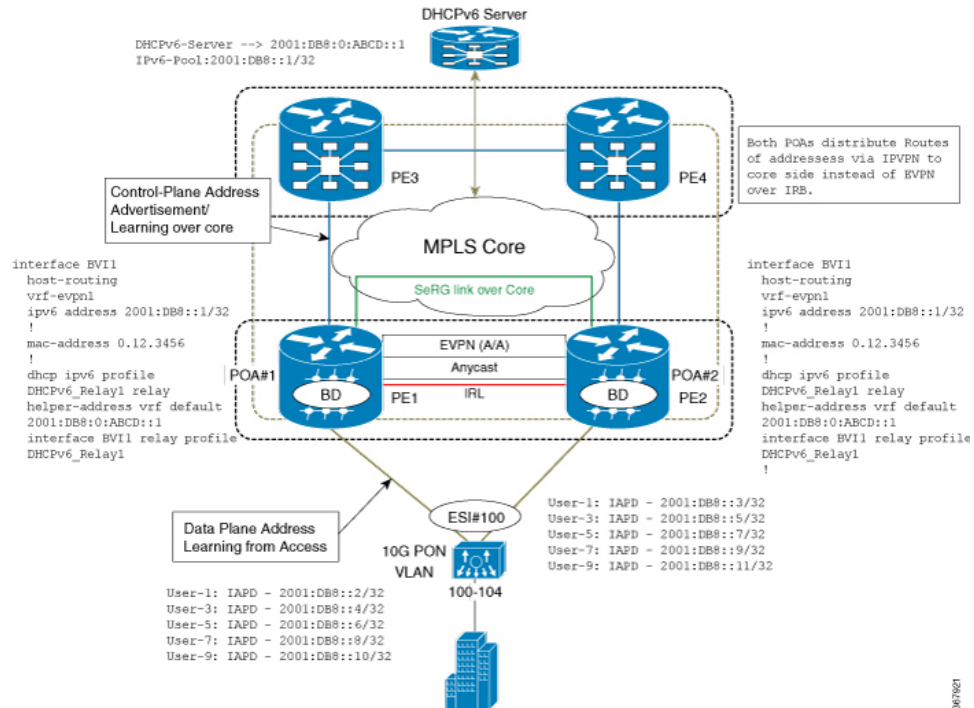
For DHCPv6 relay, access (BVI) interface is tied up with relay profile. Whenever ToRs relay the DHCPv6 packets that are received from client to DHCPv6 server, ToR discovers the best source IP address for a given defined VRF of DHCPv6 server IP address. ToRs maintain unique source IP address for each 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.

DHCPv6 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 DHCPv6-Relayed packets unlike DHCPv4.

Figure 53: DHCPv6 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 BV11 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 BV11
 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 BV11 relay profile DHCPv6_Relay1
 !

/* PE2 Configuration *//interface BV11
 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 BV11 relay profile DHCPv6_Relay1
 !

```

Verification

Verify DHCPv6 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 326](#)

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 suppressed 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.

Configure DHCPv6 PD Synchronization

Perform 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-intf)# 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 329

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 12

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.

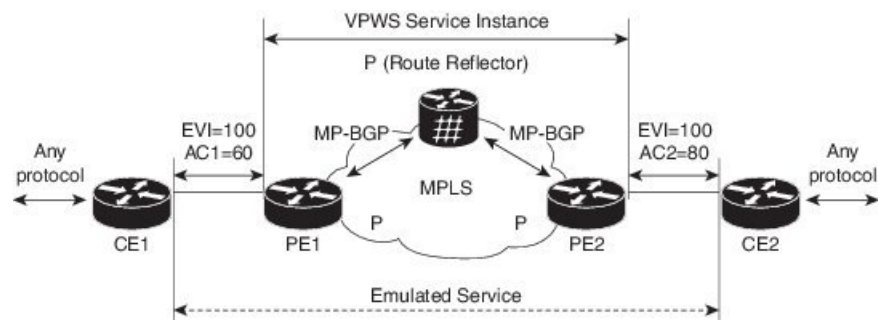
EVPN-VPWS support both single-homing and multi-homing.

- [EVPN-VPWS Single Homed, on page 333](#)
- [EVPN-VPWS Multi-Homed, on page 335](#)
- [Flow Label Support for EVPN VPWS, on page 338](#)

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 advertises 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.

```
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
```

Running Configuration

```
configure
router bgp 100
  address-family l2vpn evpn
  neighbor 10.10.10.1
  address-family l2vpn evpn
!

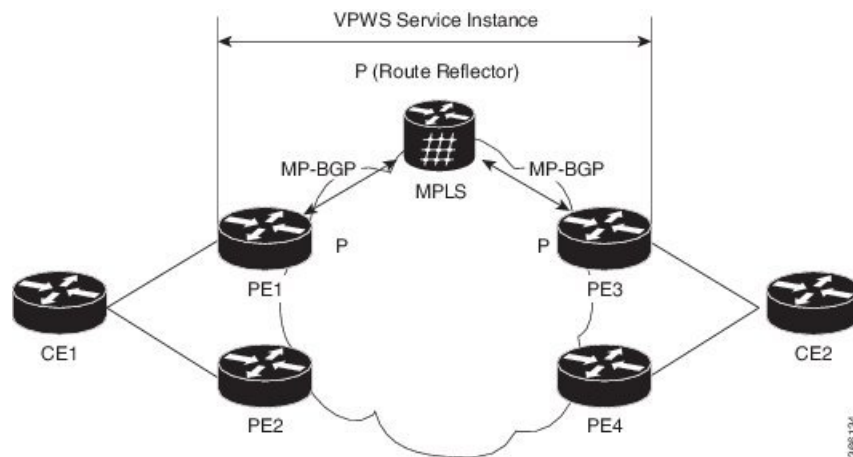
configure
l2vpn
xconnect group evpn-vpws
  p2p evpn1
  interface TenGigE0/1/0/2
  neighbor evpn evi 100 target 12 source 10
!
```

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 54: 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 Multi-Homed

This section describes how you can configure 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
!

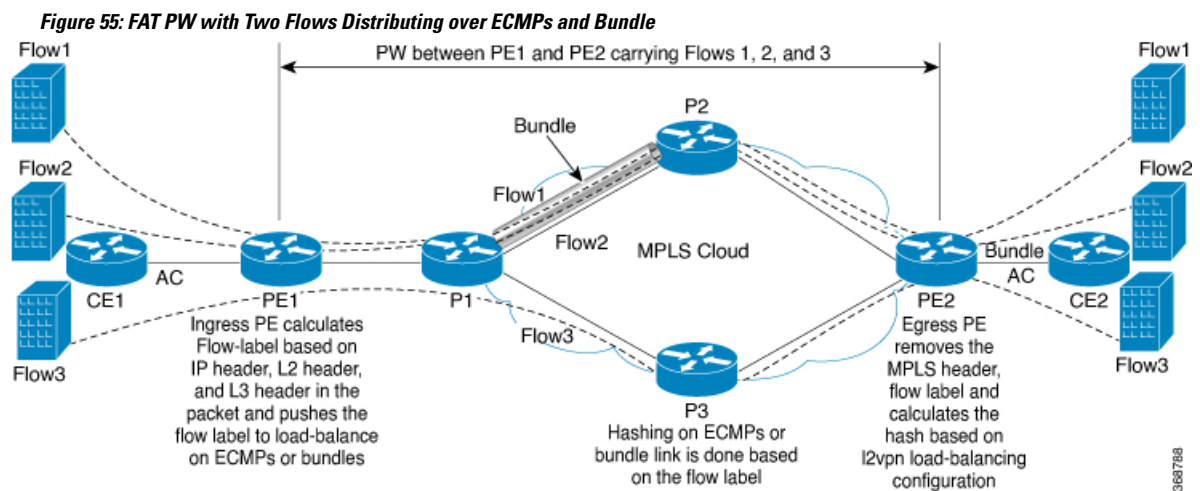
/* On PE4 */
!
configure
l2vpn xconnect group evpn_vpws
p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
  identifier type 0 00.01.00.ac.ce.55.00.14.00
!
```

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.



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)# control-word-disable
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
  control-word-disable
  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 338](#)

Associated Commands

- show evpn evi



CHAPTER 13

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:

- [L2VPN Preferred path, on page 342](#)
- [EVPN VPWS Preferred Path over SR-TE Policy, on page 342](#)
- [L2VPN VPWS Preferred Path over SR-TE Policy, on page 355](#)
- [EVPN VPWS On-Demand Next Hop with SR-TE, on page 368](#)
- [Overview of Segment Routing , on page 382](#)
- [How Segment Routing Works , on page 383](#)
- [Segment Routing Global Block , on page 384](#)

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.

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

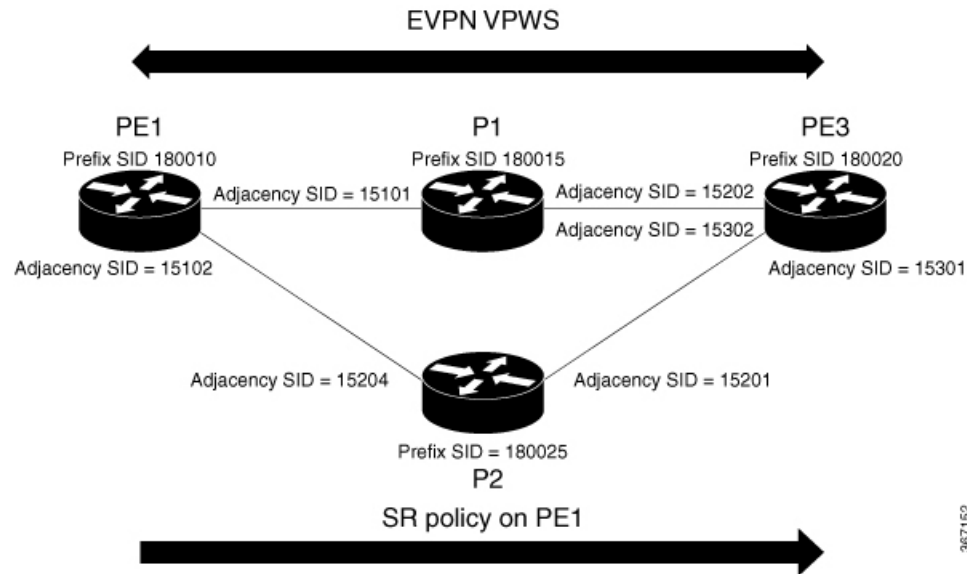
- If EVPN VPWS with On Demand Next Hop (ODN) is configured, and EVPN VPWS with preferred path is also configured for the same PW, then the preferred-path will take precedence.
- EVPN VPWS SR policy is not supported on EVPN VPWS dual homing.
- EVPN validates if the route is for a single home next hop, 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
/* Or globally */
Evpn
  ethernet-segment type 1 auto-generation-disable
```

Topology

Figure 56: 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
Router(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
Router(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
Router(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
Router(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
Router(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
Router(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
Router(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
precedence
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
```

PE2:

```
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          Next Hop          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          Next Hop          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 382
- [How Segment Routing Works](#) , on page 383
- [Segment Routing Global Block](#) , on page 384

L2VPN VPWS Preferred Path over SR-TE Policy

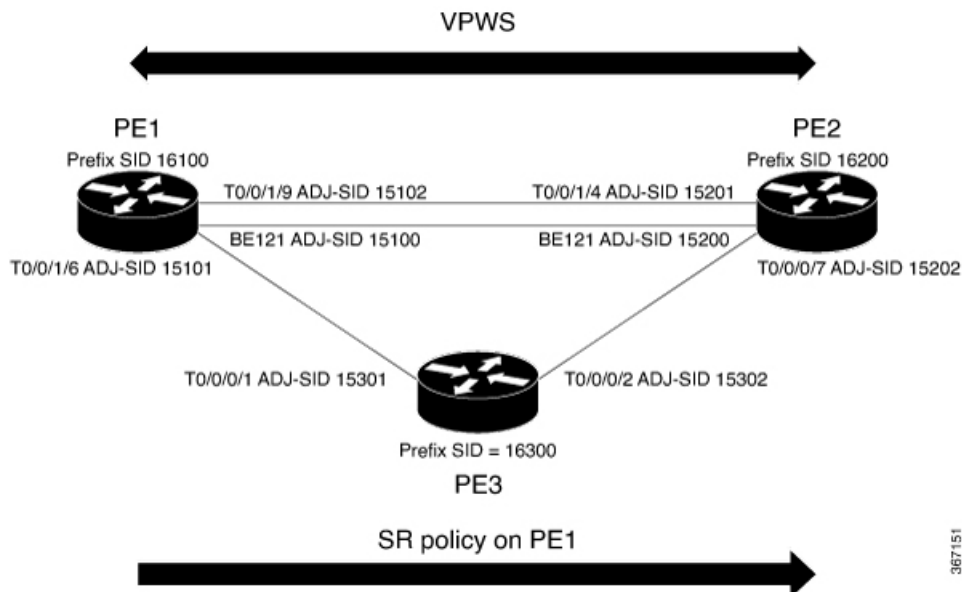
All L2VPN services such as VPLS, VPWS, and so on must use L2VPN preferred-path while using TE (SR-TE, and RSPV-TE) services as transport.

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 57: 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 49.0002.0330.2000.3030.0035.00
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 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 15101
Route(config-isis-if-af) # exit
!
Router# configure
Route(config) # router isis core
Route(config-isis) # interface TenGigE0/0/1/9
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 TenGigE0/0/1/4
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 TenGigE0/0/1/1.1002
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 TenGigE0/0/1/6

  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
    adjacency-sid absolute 15101
  !
interface TenGigE0/0/1/9
  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 TenGigE0/0/1/1.1002
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 Name	Segment List	Outgoing Label	Outgoing Interface	Next Hop	Bytes Switched
100	PE1-PE2	Pop	Te0/0/1/9	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 Name	Segment List	Outgoing Label	Outgoing Interface	Next Hop	Bytes Switched
1013	PE1-PE3-PE2_link	15302	Te0/0/1/6	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 Name	Outgoing Label	Outgoing Interface	Next Hop	Bytes Switched
1013	(SR) 15302	Te0/0/1/6	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 382
- [How Segment Routing Works](#) , on page 383
- [Segment Routing Global Block](#) , on page 384

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
/* 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 58: 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
  !
!

```

PE:

```

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 382
- [How Segment Routing Works](#) , on page 383
- [Segment Routing Global Block](#) , on page 384

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 14

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 385](#)
- [MACsec and MKA Overview, on page 385](#)
- [L2CP Tunneling, on page 386](#)
- [L2CP Tunneling in MACsec, on page 386](#)
- [Configuration , on page 386](#)

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 tunneled 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 BPDUs 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 l2transport
Router(config-if)# encaps 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
```

```

04795B232C766A6C513A5C4E37582F220F0871781167033124465525017A0C7101 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 04795B232C766A6C513A5C4E37582F220F0871781167033124465525017A0C7101
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

!

```

!



CHAPTER 15

References

This section provides additional information on understanding and implementing Layer 2 VPNs.

- [Gigabit Ethernet Protocol Standards, on page 393](#)
- [Carrier Ethernet Model References, on page 393](#)
- [Default Configuration Values for Gigabit Ethernet and 10-Gigabit Ethernet, on page 395](#)
- [References for Configuring Link Bundles, on page 396](#)

Gigabit Ethernet Protocol Standards

The 10-Gigabit Ethernet 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 the point-of-presence (POP), including core and edge routers and L2 and Layer 3 (L3) switches.

The Gigabit Ethernet interfaces in Cisco NCS 5500 Series Routers support these standards:

- Protocol standards:
 - IEEE 802.3 Physical Ethernet Infrastructure
 - IEEE 802.3ae 10 Gbps Ethernet
- Ethernet standards
 - Ethernet II framing also known as DIX
 - IEEE 802.3 framing also includes LLC and LLC/SNAP protocol frame formats
 - IEEE 802.1q VLAN tagging
 - IEEE 802.1ad Provider Bridges

For more information, see [Carrier Ethernet Model References, on page 393](#).

Carrier Ethernet Model References

This topic covers the references for Gigabit Ethernet Protocol Standards.

IEEE 802.3 Physical Ethernet Infrastructure

The IEEE 802.3 protocol standards define the physical layer and MAC sublayer of the data link layer of wired Ethernet. IEEE 802.3 uses Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access at a variety of speeds over a variety of physical media. The IEEE 802.3 standard covers 10 Mbps Ethernet. Extensions to the IEEE 802.3 standard specify implementations for Gigabit Ethernet, 10-Gigabit Ethernet, and Fast Ethernet.

IEEE 802.3ae 10 Gbps Ethernet

Under the International Standards Organization's Open Systems Interconnection (OSI) model, Ethernet is fundamentally a L2 protocol. 10-Gigabit Ethernet uses the IEEE 802.3 Ethernet MAC protocol, the IEEE 802.3 Ethernet frame format, and the minimum and maximum IEEE 802.3 frame size. 10 Gbps Ethernet conforms to the IEEE 802.3ae protocol standards.

Just as 1000BASE-X and 1000BASE-T (Gigabit Ethernet) remained true to the Ethernet model, 10-Gigabit Ethernet continues the natural evolution of Ethernet in speed and distance. Because it is a full-duplex only and fiber-only technology, it does not need the carrier-sensing multiple-access with the CSMA/CD protocol that defines slower, half-duplex Ethernet technologies. In every other respect, 10-Gigabit Ethernet remains true to the original Ethernet model.

General Ethernet Standards

- IEEE 802.1q VLAN tagging—This standard defines VLAN tagging, and also the traditional VLAN trunking between switches. Cisco NCS 5500 Series Routers do NOT support ISL.
- IEEE 802.1ad Provider Bridges—This standard is a subset of 802.1q and is often referred to as 802.1ad. Cisco NCS 5500 Series Routers do not adhere to the entire standard, but large portions of the standard's functionality are supported.

Ethernet MTU

The Ethernet Maximum Transmission Unit (MTU) is the size of the largest frame, minus the 4-byte Frame Check Sequence (FCS), that can be transmitted on the Ethernet network. Every physical network along the destination of a packet can have a different MTU.

Cisco NCS 5500 Series Routers support two types of frame forwarding processes:

- Fragmentation for IPV4 packets—In this process, IPV4 packets are fragmented as necessary to fit within the MTU of the next-hop physical network.



Note IPv6 does not support fragmentation.

- MTU discovery process determines largest packet size—This process is available for all IPV6 devices, and for originating IPV4 devices. In this process, the originating IP device determines the size of the largest IPV6 or IPV4 packet that can be sent without being fragmented. The largest packet is equal to the smallest MTU of any network between the IP source and the IP destination devices. If a packet is larger than the smallest MTU of all the networks in its path, that packet will be fragmented as necessary. This process ensures that the originating device does not send an IP packet that is too large.

Jumbo frame support is automatically enable for frames that exceed the standard frame size. The default value is 1514 for standard frames and 1518 for 802.1Q tagged frames. These numbers exclude the 4-byte FCS.

Flow Control on Ethernet Interfaces

The flow control used on 10-Gigabit Ethernet interfaces consists of periodically sending flow control pause frames. It is fundamentally different from the usual full- and half-duplex flow control used on standard management interfaces. By default, both ingress and egress flow control are off on Cisco NCS 5500 Series Routers.

Default Configuration Values for Gigabit Ethernet and 10-Gigabit Ethernet

The below table describes the default interface configuration parameters that are present when an interface is enabled on a Gigabit Ethernet or 10-Gigabit Ethernet modular services card and its associated PLIM.



Note You must use the **shutdown** command to bring an interface administratively down. The interface default is **no shutdown**. When a modular services card is first inserted into the router, if there is no established preconfiguration for it, the configuration manager adds a shutdown item to its configuration. This shutdown can be removed only by entering the **no shutdown** command.

Table 9: Gigabit Ethernet and 10-Gigabit Ethernet Modular Services Card Default Configuration Values

Parameter	Configuration File Entry	Default Value	Restrictions
Flow control	flow-control	egress on ingress off	none
MTU	mtu	1514 bytes for normal frames 1518 bytes for 802.1Q tagged frames 1522 bytes for QinQ frames	none
MAC address	mac address	Hardware burned-in address (BIA ²)	L3 only
L2 port	l2transport	off/L3	L2 subinterfaces must have L3 main parent interface
Egress filtering	Ethernet egress-filter	off	none
Link negotiation	negotiation	off	physical main interfaces only
Tunneling Ethertype	tunneling ethertype	0X8100	configured on main interface only; applied to subinterfaces only

Parameter	Configuration File Entry	Default Value	Restrictions
VLAN tag matching	encapsulation	all frames for main interface; only ones specified for subinterfaces	encapsulation command only subinterfaces

1. The restrictions are applicable to L2 main interface, L2 subinterface, L3 main interface, interflex L2 interface etc.
2. burned-in address

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 45](#).

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.