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

Notwithstanding any other warranty herein, all document files and software of these suppliers are provided “as is” with all faults. Cisco and the above-named suppliers disclaim all warranties, expressed or implied, including, without limitation, those of merchantability, fitness for a particular purpose and noninfringement or arising from a course of dealing, usage, or trade practice.

In no event shall Cisco or its suppliers be liable for any indirect, special, consequential, or incidental damages, including, without limitation, lost profits or loss or damage to data arising out of the use or inability to use this manual, even if Cisco or its suppliers have been advised of the possibility of such damages.

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

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

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

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: http://www.cisco.com/go/trademarks. 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)

© 2017–2018 Cisco Systems, Inc. All rights reserved.
CONTENTS

PREFACE

Preface xi
Changes to This Document xi
Obtaining Documentation and Submitting a Service Request xi

CHAPTER 1

New and Changed VPN Features 1
New and Changed VPN Features 1

CHAPTER 2

Configure Gigabit Ethernet for Layer 2 VPNs 5
Introduction to Layer 2 Virtual Private Networks 5
Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces 6
Configure Gigabit Ethernet Interfaces for Layer 2 Transport 7
Configure Link Loss Forwarding for Layer 2 Transport 8
Ethernet Data Plane Loopback 9
Configure Ethernet Data Plane Loopback 10
Running Configuration 11
Verification 12
Related Topics 13
Associated Commands 13
Ethernet Local Management Interface (E-LMI) 13
E-LMI Messaging 14
E-LMI Operation 15
Configure Ethernet Local Management Interface (E-LMI) 15
Running Configuration 17
Verify the Ethernet Local Management Interface (E-LMI) Configuration 18

CHAPTER 3

Configure Layer 2 Access Control Lists 21
CHAPTER 7

Configure Multipoint Layer 2 Services 61
Prerequisites for Implementing Multipoint Layer 2 Services 61
Information About Implementing Multipoint Layer 2 Services 62
Multipoint Layer 2 Services Overview 62
Bridge Domain 62
Pseudowires 62
Access Pseudowire is not supported over VPLS Bridge Domain 62
Virtual Forwarding Instance 63
VPLS for an MPLS-based Provider Core 63
VPLS for Layer 2 Switching 63
Interoperability Between Cisco IOS XR and Cisco IOS on VPLS LDP Signaling 64
MAC Address-related Parameters 64
MAC Address Flooding 64
MAC Address-based Forwarding 64
MAC Address Source-based Learning 65
MAC Address Aging 65
MAC Address Limit 65
MAC Address Withdrawal 66
Configuration Examples for Multipoint Layer 2 Services 66
Multipoint Layer 2 Services Configuration for Provider Edge-to-Provider Edge: Example 66
Multipoint Layer 2 Services Configuration for Provider Edge-to-Customer Edge: Example 67
Displaying MAC Address Withdrawal Fields: Example 67
Bridging on IOS XR Trunk Interfaces: Example 69
Bridging on Ethernet Flow Points: Example 73

CHAPTER 8

Configure Point-to-Point Layer 2 Services 77
Ethernet over MPLS 77
Ethernet Port Mode 77
VLAN Mode 78
Inter-AS Mode 79
QinQ Mode 79
QinAny Mode 80
Configure Local Switching Between Attachment Circuits 80
Flexible Cross-Connect Service 84
  Flexible Cross-Connect Service - Single-Homed 85
  Flexible Cross-Connect Service - Multi-Homed 85
Flexible Cross-Connect Service Supported Modes 86
  VLAN Unaware 86
    Configure Single-Homed Flexible Cross-Connect Service using VLAN Unaware 86
    Configure Multi-Homed Flexible Cross-Connect Service using VLAN Unaware 88
  VLAN Aware 91
    Configure Single-Homed Flexible Cross-Connect using VLAN Aware 92
    Configure Multi-Homed Flexible Cross-Connect Service using VLAN Aware 93
  Local Switching 97
    Configure Multi-Homed Flexible Cross-Connect Service using Local Switching 98
Configure Preferred Tunnel Path 100
Multisegment Pseudowire 101
  Multisegment Pseudowire Redundancy 102
  Configure Pseudowire Redundancy 103

CHAPTER 9

EVPN Features 105
  EVPN Overview 105
  EVPN Timers 106
  EVPN Concepts 108
  EVPN Operation 109
  EVPN Route Types 110
  Configure EVPN L2 Bridging Service 111
    Running Configuration 112
  Configure EVPN MAC Address Limit 112
  EVPN Software MAC Learning 115
    Configure EVPN Software MAC Learning 116
    Supported Modes for EVPN Software MAC Learning 116
    Single Home Device or Single Home Network Mode 117
      Configure EVPN in Single Home Device or Single Home Network Mode 117
    Dual Home Device—All-Active Load Balancing Mode 118
      Configure EVPN Software MAC Learning in Dual Home Device—All-Active Mode 119
  Verify EVPN Software MAC Learning 121
Stale Route Deletion after a Host Movement 156
Host Movement Detection through GARP 157
Host Move Detection with Silent Host 157
Host Move Detection without GARP with Data Packet 157
Duplicate MAC Detection 157
Configuring EVPN IRB 157
Running Configuration for EVPN IRB 159
Verify EVPN IRB 160
EVPN IPv6 Hosts with Mobility 170
  Configure EVPN IPv6 Hosts with Mobility 171

CHAPTER 11

EVPN Virtual Private Wire Service (VPWS) 183
  EVPN-VPWS Single Homed 183
    Configure EVPN-VPWS Single Homed 184
      Running Configuration 184
  EVPN-VPWS Multi-Homed 185
    Configure EVPN-VPWS Multi-Homed 185
      Running Configuration 187

CHAPTER 12

L2VPN Services over Segment Routing for Traffic Engineering Policy 189
  EVPN VPWS Preferred Path over SR-TE Policy 190
    Configure EVPN VPWS Preferred Path over SR-TE Policy 191
      Configure Prefix-SID in ISIS 191
      Configure Adjacency-SID in ISIS 193
      Configure Segment-list 195
      Configure SR-TE Policy 195
    Configure EVPN VPWS over SR-TE Policy 196
      Running Configuration 197
    Verify EVPN VPWS Preferred Path over SR-TE Policy Configuration 201
      Associated Commands 202
      Related Topics 202
  L2VPN VPWS Preferred Path over SR-TE Policy 203
    Configure L2VPN VPWS Preferred Path over SR-TE Policy 203
      Configure Prefix-SID in IS-IS 203
CHAPTER 14 References 241

Gigabit Ethernet Protocol Standards 241
Carrier Ethernet Model References 241
Default Configuration Values for Gigabit Ethernet and 10-Gigabit Ethernet 243
References for Configuring Link Bundles 244

Characteristics of Link Bundles 244
Methods of Forming Bundles of Ethernet Interfaces 245
Link Aggregation Through LACP 245
Preface

This preface contains these sections:

- Changes to This Document, on page xi
- Obtaining Documentation and Submitting a Service Request, on page xi

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2017</td>
<td>Initial release of this document.</td>
</tr>
<tr>
<td>March 2018</td>
<td>Republished for Release 6.3.2.</td>
</tr>
</tbody>
</table>

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:


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.
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 6.3.x

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Changed in Release</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet Data Plane Loopback</td>
<td>The Ethernet Data Plane Loopback feature provides a means for remotely testing the throughput of an Ethernet port.</td>
<td>Release 6.3.1</td>
<td>Ethernet Data Plane Loopback, on page 9</td>
</tr>
<tr>
<td>Ethernet Local Management Interface (E-LMI)</td>
<td>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.</td>
<td>Release 6.3.1</td>
<td>Ethernet Local Management Interface (E-LMI), on page 13</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Changed in Release</td>
<td>Where Documented</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>L2CP Tunneling</td>
<td>Layer 2 Control Protocol Tunneling (L2PT) is a Cisco proprietary protocol for tunneling Ethernet protocol frames across layer 2 switching domains. The system supports the following tunnel protocols:</td>
<td>Release 6.3.1</td>
<td>L2CP Tunneling MEF, on page 45</td>
</tr>
<tr>
<td>EVPN IPv6 Hosts with Mobility</td>
<td>EVPN IPv6 Hosts with Mobility feature enables you to provide EVPN IPv6 service over IPv4-MPLS core network.</td>
<td>Release 6.3.2</td>
<td>EVPN IPv6 Hosts with Mobility, on page 170</td>
</tr>
<tr>
<td>EVPN Multiple Services per Ethernet Segment</td>
<td>EVPN Multiple Services per Ethernet Segment (ES) feature allows you to configure multiple services over single ES.</td>
<td>Release 6.3.2</td>
<td>EVPN Multiple Services per Ethernet Segment, on page 126</td>
</tr>
<tr>
<td>EVPN VPWS On-Demand Next Hop with SR-TE</td>
<td>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).</td>
<td>Release 6.3.2</td>
<td>EVPN VPWS On-Demand Next Hop with SR-TE, on page 215</td>
</tr>
<tr>
<td>EVPN VPWS Preferred Path over SR-TE Policy</td>
<td>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.</td>
<td>Release 6.3.2</td>
<td>EVPN VPWS Preferred Path over SR-TE Policy, on page 190</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Changed in Release</td>
<td>Where Documented</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>L2VPN VPLS or VPWS Preferred Path over SR-TE Policy</td>
<td>L2VPN VPLS or VPWS Preferred Path over SR-TE Policy feature allows you to set the preferred path between the two end-points for L2VPN Virtual Private LAN Service (VPLS) or Virtual Private Wire Service (VPWS) using SR-TE policy.</td>
<td>Release 6.3.2</td>
<td>L2VPN VPWS Preferred Path over SR-TE Policy, on page 203</td>
</tr>
</tbody>
</table>
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 5
- Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces, on page 6
- Configure Gigabit Ethernet Interfaces for Layer 2 Transport, on page 7
- Configure Link Loss Forwarding for Layer 2 Transport, on page 8
- Ethernet Data Plane Loopback, on page 9
- Ethernet Local Management Interface (E-LMI), on page 13
- E-LMI Messaging, on page 14
- E-LMI Operation, on page 15
- Configure Ethernet Local Management Interface (E-LMI), on page 15

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 VLAN 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 QRangetinQ sub-interface where outer VLAN range of QRangetinQ overlaps with outer VLAN of QinQ. Attempting this configuration results in the splitting of the existing QinQ and QinQRanget interfaces. However, the system can be recovered by deleting a recently configured QinQRanget interface.

- In an interface which already has QinQRanget configuration, you cannot configure the QRangetinQ sub-interface where outer VLAN range of QRangetinQ overlaps with inner VLAN of QinQRanget. Attempting this configuration results in the splitting of the existing QinQ and QinQRanget interfaces. However, the system can be recovered by deleting a recently configured QinQRanget 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**

```plaintext
/* 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**

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

Verification

Verify that the 10-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.

---

Running Configuration

/* Configuring propagation remote-status */
interface TenGigE 0/0/0/5
The Ethernet Data Plane Loopback feature provides a means for remotely testing the throughput of an Ethernet port. 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. This feature supports two types of Ethernet loopback:

- **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.

The Ethernet Data Plane Loopback feature is not supported on:

- N540-28Z4C-SYS-A
- N540-28Z4C-SYS-D
- N540X-16Z4G8Q2C-A
- N540X-16Z4G8Q2C-D
- N540-12Z20G-SYS-A
- N540-12Z20G-SYS-D
- N540X-12Z16G-SYS-A
- N540X-12Z16G-SYS-D

**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 combination of filters are supported for external loopback:
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 12transport
RP/0/RSP0/CPU0:router(config-subif)# 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 0/2/0/0.1 12transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router(config-subif)# ethernet loopback permit external

/* Starting an Ethernet Data Plane Loopback Session */
configure
interface interface tenGigE 0/0/0/0 12transport
   ethernet loopback permit external

configure
interface interface tenGigE 0/2/0/0/0.1 12transport
   encapsulation dot1q 100

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.

configure
interface interface tenGigE 0/0/0/0 12transport
   ethernet loopback permit external

configure
interface interface tenGigE 0/2/0/0/0.1 12transport
   encapsulation dot1q 100
/* 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

<table>
<thead>
<tr>
<th>Interface</th>
<th>Dot1q(s)</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>tenGigE 0/0/0.1</td>
<td>100</td>
<td>Internal</td>
</tr>
<tr>
<td>tenGigE 0/0/0.1</td>
<td>100</td>
<td>External</td>
</tr>
</tbody>
</table>

/* 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 9

Associated Commands

- ethernet loopback
- show ethernet loopback

Related Topics

- Ethernet Data Plane Loopback, on page 9

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).
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 messages are not supported on:
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)

```bash
/* Configure EVCs for E-LMI */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface TenGigE0/3/0/9.1 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
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 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

!  
12vpn
   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
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
```
Verify the Ethernet Local Management Interface (E-LMI) Configuration

Verify CFM (UP MEP)

RP/0/RSP0/CPU0:router# show ethernet cfm peer meps
Flags:
> - Ok
R - Remote Defect received
L - Loop (our MAC received)
C - Config (our ID received)
X - Cross-connect (wrong MAID)
* - Multiple errors received

Domain irf_evpn_up (level 3), Service up_mep_evpn_1

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
Verify the Ethernet Local Management Interface (E-LMI) Configuration

Up MEP on TenGigE0/3/0/9/1.1 MEP-ID 3001

<table>
<thead>
<tr>
<th>St</th>
<th>ID</th>
<th>MAC Address</th>
<th>Port</th>
<th>Up/Downtime</th>
<th>CcmRcvd</th>
<th>SeqErr</th>
<th>RDI Error</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>1</td>
<td>008a.964b.6410</td>
<td>Up</td>
<td>00:09:59</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Related Topics
- Ethernet Local Management Interface (E-LMI), on page 13
- E-LMI Messaging, on page 14
- E-LMI Messaging, on page 14

Associated Commands
- ethernet lmi
- show ethernet lmi interfaces
- show ethernet cfm peer meps
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 21
- Prerequisites for Configuring Layer 2 Access Control Lists, on page 21
- Layer 2 Access Control Lists Feature Highlights, on page 22
- Purpose of Layer 2 Access Control Lists, on page 22
- How a Layer 2 Access Control List Works, on page 22
- Layer 2 Access Control List Process and Rules, on page 22
- Create Layer 2 Access Control List, on page 23
- Restrictions for Configuring Layer 2 Access Control Lists, on page 23
- Configuration, on page 23

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.

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.

• Only COS (Class of Service) and dei (Discard Eligibility Indication) are supported for Layer 2 access control lists.

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.ff00
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)
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.
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.

Note

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).

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.

- You can program a maximum number of 16 virtual MAC addresses on your router.

Configure VLAN Sub-interfaces

Sub-interfaces 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.

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

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

The sub-interface Maximum Transmission Unit (MTU) is inherited from the physical interface with an additional four bytes allowed for the 802.1Q VLAN tag.

The following modes of VLAN sub-interface configuration are supported:

- Basic dot1q Attachment Circuit
• Q-in-Q Attachment Circuit

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

```bash
capsulation dot1q vlan-id
```

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

```bash
capsulation dot1ad vlan-id
```

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

- ```bash
capsulation dot1q vlan-id second-dot1q vlan-id
```
- ```bash
capsulation dot1ad vlan-id dot1q vlan-id
```

### Restrictions and Limitations

To configure VLAN sub-interface, the following restrictions are applicable.

- For double tagged packet, the VLAN range is supported only on the inner tag.
- VLAN list is not supported.
  
  VLANs separated by comma is called a VLAN list. See the example below.

```bash
Router(config)# interface tenGigE 0/0/0/2.0 l2transport
Router(config-subif)# encapsulation dot1q 1,2  >> VLAN range with comma
Router(config-subif)# commit
```

- If 0x9100/0x9200 is configured as tunneling ether-type, then dot1ad (0x88a8) encapsulation is not supported.
- If any sub-interface is already configured under a main interface, modifying the tunneling ether-type is not supported.
- You can program a maximum number of 16 virtual MAC addresses on your router.

### Configuration Example

Configuring VLAN sub-interface involves:

- Creating a Ten Gigabit Ethernet sub-interface
- 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

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

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

Verification

Verify that the VLAN sub-interface 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,
...

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 31)
- 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 tag(s).

The frames cannot 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.

<table>
<thead>
<tr>
<th>Encapsulation Type</th>
<th>EFP Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single tagged frames</td>
<td>802.1Q customer-tagged Ethernet frames</td>
</tr>
<tr>
<td>Double tagged frames</td>
<td>802.1Q (ethertype 0x9100) double tagged frames</td>
</tr>
<tr>
<td></td>
<td>802.1ad (ethertype 0x9200) double tagged frames</td>
</tr>
</tbody>
</table>

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.
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:

- Valid Ingress Rewrite Actions, on page 34
- Valid Ingress-Egress Rewrite Combinations, on page 35

**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.
- Pop 2 is not supported.
- Translate 2 to 1 is not supported.

**Configuration Example**

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

- L2 single-tagged sub-interface
- L2 double-tagged sub-interface
Configuring VLAN header rewrite involves:

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

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

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

**Running Configuration**

/* Configuration without rewrite */

```
configure
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10 exact
! 
/* Configuration with rewrite */

/* PUSH 1 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10
  rewrite ingress tag push dot1q 20 symmteric
! 
/* POP 1 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10
  rewrite ingress tag pop 1
! 
/* TRANSLATE 1-1 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10
  rewrite ingress tag translate 1-to-1 dot1q 20
! 
/* TRANSLATE 1-2 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10
  rewrite ingress tag translate 1-to-2 dot1q 20 second-dot1q 30
! 
```
Running Configuration (VLAN header rewrite on double-tagged sub-interface)

/* Configuration without rewrite */

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

/* Configuration with rewrite */

/* PUSH 1 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag push dot1q 20 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
### Valid Ingress Rewrite Actions

**Table 3: Valid Ingress Rewrite Actions**

<table>
<thead>
<tr>
<th>Interface Configuration</th>
<th>Ingress Rewrite Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1 to 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1 to 2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1 to 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1 to 2</td>
</tr>
<tr>
<td>QinQ range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ range</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinQ range</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ range</td>
<td>Translate 1 to 1</td>
</tr>
<tr>
<td>QinQ range</td>
<td>Translate 1 to 2</td>
</tr>
<tr>
<td>QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinAny</td>
<td>Translate 1 to 1</td>
</tr>
<tr>
<td>QinAny</td>
<td>Translate 1 to 2</td>
</tr>
</tbody>
</table>
The following notations are used for the rewrite actions mentioned in the table:

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

## Valid Ingress-Egress Rewrite Combinations

### Table 4: Valid Ingress-Egress Rewrite Combinations

<table>
<thead>
<tr>
<th>Ingress Interface Configuration</th>
<th>Ingress Interface Rewrite Action</th>
<th>Egress Interface Configuration</th>
<th>Egress Interface Rewrite Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>dot1q</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>dot1q</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>dot1q</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>dot1q</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>dot1q</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>dot1q</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>dot1q</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>dot1q</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>dot1q</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>dot1q</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>dot1q</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>dot1q</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>dot1q</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>dot1q</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-1</td>
<td>dot1q</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-1</td>
<td>dot1q</td>
<td>Push 1</td>
</tr>
<tr>
<td>Ingress Interface Configuration</td>
<td>Ingress Interface Rewrite Action</td>
<td>Egress Interface Configuration</td>
<td>Egress Interface Rewrite Action</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------------------</td>
<td>--------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-1</td>
<td>dot1q</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>dot1q range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>dot1q range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>dot1q range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>dot1q range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>dot1q range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>dot1q range</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-1</td>
<td>dot1q range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-1</td>
<td>dot1q range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-2</td>
<td>dot1q range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-2</td>
<td>dot1q range</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>QinQ</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Translate 2-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>Ingress Interface Configuration</td>
<td>Ingress Interface Rewrite Action</td>
<td>Egress Interface Configuration</td>
<td>Egress Interface Rewrite Action</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------</td>
<td>--------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 2-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
</tbody>
</table>
## Valid Ingress-Egress Rewrite Combinations

<table>
<thead>
<tr>
<th>Ingress Interface Configuration</th>
<th>Ingress Interface Rewrite Action</th>
<th>Egress Interface Configuration</th>
<th>Egress Interface Rewrite Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>Untagged</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-1</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Translate 1-to-2</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>dot1q range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>dot1q range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>dot1q range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>dot1q range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>dot1q range</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>dot1q range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>QinQ</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
</tbody>
</table>
## Configure Virtual LANs in Layer 2 VPNs

### Valid Ingress-Egress Rewrite Combinations

<table>
<thead>
<tr>
<th>Ingress Interface Configuration</th>
<th>Ingress Interface Rewrite Action</th>
<th>Egress Interface Configuration</th>
<th>Egress Interface Rewrite Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Translate 2-to-2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ</td>
<td>Translate 2-to-2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>Ingress Interface Configuration</td>
<td>Ingress Interface Rewrite Action</td>
<td>Egress Interface Configuration</td>
<td>Egress Interface Rewrite Action</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------</td>
<td>---------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q range</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>dot1q range</td>
<td>Push 2</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ</td>
<td>Translate 2-to-2</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>QinQ</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ</td>
<td>Translate 2-to-2</td>
</tr>
<tr>
<td>Ingress Interface Configuration</td>
<td>Ingress Interface Rewrite Action</td>
<td>Egress Interface Configuration</td>
<td>Egress Interface Rewrite Action</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------</td>
<td>--------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>Ingress Interface Configuration</td>
<td>Ingress Interface Rewrite Action</td>
<td>Egress Interface Configuration</td>
<td>Egress Interface Rewrite Action</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------</td>
<td>-------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>Untagged</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ</td>
<td>Pop 1</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Push 1 / Translate 1-to-1</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ</td>
<td>Push 1 / Translate 1-to-1</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>QinQ</td>
<td>Translate 1-to-2 / Translate 2-to-2</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>Ingress Interface Configuration</td>
<td>Ingress Interface Rewrite Action</td>
<td>Egress Interface Configuration</td>
<td>Egress Interface Rewrite Action</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------</td>
<td>--------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
<td>Untagged</td>
<td>No rewrite</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Push 1 / Translate 1-to-1</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Push 1 / Translate 1-to-1</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
<tr>
<td>Untagged</td>
<td>No rewrite</td>
<td>Untagged</td>
<td>No rewrite</td>
</tr>
<tr>
<td>Untagged</td>
<td>Push 1</td>
<td>Untagged</td>
<td>Push 1</td>
</tr>
<tr>
<td>Untagged</td>
<td>Push 2</td>
<td>Untagged</td>
<td>Push 2</td>
</tr>
</tbody>
</table>

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

- Translate 1-to-1 tag: Translates the outermost tag to another tag
- Translate 1-to-2 tags: Translates the outermost tag to two tags
- Translate 2-to-2 tags: Translates the outermost two tags to two other tags
Valid Ingress-Egress Rewrite Combinations
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 45
- Configure L2CP Tunneling, on page 46

L2CP Tunneling

The system 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)

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 router allows to tunnel layer 2 packets between CEs. The following figure depicts Layer 2 Protocol Tunneling. The layer 2 traffic is sent through the S-network, and the S-network switches the traffic from end-to-end. The Cisco multicast address is added to the frames and sent from UNI to NNI. On the reverse path (NNI to UNI), protocol specific multicast address is attached to the frames and sent to the UNI. Third-party PE forwards S-tagged frames and peers untagged frames.

Figure 2: L2CP Tunneling
Prerequisites for L2CP Tunneling

A Cisco IOS XR Software that supports Layer 2 Control Protocol Tunneling must be installed previously on the router.

Restrictions for L2CP Tunneling

- If you want to peer Operation, Administration, and Maintenance (OAM) packets when the system configures the `l2proto-forward tagged` command at the interface level, you must also configure the `l2protocol peer lacp` command.
- Received L2CP Control Packets (like STP, CDP, and others) are not mirrored to the destination port.
- Forwarding L2CP tunneled packets over local x-connect is not supported.

Configure L2CP Tunneling

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

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

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Packet Type</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDP</td>
<td>Untagged</td>
<td>Peer</td>
</tr>
<tr>
<td>LACP</td>
<td>Untagged</td>
<td>Peer</td>
</tr>
<tr>
<td>LLDP</td>
<td>Untagged</td>
<td>Peer else Tunnelled</td>
</tr>
<tr>
<td>STP</td>
<td>Untagged</td>
<td>Peer</td>
</tr>
<tr>
<td>VTP</td>
<td>Untagged</td>
<td>Peer</td>
</tr>
<tr>
<td>OAM</td>
<td>Untagged</td>
<td>Peer</td>
</tr>
<tr>
<td>BPDU</td>
<td>Untagged</td>
<td>Tunnelled</td>
</tr>
<tr>
<td>CDP</td>
<td>Tagged</td>
<td>Tunnelled</td>
</tr>
<tr>
<td>LACP</td>
<td>Tagged</td>
<td>Tunnelled</td>
</tr>
<tr>
<td>LLDP</td>
<td>Tagged</td>
<td>Tunnelled</td>
</tr>
<tr>
<td>STP</td>
<td>Tagged</td>
<td>Tunnelled</td>
</tr>
<tr>
<td>VTP</td>
<td>Tagged</td>
<td>Tunnelled</td>
</tr>
<tr>
<td>BPDU</td>
<td>Tagged</td>
<td>Tunnelled</td>
</tr>
<tr>
<td>OAM</td>
<td>Tagged</td>
<td>Tunnelled</td>
</tr>
<tr>
<td>ELMI</td>
<td>Tagged</td>
<td>Tunnelled</td>
</tr>
</tbody>
</table>
Complete the following steps to configure L2CP tunneling:

/* Configure Attachment Circuit interface. */
RP/0/RP0/CPU0:ios(config)# int TenGigE0/1/0/8/3
RP/0/RP0/CPU0:ios(config-if)# no shut
RP/0/RP0/CPU0:ios(config-if)# ipv4 addr 13.1.1.1/24
RP/0/RP0/CPU0:ios(config-if)# commit
Fri Sep 1 17:02:57.130 UTC
RP/0/RP0/CPU0:ios(config-if)# int loop 1
RP/0/RP0/CPU0:ios(config-if)# ipv4 addr 2.2.2.6/32
RP/0/RP0/CPU0:ios(config-if)# commit
RP/0/RP0/CPU0:ios(config)# l2vpn
RP/0/RP0/CPU0:ios(config-l2vpn)# xconnect group g1
RP/0/RP0/CPU0:ios(config-l2vpn-xc)# p2p 1
RP/0/RP0/CPU0:ios(config-l2vpn-xc-p2p)# int tenGigE 0/1/0/8/3
RP/0/RP0/CPU0:ios(config-l2vpn-xc-p2p)# neighbor 12.12.12.1 pw-id 1
RP/0/RP0/CPU0:ios(config-l2vpn-xc-p2p-pw)# commit

/* Configure OSPF. */
RP/0/RP0/CPU0:ios(config-if)# router ospf 100 area 0
RP/0/RP0/CPU0:ios(config-ospf-ar)# int TenGigE 0/1/0/8/0
RP/0/RP0/CPU0:ios(config-ospf-ar-if)# int loop 1
RP/0/RP0/CPU0:ios(config-ospf-ar-if)# commit
RP/0/RP0/CPU0:ios(config-ospf-ar-if)# exit
RP/0/RP0/CPU0:ios(config-ospf-ar)# exit
RP/0/RP0/CPU0:ios(config-ospf)# exit

/* Configure MPLS LDP. */
RP/0/RP0/CPU0:ios(config)# mpls ldp
RP/0/RP0/CPU0:ios(config-ldp)# int tenGigE 0/1/0/8/0
RP/0/RP0/CPU0:ios(config-ldp-if)# exit
RP/0/RP0/CPU0:ios(config-ldp)#

Running Configuration
RP/0/RP0/CPU0:ios# show run
Fri Sep 1 17:27:52.682 UTC
Building configuration...
!! IOS XR Configuration version = 6.4.1.111
!! Last configuration change at Fri Sep 1 17:26:37 2017 by root
! telnet vrf default ipv4 server max-servers 10
username root
group root-lr
group cisco-support
secret 5 $1$X9aA$9qdjKAnEbvNG8pfSNbgm/0
! interface Loopback1
ipv4 address 2.2.2.6 255.255.255.255
! interface MgmtEth0/RP0/CPU0/0
ipv4 address 5.10.10.122 255.255.0.0
! interface TenGigE0/1/0/8/0
ipv4 address 13.1.1.1 255.255.255.0
! interface TenGigE0/1/0/8/1
shutdown
! interface TenGigE0/1/0/8/2
shutdown
!
interface TenGigE0/1/0/8/3
l2transport
!
controller Optics0/1/0/8
breakout 4x10
!
interface HundredGigE0/1/0/0
shutdown
!
interface HundredGigE0/1/0/1
shutdown
!
interface HundredGigE0/1/0/2
shutdown
!
interface HundredGigE0/1/0/3
shutdown
!
interface HundredGigE0/1/0/4
shutdown
!
interface HundredGigE0/1/0/5
shutdown
!
interface HundredGigE0/1/0/6
shutdown
!
interface HundredGigE0/1/0/7
shutdown
!
interface HundredGigE0/1/0/9
shutdown
!
interface HundredGigE0/1/0/10
shutdown
!
interface HundredGigE0/1/0/11
shutdown
!
interface HundredGigE0/1/0/12
shutdown
!
interface HundredGigE0/1/0/13
shutdown
!
interface HundredGigE0/1/0/14
shutdown
!
interface HundredGigE0/1/0/15
shutdown
!
interface HundredGigE0/1/0/16
shutdown
!
interface HundredGigE0/1/0/17
shutdown
!
interface HundredGigE0/1/0/18
shutdown
!
interface HundredGigE0/1/0/19
```
shutdown  
!  
interface HundredGigE0/1/0/20  
shutdown  
!  
interface HundredGigE0/1/0/21  
shutdown  
!  
interface HundredGigE0/1/0/22  
shutdown  
!  
interface HundredGigE0/1/0/23  
shutdown  
!  
router static  
address-family ipv4 unicast  
  202.153.144.0/24 5.10.0.1  
!  
router ospf 100  
area 0  
  interface Loopback1  
  !  
  interface TenGigE0/1/0/8/0  
  !  
!  
12vpn  
bridge group b1  
  bridge-domain b1  
  interface TenGigE0/1/0/8/3  
  !  
  vfi vf  
  neighbor 12.12.12.1 pw-id 1  
  !  
  !  
!  
mpls ldp  
interface TenGigE0/1/0/8/0  
!  
!  
end  
```

**Verification**

```
RP/0/RP0/CPU0:ios#show ospf neighbor  
Fri Sep 1 17:24:43.641 UTC  
* Indicates MADJ interface  
# Indicates Neighbor awaiting BFD session up  

Neighbors for OSPF 100  

<table>
<thead>
<tr>
<th>Neighbor ID</th>
<th>Pri</th>
<th>State</th>
<th>Dead Time</th>
<th>Address</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.12.12.1</td>
<td>1</td>
<td>FULL/DR</td>
<td>00:00:31</td>
<td>13.1.1.2</td>
<td>TenGigE0/1/0/8/0</td>
</tr>
</tbody>
</table>

Neighbor is up for 00:21:15

Total neighbor count: 1
RP/0/RP0/CPU0:ios#show mpls ldp neighbor  
Fri Sep 1 17:24:46.602 UTC  
```
Peer LDP Identifier: 12.12.12.1:0
TCP connection: 12.12.12.1:64120 - 2.2.2.6:646
Graceful Restart: No
Session Holdtime: 180 sec
State: Oper; Msgs sent/rcvd: 19/26; Downstream-Unsolicited
Up time: 00:01:46
LDP Discovery Sources:
IPv4: (1)
  TenGigE0/1/0/8/0
IPv6: (0)
Addresses bound to this peer:
IPv4: (8)
  5.5.5.1  5.10.23.254  12.12.12.1  13.1.1.2
  17.1.1.1  88.8.8.8  102.0.0.2  200.169.0.1
IPv6: (0)

BGP neighbor is 12.12.12.1
Remote AS 15169, local AS 15169, internal link
Remote router ID 88.8.8.8
BGP state = Established, up for 00:00:05
NSR State: None
Last read 00:00:00, 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 00:00:00, attempted 29, written 29
Second last write 00:00:05, attempted 19, written 19
Last write before reset 00:00:00, attempted 0, written 0
Second last write before reset 00:00:00, attempted 0, written 0
Last write pulse rcvd Sep 1 17:24:50.144 last full not set pulse count 6
Last write pulse rcvd before reset 00:00:00
Socket not armed for io, armed for read, armed for write
Last write thread event before reset 00:00:00, second last 00:00:00
Last KA expiry before reset 00:00:00, second last 00:00:00
Last KA error before reset 00:00:00, KA not sent 00:00:00
Last KA start before reset 00:00:00, second last 00:00:00
Precedence: internet
Non-stop routing is enabled
Multi-protocol capability received
Neighbor capabilities:
  Route refresh: advertised (old + new) and received (old + new)
  4-byte AS: advertised and received
  Address family L2VPN VPLS: advertised and received
Received 3 messages, 0 notifications, 0 in queue
Sent 3 messages, 0 notifications, 0 in queue
Minimum time between advertisement runs is 0 secs
Inbound message logging enabled, 3 messages buffered
Outbound message logging enabled, 3 messages buffered

For Address Family: L2VPN VPLS
BGP neighbor version 1
Update group: 0.2 Filter-group: 0.1 No Refresh request being processed
NEXT_HOP is always this router
Route refresh request: received 0, sent 0
0 accepted prefixes, 0 are bestpaths
Exact no. of prefixes denied : 0.
Cumulative no. of prefixes denied: 0.
Prefix advertised 0, suppressed 0, withdrawn 0
Maximum prefixes allowed 2097152
Threshold for warning message 75%, restart interval 0 min
AIGP is enabled
An EoR was received during read-only mode
Last ack version 1, Last synced ack version 0
Outstanding version objects: current 0, max 0
Additional-paths operation: None
Send Multicast Attributes

Connections established 1; dropped 0
Local host: 2.2.2.6, Local port: 34285, IF Handle: 0x00000000
Foreign host: 12.12.12.1, Foreign port: 179
Last reset 00:00:00
RP/0/RP0/CPU0:ios#

RP/0/RP0/CPU0:ios#show l2vpn bridge-domain
Fri Sep 1 17:27:25.002 UTC
Legend: pp = Partially Programmed.
Bridge group: b1, bridge-domain: b1, id: 0, 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: 1, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
List of ACs:
Te0/1/0/8/3, state: up, Static MAC addresses: 0
List of VFIs:
List of Access PWs:

RP/0/RP0/CPU0:ios#show l2vpn xconnect
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

<table>
<thead>
<tr>
<th>XConnect</th>
<th>Segment 1</th>
<th>Segment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Name</td>
<td>ST Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>UP Te0/1/0/8/3</td>
</tr>
</tbody>
</table>

RP/0/RP0/CPU0:ios#show l2vpn xconnect
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

<table>
<thead>
<tr>
<th>XConnect</th>
<th>Segment 1</th>
<th>Segment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Name</td>
<td>ST Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>UP Te0/1/0/8/3</td>
</tr>
</tbody>
</table>
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 53
• Configure VLAN Bundle, on page 56
• References for Configuring Link Bundles, on page 57

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.

**Configuration Example**

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

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

Sample values are provided in the following figure.

*Figure 3: Link Bundle Topology*

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

**Configuration**

```plaintext
/* 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)# 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 |

<table>
<thead>
<tr>
<th>Port</th>
<th>Device</th>
<th>State</th>
<th>Port ID</th>
<th>B/W, kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Te1/0/0/0</td>
<td>Local</td>
<td>Active</td>
<td>0x8000, 0x0000</td>
<td>10000000</td>
</tr>
<tr>
<td>Te1/0/0/1</td>
<td>Local</td>
<td>Active</td>
<td>0x8000, 0x0000</td>
<td>10000000</td>
</tr>
<tr>
<td>Te1/0/0/2</td>
<td>Local</td>
<td>Active</td>
<td>0x8000, 0x0000</td>
<td>10000000</td>
</tr>
</tbody>
</table>

Associated Commands

- bundle maximum-active links
- interface 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**

```plaintext
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
\!
```

- show bundle Bundle-Ether
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 53.
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.

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.
Link Aggregation Through LACP
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).

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 61
- Information About Implementing Multipoint Layer 2 Services, on page 62
- Configuration Examples for Multipoint Layer 2 Services, on page 66

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.

  The loopback interface is not needed in all cases. For example, tunnel selection does not need a loopback interface when Multipoint Layer 2 Services are directly mapped to a TE tunnel.
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.

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.

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 is not supported over VPLS Bridge Domain

Access PW is not supported over VPLS bridge domain. Only core PW which is configured under VFI is supported.

Configuration Example

```
12vpn
  bridge group bg1
  bridge-domain 12vpn
```
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.
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.

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:

```plaintext
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
```

**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.

MAC Address Limit

The MAC address limit is used to limit the number of learned MAC addresses.

When a limit is exceeded, the system is configured to perform these notifications:

- Syslog (default)
- Simple Network Management Protocol (SNMP) trap
- Syslog and SNMP trap
- None (no notification)

To generate syslog messages and SNMP trap notifications, use the `mac limit notification both` command in the L2VPN bridge-domain configuration mode.

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.

**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.

**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 10.2.2.2 pw-id 1
    neighbor 10.3.3.3 pw-id 1

interface loopback 0
  ipv4 address 10.1.1.1 255.255.255.255
```

This configuration example shows how to configure PE 2:
Configure Multipoint Layer 2 Services

configure
l2vpn
bridge group 1
  bridge-domain PE2-VPLS-A
    interface TenGigE0/0/0/1
      vfi 1
        neighbor 10.1.1.1 pw-id 1
        neighbor 10.3.3.3 pw-id 1
      !
    !
  interface loopback 0
    ipv4 address 10.2.2.2 255.255.255.25

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 10.1.1.1 pw-id 1
        neighbor 10.2.2.2 pw-id 1
      !
    !
  interface loopback 0
    ipv4 address 10.3.3.3 255.255.255.25

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 FW: 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), FBBs: 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 1.1.1.1, 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

<table>
<thead>
<tr>
<th>MPLS</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>24017</td>
<td>24010</td>
</tr>
<tr>
<td>Group ID</td>
<td>0x0</td>
<td>0x0</td>
</tr>
<tr>
<td>Interface</td>
<td>222</td>
<td>222</td>
</tr>
<tr>
<td>MTU</td>
<td>1500</td>
<td>1500</td>
</tr>
<tr>
<td>Control word disabled</td>
<td>disabled</td>
<td>disabled</td>
</tr>
<tr>
<td>PW type</td>
<td>Ethernet</td>
<td>Ethernet</td>
</tr>
<tr>
<td>VCCV CV type</td>
<td>0x2</td>
<td>0x2</td>
</tr>
<tr>
<td>(LSP ping verification)</td>
<td>(LSP ping verification)</td>
<td></td>
</tr>
<tr>
<td>VCCV CC type</td>
<td>0x6</td>
<td>0x6</td>
</tr>
<tr>
<td>(router alert label)</td>
<td>(router alert label)</td>
<td></td>
</tr>
<tr>
<td>(TTL expiry)</td>
<td>(TTL expiry)</td>
<td></td>
</tr>
</tbody>
</table>

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-l2)#l2vpn
RP/0/RSP0/CPU0:router(config-if-l2)#bridge group examples
RP/0/RSP0/CPU0:router(config-if-l2)#bridge-domain test-switch
RP/0/RSP0/CPU0:router(config-if-l2-bd)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-if-l2-bd)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-if-l2-bd)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-if-l2-bd)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-if-l2-bd)#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-if-l2-bd-ac)#end
RP/0/RSP0/CPU0:Jun 26 10:48:21.342 EDT: config[65751]: %MGBL-SYS-5-CONFIG_I : Configured from console by lab
```

Bundle-Ether10

<table>
<thead>
<tr>
<th>Status</th>
<th>Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local links &lt;active/standby/configured&gt;</td>
<td>0 / 0 / 2</td>
</tr>
<tr>
<td>Local bandwidth &lt;effective/available&gt;</td>
<td>0 (0) kbps</td>
</tr>
<tr>
<td>MAC address (source)</td>
<td>0024.f71e.22eb (Chassis pool)</td>
</tr>
<tr>
<td>Minimum active links / bandwidth</td>
<td>1 / 1 kbps</td>
</tr>
<tr>
<td>Maximum active links</td>
<td>64</td>
</tr>
<tr>
<td>Wait while timer</td>
<td>2000 ms</td>
</tr>
<tr>
<td>LACP</td>
<td>Operational</td>
</tr>
<tr>
<td>mLACP</td>
<td>Not configured</td>
</tr>
<tr>
<td>IPV4 BFD</td>
<td>Not configured</td>
</tr>
</tbody>
</table>

### Port | Device | State | Port ID | B/W, kbps
--- | --- | --- | --- | ---
G10/2/0/5 | Local | Configured | 0x8000, 0x0001 | 1000000
G10/2/0/6 | Local | Configured | 0x8000, 0x0002 | 1000000
Link is down

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), VFI: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
List of ACSs:
  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:

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
En ters 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

```plaintext
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)#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-bd)#interface Bundle-ether10.999
RP/0/RSP1/CPU0:router(config-l2vpn-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bd)#interface GigabitEthernet0/6/0/7.999
RP/0/RSP1/CPU0:router(config-l2vpn-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bd)#interface TenGigE0/1/0/2.999
RP/0/RSP1/CPU0:router(config-l2vpn-bd-ac)#commit
RP/0/RSP1/CPU0:router(config-l2vpn-bd-ac)#end
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  \textbf{interface TenGigE0/1/0/2.999 l2transport}
Creates an EFP under TenGigE0/1/0/2.

Step 12  \textbf{encapsulation dot1q 999}
Assigns VLAN ID of 999 to this EFP.

Step 13  \textbf{l2vpn}
Enters L2VPN configuration mode.

Step 14  \textbf{bridge group examples}
Creates the bridge group named \texttt{examples}.

Step 15  \textbf{bridge-domain test-efp}
Creates the bridge domain named \texttt{test-efp}, that is a member of bridge group \texttt{examples}.

Step 16  \textbf{interface Bundle-ether10.999}
Establishes Bundle-ether10.999 as an AC of the bridge domain named \texttt{test-efp}.

Step 17  \textbf{exit}
Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 18  \textbf{interface GigabitEthernet0/6/0/7.999}
Establishes GigabitEthernet0/6/0/7.999 as an AC of the bridge domain named \texttt{test-efp}.

Step 19  \textbf{exit}
Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 20  \textbf{interface TenGigE0/1/0/2.999}
Establishes interface TenGigE0/1/0/2.999 as an AC of bridge domain named \texttt{test-efp}.

Step 21  Use the \texttt{commit} or \texttt{end} command.
\texttt{commit} - Saves the configuration changes and remains within the configuration session.
\texttt{end} - Prompts user to take one of these actions:
  - \texttt{Yes} - Saves configuration changes and exits the configuration session.
  - \texttt{No} - Exits the configuration session without committing the configuration changes.
  - \texttt{Cancel} - Remains in the configuration mode, without committing the configuration changes.
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.

Point-to-point Layer 2 services are also called as MPLS Layer 2 VPNs.

- Ethernet over MPLS, on page 77
- Configure Local Switching Between Attachment Circuits, on page 80
- Flexible Cross-Connect Service, on page 84
- Flexible Cross-Connect Service Supported Modes, on page 86
- Configure Preferred Tunnel Path, on page 100
- Multisegment Pseudowire, on page 101
- Configure Pseudowire Redundancy, on page 103

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 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 4: 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 5: 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.

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 NCS500x Series Routers implement the Layer 2 tunneling or Layer 3 forwarding depending on the sub-interface configuration at provider edge routers. This function only supports up to two QinQ tags on the router:

- Layer 2 QinQ VLANs in L2VPN attachment circuit: QinQ L2VPN attachment circuits are configured under the Layer 2 transport sub-interfaces for point-to-point EoMPLS based cross-connects using both virtual circuit type 4 and type 5 pseudowires and point-to-point local-switching-based cross-connects including full inter-working support of QinQ with 802.1q VLANs and port mode.

- Layer 3 QinQ VLANs: Used as a Layer 3 termination point, both VLANs are removed at the ingress provider edge and added back at the remote provider edge as the frame is forwarded.

Layer 3 services over QinQ include:

- IPv4 unicast and multicast
- IPv6 unicast and multicast
- MPLS
- Connectionless Network Service (CLNS) for use by Intermediate System-to-Intermediate System (IS-IS) Protocol
In QinQ mode, each CE VLAN is carried into an SP VLAN. QinQ mode should use VC type 5, but VC type 4 is also supported. On each Ethernet PE, you must configure both the inner (CE VLAN) and outer (SP VLAN).

The following figure illustrates QinQ using VC type 4.

**Figure 7: 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 8: 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 0/0/0/1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# exit
Configure Local Switching Between Attachment Circuits

/* Configure ethernet link bundles */
Router(config)# 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)# exit
Router(config)# interface TenGigE 0/0/0/2
Router(config-if)# bundle id 3 mode on
Router(config)# exit
Router(config)# interface TenGigE 0/0/0/9
Router(config-if)# bundle id 2 mode on
Router(config)# exit
Router(config)# interface TenGigE 0/0/0/8
Router(config-if)# bundle id 2 mode on
Router(config)# 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-l2)# 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)# 12vpn
Router(config-12vpn)# xconnect group XCON1
Router(config-12vpn-xc)# p2p XCON1_P2P3
Router(config-12vpn-xc-p2p)# interface TenGigE0/0/0/1
Router(config-12vpn-xc-p2p)# interface TenGigE0/0/0/9
Router(config-12vpn-xc-p2p)# commit
Router(config-12vpn-xc-p2p)# exit

/* Configure local switching on the VLAN sub-interfaces */
Router(config-12vpn-xc)# p2p XCON1_P2P1
Router(config-12vpn-xc-p2p)# interface TenGigE0/0/0.1
Router(config-12vpn-xc-p2p)# interface TenGigE0/0/0/8.1
Router(config-12vpn-xc-p2p)# commit
Router(config-12vpn-xc-p2p)# exit
Configure Point-to-Point Layer 2 Services

Configure Local Switching Between Attachment Circuits

/* 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 0/0/0/1 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

<table>
<thead>
<tr>
<th>Like-to-Like</th>
<th>UP</th>
<th>DOWN</th>
<th>UNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>XConnect</th>
<th>Group Name</th>
<th>ST</th>
<th>Description</th>
<th>ST</th>
<th>Description</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>XCON1</td>
<td>XCON_P2P1</td>
<td>UP</td>
<td>Te0/0/0/1</td>
<td>UP</td>
<td>Te0/0/0/9</td>
<td>UP</td>
</tr>
<tr>
<td>XCON1</td>
<td>XCON_P2P3</td>
<td>UP</td>
<td>Te0/0/0/0.1</td>
<td>UP</td>
<td>Te0/0/0/8.1</td>
<td>UP</td>
</tr>
</tbody>
</table>

**Associated Commands**

- `interface (p2p)`
- `l2vpn`
- `p2p`
- `xconnect group`

**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 9: 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 10: 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 symmetric
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 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# 12vpn
Router(config-12vpn)# flexible-xconnect-service vlan-unaware fxs1
Router(config-12vpn-fxs-vu)# interface GigabitEthernet 0/2/0/3.1
Router(config-12vpn-fxs-vu)# interface GigabitEthernet 0/2/0/0.1
Router(config-12vpn-fxs-vu)# neighbor evpn evi 1 target 1
Router(config-12vpn-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 symmetric

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
Running Configuration

/* On PE1 */
!
Configure
interface GigabitEthernet 0/2/0/3.1 l2transport
en encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 symmetric
!
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 GigabitEthernet 0/2/0/0.1
  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 GigabitEthernet 0/0/0/0.1
  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 fxcl_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-l2vpn)# 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-l2vpn)# 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-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# 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 fxcl_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)# 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-l2vpn)# 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-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# 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 fxcl_16
Router(config-l2vpn-fxs)# 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)# 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)# 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-l2vpn-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)# 12vpn
Router(config-12vpn)# flexible-xconnect-service vlan-unaware fxcl_16
Router(config-12vpn-fxs)# interface Bundle-Ether20.11
Router(config-12vpn-fxs)# interface Bundle-Ether20.12
Router(config-12vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-12vpn-fxs)# commit
Router(config-12vpn-fxs)# exit
Router(config)# interface Bundle-Ether20.11 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 1
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-subif)# exit
Router(config)# interface Bundle-Ether20.12 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 2
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-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
12vpn
flexible-xconnect-service vlan-unaware fxcl_16
interface Bundle-Ether10.11
interface Bundle-Ether10.12
neighbor evpn evi 1 target 16

!}

configure
interface Bundle-Ether10.11 l2transport
encapsulation dot1q 1
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 fxcl_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 fxcl_16
  interface Bundle-Ether20.11
  interface Bundle-Ether20.12
  neighbor evpn evi 1 target 16
!

configure
interface Bundle-Ether20.11 l2transport
  encapsulation dot1q 1
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.

```cisco
/* Configure PE1 */
Router# configure
Router(config)# interface GigabitEthernet 0/2/0/7.1 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 1
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-subif)# exit
Router(config)# interface GigabitEthernet 0/2/0/7.2 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 2
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-subif)# exit
Router(config)# l2vpn
Router(config-12vpn)# flexible-xconnect-service vlan-aware evi 4
Router(config-12vpn-fxs)# interface GigabitEthernet 0/2/0/7.1
Router(config-12vpn-fxs)# interface GigabitEthernet 0/2/0/7.2
Router(config-12vpn-fxs)# commit

/* Configure PE2 */
Router# configure
Router(config)# interface GigabitEthernet 0/0/0/7.1 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 1
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-subif)# exit
Router(config)# interface GigabitEthernet 0/0/0/7.2 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 2
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-subif)# exit
Router(config)# l2vpn
Router(config-12vpn)# flexible-xconnect-service vlan-aware evi 4
Router(config-12vpn-fxs)# interface GigabitEthernet 0/0/0/7.1
Router(config-12vpn-fxs)# interface GigabitEthernet 0/0/0/7.2
Router(config-12vpn-fxs)# commit
```

**Running Configuration**

```cisco
/* 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 symmetric
!

Configure
interface GigabitEthernet 0/2/0/7.2 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symmetric
```
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)# interface Bundle-Ether2.1
Router(config-l2vpn-fxs)# interface Bundle-Ether3.1
Router(config-l2vpn-fxs)# 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-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

Configure Multi-Homed Flexible Cross-Connect Service using VLAN Aware
Configure Multi-Homed Flexible Cross-Connect Service using VLAN Aware

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)# 12vpn
Router(config-12vpn)# flex-xconnect-service vlan-aware evi 6
Router(config-12vpn-fxs)# interface Bundle-Ether2.1
Router(config-12vpn-fxs)# interface Bundle-Ether3.1
Router(config-12vpn-fxs)# commit
Router(config-12vpn-fxs)# exit
Router(config-12vpn)# exit
Router(config)# interface Bundle-Ether2.1 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 1
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 2
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-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)# 12vpn
Router(config-12vpn)# flex-xconnect-service vlan-aware evi 6
Router(config-12vpn-fxs)# interface Bundle-Ether4.1
Router(config-12vpn-fxs)# interface Bundle-Ether5.1
Router(config-12vpn-fxs)# commit
Router(config-12vpn-fxs)# exit
Router(config-12vpn)# exit
Router(config)# interface Bundle-Ether4.1 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 1
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-subif)# exit
Router(config)# interface Bundle-Ether5.1 l2transport
Router(config-12vpn-subif)# encapsulation dot1q 2
Router(config-12vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-12vpn-subif)# commit
Router(config-12vpn-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)# interface Bundle-Ether5
Router(config-evpn-ac)# ethernet-segment

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
Configure Point-to-Point Layer 2 Services

Running Configuration

/* Configure PE4 */
Router# configure
Router(config)# configure
Router(config)# 12vpn
Router(config-12vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-12vpn)# interface Bundle-Ether4.1
Router(config-12vpn)# interface Bundle-Ether5.1
Router(config-12vpn)# commit
Router(config-12vpn)# exit
Router(config)# interface Bundle-Ether4.1 12transport
Router(config)# encapsulation dot1q 1
Router(config)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config)# commit
Router(config)# exit
Router(config)# interface Bundle-Ether5.1 12transport
Router(config)# encapsulation dot1q 2
Router(config)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config)# commit
Router(config)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether4
Router(config-evpn-ac-es)# 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)# interface Bundle-Ether5
Router(config-evpn-ac-es)# 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

Running Configuration

/* On PE1 */
!
configure
12vpn
 flexible-xconnect-service vlan-aware evi 6
 interface Bundle-Ether2.1
 interface Bundle-Ether3.1
!

configure
interface Bundle-Ether2.1 12transport
 encapsulation dot1q 1
 rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
!

configure
interface Bundle-Ether3.1 12transport
 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
/* 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 11: Local Switching

![Local Switching Diagram]
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)# interface Bundle-Ether2.1
Router(config-l2vpn-fxs)# interface Bundle-Ether3.1
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether2.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric
Router(config-subif)# commit
Router(config-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)# interface Bundle-Ether2.1
Router(config-l2vpn-fxs)# interface Bundle-Ether3.1
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether2.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric
Router(config-subif)# commit
Router(config-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
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
### 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)# 12vpn

/* Configure pseudowire class name */
Router(config-12vpn)# pw-class path1

/* Configure MPLS encapsulation for the pseudowire */
Router(config-12vpn-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-12vpn-pwc-encap-mpls)# preferred-path

interface tunnel-te 11 fallback disable

/* Commit your configuration */
Router(config-12vpn-pwc)# exit
Router(config-12vpn)# 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.
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.
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 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.
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)# backup neighbor 10.2.2.2 pw-id 5

/*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
...
12vpn
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
    !
    !
...

CHAPTER 9

EVPN Features

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

• EVPN Overview, on page 105
• EVPN Concepts, on page 108
• EVPN Operation, on page 109
• EVPN Route Types, on page 110
• Configure EVPN L2 Bridging Service, on page 111
• Configure EVPN MAC Address Limit, on page 112
• EVPN Software MAC Learning, on page 115
• EVPN Out of Service, on page 123
• EVPN Multiple Services per Ethernet Segment, on page 126
• EVPN Routing Policy, on page 132

EVPN Overview

Ethernet VPN (EVPN) is a next generation 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 multi-homing 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 multi-homing 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 - This enables you to connect a customer edge (CE) device to one provider edge (PE) device.

• Multihoming - This 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 Timers**

The following table shows various EVPN timers:
### Table 5: EVPN Timers

<table>
<thead>
<tr>
<th>Timer</th>
<th>Range</th>
<th>Default Value</th>
<th>Trigger</th>
<th>Applicability</th>
<th>Action</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>startup-cost-in</td>
<td>30-86400</td>
<td>disabled</td>
<td>node recovered*</td>
<td>Single-Homed, All-Active, Single-Active</td>
<td>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.</td>
<td>1</td>
</tr>
<tr>
<td>recovery</td>
<td>20-3600s</td>
<td>30s</td>
<td>node recovered, interface recovered**</td>
<td>Single-Homed***, Single-Active</td>
<td>Postpone EVPN Startup procedure. Recovery timer allows PE to set access protocols (STP) before reachability towards EVPN core is advertised.</td>
<td>2</td>
</tr>
<tr>
<td>peering</td>
<td>0-3600s</td>
<td>3s</td>
<td>node recovered, interface recovered</td>
<td>All-Active, Single-Active</td>
<td>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.</td>
<td>3</td>
</tr>
</tbody>
</table>
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.

Note

* 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.

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 14: 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 6: EVPN Route Types

<table>
<thead>
<tr>
<th>Route Type</th>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ethernet Auto-Discovery (AD) Route</td>
<td>Few routes are sent per ES, carries the list of EVIs that belong to ES</td>
</tr>
<tr>
<td>2</td>
<td>MAC/IP Advertisement Route</td>
<td>Advertise MAC, address reachability, advertise IP/MAC binding</td>
</tr>
<tr>
<td>3</td>
<td>Inclusive Multicast Ethernet Tag Route</td>
<td>Multicast Tunnel End point discovery</td>
</tr>
<tr>
<td>4</td>
<td>Ethernet Segment Route</td>
<td>Redundancy group discovery, DF election</td>
</tr>
<tr>
<td>5</td>
<td>IP Prefix Route</td>
<td>Advertise IP prefixes.</td>
</tr>
</tbody>
</table>

Route Type 1: Ethernet Auto-Discovery (AD) Route

The Ethernet Auto-Discovery (AD) routes are advertised on per EVI and per ESI basis. These routes are sent per ES. They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed. This route type is used for mass withdrawal of MAC addresses and aliasing for load balancing.
Route Type 2: MAC/IP Advertisement Route

These routes are per-VLAN routes, so only PEs that are part of a VNI require these routes. The host's IP and MAC addresses are advertised to the peers within NRLI. The control plane learning of MAC addresses reduces unknown unicast flooding.

Route Type 3: Inclusive Multicast Ethernet Tag Route

This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.

Route Type 4: Ethernet Segment Route

Ethernet segment routes enable to connect a CE device to two or PE devices. ES route enables the discovery of connected PE devices that are connected to the same Ethernet segment.

Route Type 5: IP Prefix Route

The IP prefixes are advertised independently of the MAC-advertised routes. With EVPN IRB, host route /32 is advertised using RT-2 and subnet /24 is advertised using RT-5.

Note

With EVPN IRB, host route /32 are advertised using RT-2 and subnet /24 are advertised using RT-5.

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

Flooding disable is not 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)# neighbor 10.10.10.10 remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn

/* Configure EVI and define the corresponding BGP route targets */

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)# bridge-domain 1-1
Router(config-l2vpn-bd)# interface GigabitEthernet 0/0/0/1.1
Router(config-l2vpn-bd-ac)# evi 6005
Router(config-l2vpn-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 12vpn evpn
    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
```

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 action is applied only when the number of local MAC addresses exceeds the configured limit. If the MAC address limit is exhausted by remote and local MAC addresses, or only by remote MAC addresses, the MAC address limit action is not applied. The surpassed traffic is treated as unknown unicast and is flooded across the bridge domain.

- 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.

### Configuration Example

Perform this task to configure EVPN MAC address limit.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC address limit</td>
<td>50</td>
</tr>
<tr>
<td>MAC limit action</td>
<td>limit, no-flood</td>
</tr>
<tr>
<td>MAC Notification</td>
<td>Syslog and SNMP trap messages</td>
</tr>
<tr>
<td>MAC limit threshold</td>
<td>80%</td>
</tr>
</tbody>
</table>

```
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)# mac
Router(config-l2vpn-bg-bd)# limit
Router(config-l2vpn-bg-bd-mac-limit)# maximum 50
Router(config-l2vpn-bg-bd-mac-limit)# action no-flood
Router(config-l2vpn-bg-bd-mac-limit)# notification both
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
        action no-flood
```

---

**EVPN Features**

**Configure EVPN MAC Address Limit**

- MAC address limit action is applied only when the number of local MAC addresses exceeds the configured limit. If the MAC address limit is exhausted by remote and local MAC addresses, or only by remote MAC addresses, the MAC address limit action is not applied. The surpassed traffic is treated as unknown unicast and is flooded across the bridge domain.

- 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.

### Configuration Example

Perform this task to configure EVPN MAC address limit.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC address limit</td>
<td>50</td>
</tr>
<tr>
<td>MAC limit action</td>
<td>limit, no-flood</td>
</tr>
<tr>
<td>MAC Notification</td>
<td>Syslog and SNMP trap messages</td>
</tr>
<tr>
<td>MAC limit threshold</td>
<td>80%</td>
</tr>
</tbody>
</table>

```
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)# mac
Router(config-l2vpn-bg-bd)# limit
Router(config-l2vpn-bg-bd-mac-limit)# maximum 50
Router(config-l2vpn-bg-bd-mac-limit)# action no-flood
Router(config-l2vpn-bg-bd-mac-limit)# notification both
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
        action no-flood
```
Configure EVVPN MAC Address Limit

notification both
!
!
mac limit threshold 80
commit

Verification

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

Router# show l2vpn bridge-domain bd-name EVVPN_2701 detail
Legend: pp = Partially Programmed.
Bridge group: EVVPN-BG-SH, bridge-domain: EVVPN_2701, id: 25, state: up, ShgId: 0, MSTi: 0
Coupled state: disabled
VINE state: EVVPN 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: limit, no-flood, Notification: syslog, trap
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), VFI: 0, PWS: 0 (0 up), PBB: 0 (0 up), VNIs: 0 (0 up)
List of EVPNs:
  EVVPN, 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: limit, no-flood, Notification: syslog, trap
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 VFI:
List of Access VFI:

**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.

*Figure 15: 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.

```bash
/* 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(config)# configure
RP/0/RSP0/CPU0:router(config)# router bgp 200
RP/0/RSP0/CPU0:router(config-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-evpn)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router(config-bgp-evpn)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp-evpn)# description MPLSFACINGPEER
RP/0/RSP0/CPU0:router(config-bgp-evpn)# update-source Loopback 0
RP/0/RSP0/CPU0:router(config-bgp-evpn)# 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 16: 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 BundleEther1.2001
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(config)# configure
RP/0/RSP0/CPU0:router(config)# router bgp 200
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)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn

Running Configuration

```
l2vpn
bridge group EVPN_ALL_ACTIVE
bridge-domain EVPN_2001
interface BundleEther1.2001
evi 2001
```
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 17: 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-12vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-12vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-12vpn-bg-bd)# interface BundleEther1.2001
RP/0/RSP0/CPU0:router(config-12vpn-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(config)# 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 Link Aggregation Control Protocol (LACP) bundle. */
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1 300
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(config)# interface bundle-Ether1.2001 l2transport
RP/0/RSP0/CPU0:router(config-if)# encapsulation dot1q 10
RP/0/RSP0/CPU0:router(config-if)# rewrite ingress tag pop 1 symmetric

Running Configuration

l2vpn
bridge group EVPN_ALL_ACTIVE
bridge-domain EVPN_2001
interface Bundle-Ether1.2001
! evi 2001
!
evpn
evi 2001
!
advertise-mac
!
interface bundle-ether1
eternet-segment
identifier type 0 01.11.00.00.00.00.00.00.01
!
router bgp 200
bgp router-id 209.165.200.227
address-family l2vpn evpn
!
neighbor 10.10.10.10
remote-as 200
description MPLS-FACING-PEER
update-source Loopback0
address-family l2vpn evpn
!
interface Bundle-Ether1
lacp switchover suppress-flaps 300
load-interval 30
!
interface bundle-Ether1.2001 l2transport
encapsulation dot1aq 2001
rewrite ingress tag pop 1 symmetric
!
Verification
Verify EVPN in dual home devices in All-Active mode.

RP/0/RSP0/CP00:router# show evpn ethernet-segment interface bundle-Ether 1 carvin$

<table>
<thead>
<tr>
<th>Ethernet Segment Id</th>
<th>Interface</th>
<th>Nexthops</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100.211b.fce5.df00.0b00</td>
<td>BE11</td>
<td>10.10.10.10</td>
</tr>
<tr>
<td>209.165.201.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Topology:
Operational : MHN
Configured : All-active (AApF) (default)
Primary Services : Auto-selection
Secondary Services: Auto-selection
Service Carving Results:
Forwards : 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.

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
ev1: 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
  875013022336
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 all location 0/6/cPU0
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
```

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
```

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
```
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.

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 18: 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 Remote 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 Remote 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
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)# 12vpn
Router (config-12vpn)# flexible-xconnect-service vlan-unaware fxc_mh1
Router (config-12vpn-fxs-vu)# interface Bundle-Ether22001.1
Router (config-12vpn-fxs-vu)# interface Bundle-Ether22001.2
Router (config-12vpn-fxs-vu)# interface Bundle-Ether22001.3
Router (config-12vpn-fxs-vu)# neighbor evpn evi 21006 target 22016
Router (config-12vpn-fxs-vu)# commit

/* Configure VLAN Aware FXC Service */
Router (config)# 12vpn
Router (config-12vpn)# flexible-xconnect-service vlan-aware evi 24001
Router (config-12vpn-fxs-vu)# interface Bundle-Ether22001.12
Router (config-12vpn-fxs-vu)# interface Bundle-Ether22001.13
Router (config-12vpn-fxs-vu)# interface Bundle-Ether22001.14
Router (config-12vpn-fxs-vu)# 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)# interface Bundle-Ether2001.1400
Router(config-l2vpn-fxs)# interface Bundle-Ether23001.1400
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit

PE2
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 31401
Router(config-l2vpn-fxs)# interface Bundle-Ether2001.1401
Router(config-l2vpn-fxs)# interface Bundle-Ether23001.1401
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit

/* Configure EVPN-VPWS xconnect service and native EVPN with IRB */

Router(config)# l2vpn
Router(config-l2vpn)# interface Bundle-Ether2001.11 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(config)# l2vpn
Router(config-l2vpn)# interface Bundle-Ether2001.21 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(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)# commit
Router(config-l2vpn-xc-p2p)# exit

/* Configure Native EVPN */

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

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20001
Router(config-evpn-ac)# ethernet-segment identifier type 0 ff.ff.ff.ff.ff.ff.ff.ff.ee
Router(config-evpn-ac-es)# bgp route-target 2200.0001.0001
Router(config-evpn-ac-es)# exit
Router(config-evpn)# evi 24001
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# route-target import 64:24001
Router(config-evpn-evi-bgp)# route-target export 64:24001
Router(config-evpn-evi-bgp)# exit
Running Configuration

/* Configure attachment circuits */
interface Bundle-Ether22001.12 l2transport
capsulation dot1q 1 second-dot1q 12

interface Bundle-Ether22001.13 l2transport
capsulation dot1q 1 second-dot1q 13

interface Bundle-Ether22001.14 l2transport
capsulation dot1q 1 second-dot1q 14

interface Bundle-Ether22001.1 l2transport
capsulation dot1q 1 second-dot1q 1

interface Bundle-Ether22001.2 l2transport
capsulation dot1q 1 second-dot1q 2

interface Bundle-Ether22001.3 l2transport
capsulation dot1q 1 second-dot1q 3

interface Bundle-Ether22001.4 l2transport
capsulation 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
eighbor 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.ffe
    bgp route-target 2200.0001.0001

    evi 24001
    bgp
    route-target import 64:24001
    route-target export 64:24001

    evi 21006
    bgp
    route-target 64:10006

    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 FW-PW: 0 Monitor-Session-FW: 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
------------------------ ----------------------------- -----------------------------
evii:24001 UP AC: BE22001.11 UP
         AC: BE22001.12 UP
         AC: BE22001.13 UP
         AC: BE22001.14 UP

------------------------ ----------------------------- -----------------------------

Router# show l2vpn xconnect group xg22001 xc-name evpn-vpws-mclag-22001
Fri Sep 1 17:28:58.259 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
XConnect Segment 1 Segment 2
Group Name ST Description ST Description ST
------------------------ ----------------------------- -----------------------------
xg22001 evpn-vpws-mclag-22001 UP BE22001.101 UP EVPN 22101, 220101,64.1.1.6 UP
Associated Commands

- evpn
- evi
- ethernet-segment
- advertise-mac
- show evpn ethernet-segment
- show evpn evi
- show evpn summary
- show l2vpn xconnect summary
- show l2vpn flexible-xconnect-service
- show l2vpn xconnect group

EVPN 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:

```
<table>
<thead>
<tr>
<th>Route Type (1 octet)</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (1 octet)</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>---</td>
</tr>
<tr>
<td>Route Distinguisher (RD) (8 octets)</td>
<td>*</td>
</tr>
<tr>
<td>----------------------</td>
<td>---</td>
</tr>
<tr>
<td>Ethernet Segment Identifier (10 octets)</td>
<td>*</td>
</tr>
<tr>
<td>----------------------</td>
<td>---</td>
</tr>
<tr>
<td>Ethernet Tag ID (4 octets)</td>
<td>*</td>
</tr>
<tr>
<td>----------------------</td>
<td>---</td>
</tr>
<tr>
<td>MPLS Label (3 octets)</td>
<td></td>
</tr>
</tbody>
</table>
```

**NLRI Format: Route-type 1:**

```
[Type][Len][RD][ESI][ETag][MPLS Label]
```

**Net attributes:** [Type][RD][ESI][ETag]

**Path attributes:** [MPLS Label]

**Example**

```plaintext
route-policy evpn-policy
  if rd in (1.1.1:0) [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 (1.1.2:0) [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:

```
+-------------------------------+
<table>
<thead>
<tr>
<th>Route Type (1 octet)</th>
</tr>
</thead>
</table>
+-------------------------------+
|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)     |
+-------------------------------+
```

**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**

```bash
route-policy evpn-policy
  if rd in (1.1.2.0) [and/or evpn-route-type is 2] [and/or esi in
  (0000.0000.0000.0000.0000) [and/or etag is 0] [and/or macaddress in (0013.aabb.ccdd)]
  [and/or destination in (1.2.3.4/32)] then
    set ..
  endif
end-policy
```
**Route Type 3: Inclusive Multicast Ethernet Tag Route**

This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.

An Inclusive Multicast Ethernet Tag route type specific EVPN NLRI consists of the following fields:

```
+----------------------------------+
| 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**

```plaintext
route-policy evpn-policy
  if rd in (1.1.1.1:300) [and/or evpn-route-type is 3] [and/or etag is 0] [and/or evpn-originator in (1.1.1.1)] 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:
**EVPN Route Types**

**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 (1.1.1.1:0) [and/or evpn-route-type is 4] [and/or esi in
  {00a1.a2a3.a4a5.a6a7.a8a9}] [and/or evpn-originator in (1.1.1.1)] then
    set ...
  endif
end-policy
```

**Route Type 5: IP Prefix Route**

An IP Prefix Route type specific EVPN NLRI consists of the following fields:
NLRI Format: Route-type 5:

[Type][Len][RD][ESI][ETag][IP Addr Len][IP Addr][GW IP Addr][Label]

Net attributes: [Type][RD][ETag][IP Addr Len][IP Addr]
Path attributes: [ESI], [GW IP Addr], [Label]

Example

route-policy evpn-policy
  if rd in (30.30.30.30:1) [and/or evpn-route-type is 5] [and/or esi in
  (0000.0000.0000.0000.0000.0000.0000.0000)] [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**

```bash
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**

```bash
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**

```bash
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.

```bash
/* Configuring a mac-set and refering it in a route-policy (Attach point - neighbor-in) */
Router# configure
Router(config)# mac-set demo_mac_set
Router(config-mac)# 1234.ffff.aaa3,
Router(config-mac)# 2323.4444.ffff
Router(config-mac)# end-set
Router(config)# !
Router(config)# route-policy policy_use_pass_mac_set
Router(config-rpl)# if mac in demo_mac_set then
Router(config-rpl-if)# set med 200
Router(config-rpl-if)# else
Router(config-rpl-else)# set med 1000
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit
```
Router(config)# router bgp 100
Router(config-bgp)# address-family 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 policy_use_pass_mac_set in
Router(config-bgp-nbr#af)# commit

/* Configuring a esi-set and refering it in a route-policy (Attach point = neighbor-in) */
Router(config)# 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(config)# configure
Router(config)# route-policy gateway_demo
Router(config-rpl)# if evpn-gateway in (10.0.0.0/32) then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
Router(config)# commit
Router(config)# route-policy originator_demo
Router(config-rpl)# if evpn-originator in (10.0.0.1/32) then
Router(config-rpl-if)# set local-preference 100
Router(config-rpl-if)# else
Router(config-rpl-else)# set med 200
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp#af)# !
Router(config-bgp#af)# neighbor 10.0.0.10
Router(config-bgp-nbr)# remote-as 8
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr#af)# route-policy gateway_demo in
Router(config-bgp-nbr#af)# route-policy originator_demo out
Router(config-bgp-nbr#af)# commit

Running Configuration

/* Configuring a mac-set and refering it in a route-policy (Attach point = neighbor-in) */
mac-set demo_mac_set
   1234.ffff.aaa3,
   2323.4444.ffff
end-set
!
route-policy policy_use_pass_mac_set
if mac in demo_mac_set then
    set med 200
else
    set med 1000
endif
end-policy
!
router bgp 100
    address-family ipv4 unicast
    !
    neighbor 10.0.0.10
        remote-as 8
        address-family ipv4 unicast
        route-policy policy_use_pass_mac_set in
    !
end

/* Configuring a esi-set and refering it in a route-policy (Attach point - neighbor-in) */
Wed Oct 26 11:52:23.720 IST
esi-set demo_esi
    ad34.1233.1222.ffff.44ff,
    ad34.1233.1222.ffff.6666
end-set
!
route-policy use_esi
    if esi in demo_esi then
        set local-preference 100
    else
        set local-preference 300
    endif
end-policy

EVPN Route Policy Examples

route-policy ex_2
    if rd in (2.2.18.2:1004) and evpn-route-type is 1 then
        drop
    elseif rd in (2.2.18.2:1009) and evpn-route-type is 1 then
        drop
    else
        pass
    endif
end-policy
!
route-policy ex_3
    if evpn-route-type is 5 then
        set extcommunity bandwidth (100:9999)
    else
        pass
    endif
end-policy
!
route-policy samp
end-policy
!
route-policy samp1
    if rd in (30.0.101.2:0) then
        pass
    endif
end-policy

EVPN Features
! route-policy samp2
  if rd in (30.0.101.2:0, 1:1) then
    pass
  endif
end-policy
!
route-policy samp3
  if rd in (*:* *) then
    pass
  endif
end-policy
!
route-policy samp4
  if rd in (30.0.101.2:* ) then
    pass
  endif
end-policy
!
route-policy samp5
  if evpn-route-type is 1 then
    pass
  endif
end-policy
!
route-policy samp6
  if evpn-route-type is 2 or evpn-route-type is 5 then
    pass
  endif
end-policy
!
route-policy samp7
  if evpn-route-type is 4 or evpn-route-type is 3 then
    pass
  endif
end-policy
!
route-policy samp8
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 then
    pass
  endif
end-policy
!
route-policy samp9
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 or evpn-route-type
is 4 then
    pass
  endif
end-policy
!
route-policy test1
  if evpn-route-type is 2 then
    set next-hop 10.2.3.4
  else
    pass
  endif
end-policy
!
route-policy test2
  if evpn-route-type is 2 then
    set next-hop 10.10.10.10
  else
    drop
  endif

end-policy
!
route-policy test3
  if evpn-route-type is 1 then
    set tag 9988
  else
    pass
  endif
end-policy
!
route-policy samp21
  if mac in (6000.6000.6000) then
    pass
  endif
end-policy
!
route-policy samp22
  if extcommunity rt matches-any (100:1001) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp23
  if evpn-route-type is 1 and esi in (aaaa.bbbb.cccc.dddd.eeee) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp24
  if evpn-route-type is 5 and extcommunity rt matches-any (100:1001) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp25
  if evpn-route-type is 2 and esi in (1234.1234.1234.1234.1236) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp26
  if etag in (20000) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp27
  if destination in (99.99.99.1) and etag in (20000) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp31
   if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 or evpn-route-type is 4 or evpn-route-type is 5 then
      pass
   else
      drop
   endif
end-policy
!
route-policy samp33
   if esi in evpn_esi_set1 then
      pass
   else
      drop
   endif
end-policy
!
route-policy samp34
   if destination in (90:1:1::9/128) then
      pass
   else
      drop
   endif
end-policy
!
route-policy samp35
   if destination in evpn_prefix_set1 then
      pass
   else
      drop
   endif
end-policy
!
route-policy samp36
   if evpn-route-type is 3 and evpn-originator in (80:1:1::3) then
      pass
   else
      drop
   endif
end-policy
!
route-policy samp37
   if evpn-gateway in (10:10::10) then
      pass
   else
      drop
   endif
end-policy
!
route-policy samp38
   if mac in evpn_mac_set1 then
      pass
   else
      drop
   endif
end-policy
!
route-policy samp39
   if mac in {6000.6000.6002} then
      pass
   else
      drop
   endif
end-policy
route-policy samp41
  if evpn-gateway in (10.10.10.10, 10:10::10) then
    pass
  else
    drop
  endif
end-policy

route-policy samp42
  if evpn-originator in (24.162.160.1/32, 70:1:1::1/128) then
    pass
  else
    drop
  endif
end-policy

route-policy example
  if rd in (62300:1903) and evpn-route-type is 1 then
drop
  elseif rd in (62300:19032) and evpn-route-type is 1 then
drop
  else
    pass
  endif
end-policy

route-policy samp100
  if evpn-route-type is 4 or evpn-route-type is 5 then
    drop
  else
    pass
  endif
end-policy

route-policy samp101
  if evpn-route-type is 4 then
    drop
  else
    pass
  endif
end-policy

route-policy samp102
  if evpn-route-type is 4 then
    drop
  elseif evpn-route-type is 5 then
    drop
  else
    pass
  endif
end-policy

route-policy samp103
  if evpn-route-type is 2 and destination in evpn_prefix_set1 then
    drop
  else
    pass
  endif
end-policy

route-policy samp104
  if evpn-route-type is 1 and etag in evpn_etag_set1 then
    drop
elseif evpn-route-type is 2 and mac in evpn_mac_set1 then
drop
elseif evpn-route-type is 5 and esi in evpn_esi_set1 then
drop
else
pass
endif
end-policy
!

EVPN Features

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
CHAPTER 10

Configure EVPN IRB

This chapter introduces you to Ethernet VPN (EVPN) Integrated Routing and Bridging (IRB) feature and describe how you can configure the EVPN IRB feature.

- EVPN IRB, on page 149
- EVPN Single-Homing Access Gateway, on page 150
- EVPN Multihoming All-Active, on page 151
- Enable Auto-BGP RT with Manual ESI Configuration, on page 152
- Supported EVPN IRB Scenarios, on page 152
- Distributed Anycast Gateway, on page 152
- VM Mobility Support, on page 155

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 19: EVPN IRB

The benefit of EVPN IRB is that it allows the hosts in an IP subnet to be provisioned anywhere in the data center. When a virtual machine (VM) in a subnet is provisioned behind a EVPN PE, and another VM is required in the same subnet, it can be provisioned behind another EVPN PE. The VMs do not have to be localized; they need not be directly connected; or be in the same complex. The VM is allowed to move across
in the same subnet. Availability of IP MPLS network across all the EVPN PEs enables the provisioning of VM mobility. The EVPN PEs route traffic to each other through MPLS encapsulation.

The EVPN PEs are connected to each other by a spine so they have IP reachability to each other's loopback interfaces. The IP network and MPLS tunnels existing between these EVPN PEs constitute the IP MPLS underlay fabric.

You can configure the MPLS tunnels to tunnel Layer 2 traffic, and to overlay VPN on these tunnels. EVPN control plane distributes both Layer 2 MAC reachability and Layer 3 IP reachability for hosts within the context of the VPN; it overlays a tenant's VPN network on top of the MPLS underlay fabric. Thus you can have tenant's hosts, which are in the same subnet layer 2 domain, but distributed across the fabric, communicate to each other as if they are in a Layer 2 network.

The Layer 2 VLAN and the corresponding IP subnet are not only a network of physically connected hosts on Layer 2 links, but an overlayed network on top of underlayed IP MPLS fabric which is spread across the datacenter.

A routing service, which enables stretching of the subnet across the fabric, is available. It also provides Layer 3 VPN and performs routing between subnets within the context of the Layer 3 VPN. The EVPN PEs provide Layer 2 bridging service between hosts that are spread across the fabric within a Layer 2 domain that is stretched across the fabric, and Layer 3 VPN service or inter-subnet routing service for hosts in different subnets within Layer 3 VPN. For example, as shown in the above topology diagram, the two VM are in the same subnet but they are not connected directly through each other through a Layer 2 link. The Layer 2 link is replaced by MPLS tunnels that are connecting them. The whole fabric acts as a single switch and bridges traffic from one VM to the other. This also enables VM mobility.

---

**Note**

Egress marking is not supported on L2 interfaces in a bridge domain.

In the above topology diagram, the VMs, VM1 and VM2 are connected each other. When VM2 migrates to a different switch and different server, the VM's current MAC address and IP address are retained. When the subnet is stretched between two EVPN PEs, the same IRB configuration is applied on both the devices.

For stretching within the same subnet, you must configure the AC interface and the EVI; it is not required to configure IRB interface or VRF.

**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 CE.
The above topology depicts how EVPN single-homing access gateway enables network connectivity by allowing a CE device to connect to one PE device. The PE device is attached to the Ethernet Segment through bundle or physical interfaces. Null Ethernet Segment Identifier (ESI) is used for single-homing.

**EVPN Multihoming All-Active**

In EVPN IRB, both EVPN and IP VPN (both VPNv4 and VPNv6) address families are enabled between routers and Data Center Interconnect (DCI) gateways. When Layer 2 (L2) stretch is not available in multiple data centers (DC), routing is established through VPNv4 or VPNv6 routes. When Layer 2 stretch is available, host routing is applied where IP-MAC routes are learnt by ARP and are distributed to EVPN/BGP. In remote peer gateway, these IP-MAC EVPN routes are imported into IP VPN routing table from EVPN route-type 2 routes with secondary label and Layer 3 VRF route-target.

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.
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 one EFP is supported per ESI per EVI
  - 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 only supported in VRF and is not supported in global VRF. In other words, EVPN IRB with EV-LAG multihoming is supported in global VRF without subnet being stretched beyond the multi-homing leafs

Distributed Anycast Gateway

EVPN IRB for the given subnet is configured on all the EVPN PEs that are hosted on this subnet. To facilitate optimal routing while supporting transparent virtual machine mobility, hosts are configured with a single default gateway address for their local subnet. That single (anycast) gateway address is configured with a single (anycast) MAC address on all EVPN PE nodes locally supporting that subnet. This process is repeated for each locally defined subnet requires Anycast Gateway support.

The host-to-host Layer 3 traffic, similar to Layer 3 VPN PE-PE forwarding, is routed on the source EVPN PE to the destination EVPN PE next-hop over an IP or MPLS tunnel, where it is routed again to the directly connected host. Such forwarding is also known as Symmetric IRB because the Layer 3 flows are routed at both the source and destination EVPN PEs.

The following are the solutions that are part of the Distributed Anycast Gateway feature:
EVPN IRB with All-Active Multi-Homing without Subnet Stretch or Host-Routing across the Fabric

For those subnets that are local to a set of multi-homing EVPN PEs, EVPN IRB Distributed Anycast Gateway is established through subnet routes that are advertised using EVPN Route Type 5 to VRF-hosting remote leafs. Though there is no need for the /32 routes within the subnet to be advertised, host MAC and ARP entries have to synced across the EVPN PE to which the servers are multi-homed.

This type of multi-homing has the following characteristics:

- All-active EV-LAG on access
- Layer 3 ECMP for the fabric for dual-homed hosts based on subnet routes
- Absence of Layer 2 subnet stretch over the fabric
- Layer 2 stretch within redundancy group of leafs with orphan ports

Prefix-routing solution for a non-stretched subnet is summarized as below:

Across multi-homing EVPN PEs:

- Local ARP cache and MAC addresses are synchronized for dual-homed hosts through EVPN MAC+IP host route advertisements. They are imported as local, and are based on the local ESI match, for optimal forwarding to the access gateway.
- Orphan MAC addresses and host IP addresses are installed as remote addresses over the fabric.
- ES/EAD routes are exchanges for the designated forwarder (DF) election and split-horizon label.

Across remote EVPN PEs:

- Dual-homed MAC+IP EVPN Route Type 2 is exchanged with the ESI, EVI Label, Layer 2-Route Type. It is not imported across the fabric, if there is no subnet stretch or host-routing.
- The subnet IP EVPN Route Type 5 is exchanged with VRF label and Layer 3-Route Type.
- Layer 3 Route Type for the VRFs is imported that are present locally.
- Layer 2 Route Type for locally present BDs is imported. It is only imported from the leaf in the same redundancy group, if BD is not stretched.

EVPN IRB with All-Active Multihoming with Subnet Stretch or Host-Routing across the Fabric

For a bridge domain or subnet that is stretched across remote EVPN PEs, both /32 host routes and MAC routes are distributed in a EVPN overlay control plane to enable Layer 2 and Layer 3 traffic to the end points in a stretched subnet.

This type of multihoming has the following characteristics:

- All-active EV-LAG on the access gateway
- Layer 2 or Layer 3 ECMP for the fabric for dual-homed hosts based on Route Type 1 and Route Type 2
• Layer 3 unipath over the fabric for single-homed hosts based on Route Type 2
• Layer 2 subnet stretch over the fabric
• Layer 2 stretch within redundancy group of leafs with orphan ports

MAC and host routing solution for a stretched subnet is summarized as follows:

Across multihoming EVPN PEs:
• The Local ARP cache and MAC addresses are synchronized for dual-homed hosts through EVPN
  MAC+IP host route advertisements. They are imported as local, based on the local ESI match, for optimal
  forwarding to the access gateway.
• Synchronized MAC+IP are re-originated for inter-subnet Layer 3 ECMP.
• Orphan MAC address and host IP address are installed as remote addresses over the fabric.
• ES/EAD route is exchanged for designated forwarder (DF) election and split-horizon label.

Across remote EVPN PEs:
• Dual-homed MAC+IP EVPN Route Type 2 is exchange with ESI, EVI label, Layer 2-Route Type, VRF
  label, and Layer 3-Route Type.
• Subnet IP EVPN Route Type 5 is exchanged for VRF label, Layer 3-Route Type for silent hosts, and
  non-stretched subnets.
• Layer 3 Route Type is imported for locally present VRFs.
• Layer 2 Route Type is imported for locally present bridge domains.

**MAC and IP Unicast Control Plane**

This use case has following types:

**Prefix Routing or No Subnet Stretch**

IP reachability across the fabric is established using subnet prefix routes that are advertised using EVPN Route
Type 5 with the VPN label and VRF RTs. Host ARP and MAC sync are established across multi-homing
EVPN PEs using MAC+IP Route Type 2 based on a shared ESI to enable local switching through both the
multi-homing EVPN PEs.

**Host Routing or Stretched Subnet**

When a host is discovered through ARP, the MAC and IP Route Type 2 is advertised with both MAC VRF
and IP VRF router targets, and with VPN labels for both MAC-VRF and IP-VRF. Particularly, the VRF route
targets and Layer 3 VPN label are associated with Route Type 2 to achieve PE-PE IP routing identical to
traditional L3VPNs. A remote EVPN PE installs IP/32 entries directly in Layer 3 VRF table through the
advertising EVPN PE next-hop with the Layer 3 VPN label encapsulation, much like a Layer 3 VPN imposition
PE. This approach avoids the need to install separate adjacency rewrites for each remote host in a stretched
subnet. Instead, it inherits a key Layer 3 VPN scale benefit of being able to share a common forwarding rewrite
or load-balance resource across all IP host entries reachable through a set of EVPN PEs.

**ARP and MAC sync**

For hosts that are connected through LAG to more that one EVPN PE, the local host ARP and MAC entries
are learnt in data plane on either or both of the multihoming EVPN PEs. Local ARP and MAC entries are
synced across the two multihoming EVPN PEs using MAC and IP Route Type 2 based on a shared ESI to enable local switching through both the multihoming EVPN PEs. Essentially, a MAC and IP Route Type 2 that is received with a local ESI causes the installation of a synced MAC entry that points to the local AC port, and a synced ARP entry that is installed on the local BVI interface.

Only one Ethernet Flow Point (EFP) is supported per non-Zero ESI per bridge domain or EVI. This is a limitation of EVPN.

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.

If custom or static MAC address is configured on a BVI interface, the MAC address on the wire may be different than what is configured. This has no operational or functional impact.

Intra-subnet Unicast Data Plane

The Layer 2 traffic is bridged on the source EVPN PE using ECMP paths to remote EVPN PEs, established through MAC+IP RT2, for every ES and for every EVI, ES and EAD Route Type 2 routes that are advertised from the local EVPN PEs.

Inter-subnet Unicast Data Plane

Inter-subnet traffic is routed on the source ToRs through overlay ECMP to the destination ToR next-hops. Data packet are encapsulated with the VPN label advertised from the ToR and tunnel label for the BGP next-hop towards the spine. It is then routed again on the destination ToR using a local ARP adjacency towards the host. IP ECMP on the remote ToRs is established through local and re-originated routes advertised from the local ToRs.

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/CP00:router# configure
RP/0/RSP0/CP00:router(config)# interface Bundle-Ether 3
RP/0/RSP0/CP00:router(config-if)# lACP system mac 1.1.1
RP/0/RSP0/CP00:router(config-if)# exit
RP/0/RSP0/CP00:router(config)# cef adjacency route override rib
/* Configure EVPN L3 VRF 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)# 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 2001:DB8::1

/* 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-bd)# bridge-domain 1-1
Router(config-l2vpn-bd-ac)# interface GigabitEthernet 0/0/0/1.1
Router(config-l2vpn-bd-ac)# evi 1
Router(config-l2vpn-bd-ac-evi)# commit
Router(config-l2vpn-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
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# redistribute connected
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# redistribute static

/* 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. */
configure

l2vpn

bridge group irb

bridge-domain irb1

interface bundle-Ether3.1001

routed interface BVI100

split-horizon group core

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; only multi-homing active-active mode is supported for EVPN IRB.

RP/0/RSP0/CPU0# 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.
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/RSP0/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: 0x77d7bc380
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) 10000001000100100001000100010800 MTU: 1500
Adjacency pointer is: 0x770ab778
Platform adjacency pointer is: 0x77d7bc3b10
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 sequence ID for VM mobility.

```
RP/0/RSP0/CPU0:router# show l2vpn mac-learning mac-ipv4 all location 0/0/cPU0
```

```
Topo ID Producer Next Hop(s) Mac Address IP Address
--- --- --- ---
 6 0/0/CPU0 BVI 1000.0001.0001 10.1.1.11
 7 0/0/CPU0 BV1 0000.f65a.3570 10.1.2.91
 7 0/0/CPU0 BV2 0000.f65a.357d 10.1.2.93
```

```
RP/0/RSP0/CPU0:router# show l2vpn mac-learning mac-ipv4 all location 0/0/cPU0
```

```
Topo ID Producer Next Hop(s) Mac Address IP Address
--- --- --- ---
 6 0/0/CPU0 BVI 0000.f65a.357c fe80::200:f6ff:fe5a:357c
 7 0/0/CPU0 BV1 0000.f65a.3570 10:1:2:1:91
 7 0/0/CPU0 BV2 0000.f65a.357d 10:1:2:1:93
 7 0/0/CPU0 BV2 0000.f65a.3570 fe80::200:f6ff:fe5a:3570
```

Verify sequence ID for VM mobility.
Verify EVPN IRB

RP/0/RSP0/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;
(S)=Peer Sync; (F)=Flush;
(D)=Duplicate MAC; (Z)=Frozen MAC;

<table>
<thead>
<tr>
<th>Topo ID</th>
<th>Mac Address</th>
<th>IP Address</th>
<th>Prod</th>
<th>Next Hop(s)</th>
<th>Seq No</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>0022.6730.0001</td>
<td>10.130.0.2</td>
<td>L2VPN</td>
<td>Bundle-Ether6.1300</td>
<td>0</td>
<td>SB 0 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LOCAL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Last Update: Sun Apr 30 15:00:01.911 PDT

RP/0/RSP0/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;

<table>
<thead>
<tr>
<th>Topo ID</th>
<th>Mac Address</th>
<th>Prod</th>
<th>Next Hop(s)</th>
<th>Seq No</th>
<th>Flags</th>
<th>Slot</th>
<th>ESI</th>
<th>Opaque Data Type</th>
<th>Opaque Data Len</th>
<th>Opaque Data Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>0022.6730.0002</td>
<td>10.130.0.3</td>
<td>LOCAL</td>
<td>Bundle-Ether6.1300</td>
<td>0</td>
<td>B</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Last Update: Thu Apr 20 09:04:44.358 PDT

Verify duplicate detection and recovery parameters.

/* Use the show run evpn mac to verify the current parameters: */

RP/0/RSP0/CPU0:router# show run evpn mac

evpn
  mac
    secure
      freeze-time 5
      move-count 1000
      move-interval 60
      retry-count 1000
    !
  !

/* Perform the following steps to change the existing parameters. */

RP/0/RP0/CPU0:EVPN-LF1# configure
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.

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/RSP0/CPU0:router# show l2route evpn mac-ip evi 6

<table>
<thead>
<tr>
<th>Topo ID</th>
<th>Mac Address</th>
<th>IP Address</th>
<th>Prod</th>
<th>Next Hop(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0000.f65a.3569</td>
<td>10.1.1.1101</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.3575</td>
<td>10.1.1.97</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.3575</td>
<td>10.1.1.97</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.3575 fe80::200:f6ff:fe5a:3575</td>
<td>172.16.0.2/24</td>
<td>014/ME</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>L2VPN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>L2VPN</td>
<td></td>
</tr>
</tbody>
</table>
Verify entries to obtain details of EVPN.

RP/0/RSP0/CPU0:router# show evpn evi vpn-id 1 mac ipv4 10.1.1.93 detail

<table>
<thead>
<tr>
<th>EVI</th>
<th>MAC address</th>
<th>IP address</th>
<th>Nexthop</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0000.f65a.357c</td>
<td>10.1.1.93</td>
<td>172.16.0.2</td>
<td>24014</td>
</tr>
</tbody>
</table>

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/RSP0/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
  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

>>> Second label when IRB host-routing is enabled.
EVVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1

RP/0/RSP0/CPU0# show bgp l2vpn evpn rd 172.16.0.1:1
[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: 172.16.0.1:1
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 EVVPN, 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 EVVPN gateway. EVPN type-2 routes are imported into EVVPN. The host routes of IPv4 /32 addresses are imported only into IP VRF route-table in the remote EVVPN gateway, but not in the local EVVPN gateway where local BVI adjacency is used to overwrite RIB entries.

RP/0/RSP0/CPU0# 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: 0100.6cbe.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: 0100.6cbe.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1

```
RP/0/RSP0/CP0:router# show bgp l2vpn evpn rd 172.16.0.7:1
[2][0][48][0000.656a.357c][128][10:1:1::93]/232
BGP routing table entry for [2][0][48][0000.656a.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: 0100.6cbe.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: 0100.6cbe.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/RSP0/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/RSP0/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
```

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
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.

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/RSP0/CPU0:router# show bgp vpv6 unicast vrf evpn1 10:1::93/128

BGP routing table entry for 10:1::93/128, Route Distinguisher: 172.16.0.7:11
Versions:
Process brRIB/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/RSP0/CPU0:router# show bgp vpv4 unicast vrf evpn1 10.1.1.93/32

-- For local routing and forwarding
RP/0/RSP0/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.
Next hop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
Route metric is 0
No advertising protos.

RP/0/RSP0/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 [0x7f5f31f88 0x0]
next hop local adjacency >>> Forwarding with local synced ARP adjacency entries.

For remote routing and forwarding:

RP/0/RSP0/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
Next hop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
Route metric is 0
No advertising protos.

RP/0/RSP0/CPU0:router# show cef vrf evpn1 10.1.1.93 location 0/0/CPU0
10.1.1.93/32, version 86, internal 0x5000001 0x0 (ptr 0x99fac884) [1], 0x0 (0x0), 0x208 (0x96c58494)
Updated Feb 28 16:06:39.285
Prefix Len 32, traffic index 0, precedence n/a, priority 3 via 172.16.0.1/32, 15 dependencies, recursive [flags 0x6000]
path-idx 0 NHID 0x0 [0x97955380 0x0]
recursion-via-/32
next hop VRF = 'default', table = 0xe0000000
next hop 172.16.0.1/32 via 34034/0/21
next hop 100.0.57.5/32 Te0/0/0/3 labels imposed {ImplNull 24011 24027}
next hop 100.0.67.6/32 Te0/0/0/1 labels imposed {ImplNull 24009 24027}
via 172.16.0.2/32, 11 dependencies, recursive, backup [flags 0x6100]
path-idx 1 NHID 0x0 [0x979554a0 0x0]
recursion-via-/32
next hop VRF = 'default', table = 0xe0000000
next hop 172.16.0.2/32 via 34035/0/21
next hop 100.0.57.5/32 Te0/0/0/3 labels imposed {ImplNull 24012 24031}
next hop 100.0.67.6/32 Te0/0/0/1 labels imposed {ImplNull 24010 24031}

The following sections describe how to verify the subnet stretching.
Verify the VRF.
RP/0/RP0/CPU0:leafW# show run vrf cust130

vrf cust130
address-family ipv4 unicast
    import route-target 130:130
    export route-target 130:130

Verify the BGP configuration.

RP/0/RP0/CPU0:leafW# show run router bgp | begin vrf cust130

vrf cust130
    rd auto
    address-family ipv4 unicast
        label mode per-vrf
        maximum-paths ibgp 10
        redistribute connected

Verify the L2VPN.

RP/0/RP0/CPU0:leafW# show run l2vpn bridge group bg130

l2vpn
    bridge group bg130
        bridge-domain bd130
            interface Bundle-Ether1.1300
            interface Bundle-Ether5.1300
            routed interface BVI130
evi 130

---

EVPN IPv6 Hosts with Mobility

EVPN IPv6 Hosts with Mobility feature enables you to provide EVPN IPv6 service over IPv4-MPLS core network. This feature supports all-active multihoming and virtual machine (VM) or host move.

Service Providers (SPs) use a stable and established core with IPv4-MPLS backbone for providing IPv4 VPN services. The IPv6 VPN Provider Edge Transport over MPLS (IPv6 on Provider Edge Routers [6PE] and IPv6 on VPN Provider Edge Routers [6VPE]) facilitates SPs to offer IPv6 VPN services over IPv4 backbone without an IPv6 core. The provide edge (PE) routers run MP-iBGP to advertise IPv6 reachability and IPv6 label distribution. For 6PE, the labels are allocated per IPv6 prefix learnt from connected customer edge (CE) routers and for 6VPE, the PE router can be configured to allocate labels on a per-prefix or per-CE and per-VRF level.
Mobility Support

In global VRF, mobility is not supported. However, you can move a host from one ES to another ES within the same bridge domain. The host gets a new MAC address and IP address. The host can have multiple IP addresses for the same MAC address.

In non-default VRF, mobility is supported with the following conditions:

- Basic MAC move: The IP address and MAC address remains the same. You can move a host from one ES to another ES with the same IP address and MAC address
- Same MAC address but with a different IP address: The host gets a new IP address
- Same IP address but with a different MAC address: The host gets a new MAC address but retains the same IP address
- Multiple IP addresses with the same MAC address: Many VMs are involved in the same the MAC move

Restrictions

- In customer VRFs, when host routing is not configured, MAC-IP advertisement is different between zero ESI and none-zero ESI. When host routing is not configured, MAC-IP with non-zero ESI is advertised without L3 RT (VRF RT). MAC-IP with zero ESI is not advertised. The following table lists the behavior of MAC-IP advertisement with respect to ESI and host routing.

<table>
<thead>
<tr>
<th>ESI Type</th>
<th>With host routing</th>
<th>Without host routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC-IP with non-zero ESI</td>
<td>Advertised with L3 VRF RT</td>
<td>Advertised without L3 VRF RT</td>
</tr>
<tr>
<td>MAC-IP with zero ESI</td>
<td>Advertised with L3 VRF RT</td>
<td>Not advertised</td>
</tr>
</tbody>
</table>

- 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
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.

---

Note

```c
/* 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 */
```

```c
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)# afi-ipv4 mx-prefix max 1000 primary
Route(config-isis-af)# metric 10
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
```
Configure EVPN IRB

Configure EVPN IPv6 Hosts with Mobility

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

/* 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
Router(config-bgp-af)# maximum-paths ibgp 10 unequal-cost
Router(config-bgp-af)# redistribute connected
Router(config-bgp-af)# exit

Router(config-bgp)# address-family ipv4 unicast
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
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
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
Router(config-bgp-af)# bgp implicit-import
Router(config-bgp-af)# exit

Router(config-bgp)# neighbor-group evpn-rr
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# update-source loopback0
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy nh-lo10 out
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp-nbr)# address-family ipv6 labeled-unicast ---> For 6PE
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy nh-lo10 out
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated ---> For Route Type 5
Router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated ---> For Route Type 5
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp)# neighbor 160.0.0.1
Router(config-bgp-nbr)# use neighbor-group evpn-rr
Router(config-bgp-nbr)# exit
!
Router(config-bgp)# neighbor 160.0.0.2
Router(config-bgp-nbr)# use neighbor-group evpn-rr
Router(config-bgp-nbr)# exit
!
Router(config-bgp)# vrf all
Router(config-bgp-vrf)# rd 1605:102
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# maximum-paths ibgp 10 unequal-cost
Router(config-bgp-vrf-af)# redistribute connected ---> Triggers Route Type 5
Router(config-bgp-vrf-af)# exit
!
Router(config-bgp-vrf)# address-family ipv6 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# maximum-paths ibgp 10 unequal-cost
Router(config-bgp-vrf-af)# redistribute connected
Router(config-bgp-vrf-af)# exit
!
/* Configure AC interface */

Router(config)# interface Bundle-Ether1.102 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 102
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

/* Configure BVI interface */

Router(config)# interface BVI100
Router(config-if)# ipv4 address 56.78.100.1 255.255.255.0
Router(config-if)# ipv6 address 56:78:100::1/64
Router(config-if)# mac-address 22.22.22
Router(config-if)# exit

Router(config)# interface BVI102
Router(config-if)# host-routing
Router(config-if)# vrf cust102
Router(config-if-vrf)# ipv4 address 56.78.102.1 255.255.255.0
Router(config-if-vrf)# ipv6 address 56:78:100::1/64
Router(config-if-vrf)# ipv6 address 56:78:102::1/64
Router(config-if-vrf)# mac-address 22.22.22
Router(config-if)# commit
/** Configure CEF */ [Required for dual homing]

Router# configure
Router(config)# cef adjacency route override rib

/** Configure EVPN, and configure main bundle ethernet segment parameters in EVPN */

Router# configure
Router(config)# evpn
Router(config-evpn)# evi 102
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# rd 1605:102
Router(config-evpn-evi-bgp)# route-target import 160102:102
Router(config-evpn-evi-bgp)# route-target export 160102:102
Router(config-evpn-evi-bgp)# exit
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# exit

Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 56.56.56.56.56.56.56.56.01
Router(config-evpn-ac-es)# exit

Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 56.56.56.56.56.56.56.56.02
Router(config-evpn-ac-es)# commit

/** Configure L2VPN */

Router# configure
Router(config)# 12vpn
Router(config-12vpn)# bridge group bg102
Router(config-12vpn-bg)# bridge-domain bd102
Router(config-12vpn-bg-bd)# interface Bundle-Ether1.102
Router(config-12vpn-bg-bd-ac)# exit

Router(config-12vpn-bg-bd)# interface Bundle-Ether2.102
Router(config-12vpn-bg-bd-ac)# exit

Router(config-12vpn-bg-bd)# interface Bundle-Ether3.102
Router(config-12vpn-bg-bd-ac)# exit

Router(config-12vpn-bg-bd)# interface Bundle-Ether4.102
Router(config-12vpn-bg-bd-ac)# exit

Router(config-12vpn-bg-bd)# interface Bundle-Ether5.102
Router(config-12vpn-bg-bd-ac)# routed interface BVI102
Router(config-12vpn-bg-bd-bvi)# evi 102
Router(config-12vpn-bg-bd-bvi-es)# commit

/** Configure VRF */

vrf cust102
address-family ipv4 unicast
import route-target
160102:16102

Running Configuration
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
!
neighbor-group evpn-rr
remote-as 100
bfd fast-detect
update-source Loopback0
address-family ipv4 unicast
   route-policy pass-all in
   route-policy nh-lo10 out
!
address-family ipv6 labeled-unicast ---> For 6PE
   route-policy pass-all out
!
address-family l2vpn evpn
   route-policy pass-all in
   route-policy nh-lo10 out
   advertise vpnv4 unicast re-originated ---> For Route Type 5
   advertise vpnv6 unicast re-originated ---> For Route Type 5
!
neighbor 160.0.0.1
use neighbor-group evpn-rr
!
neighbor 160.0.0.2
use neighbor-group evpn-rr
!
vrf cust102
rd 1605:102
address-family ipv4 unicast
   label mode per-vrf
   maximum-paths ibgp 10 unequal-cost
   redistribute connected <<---- Triggers Route Type 5
!
address-family ipv6 unicast
   label mode per-vrf
maximum-paths ibgp 10 unequal-cost
redistribute connected
!

/* Configure AC interface */

interface Bundle-Ether1.102 l2transport
  encapsulation dot1q 102
  rewrite ingress tag pop 1 symmetric
!
/* Configure BVI interface */

interface BVI100
  ipv4 address 56.78.100.1 255.255.255.0
  ipv6 address 56:78:100::1/64
  mac-address 22.22.22.
!
interface BVI102
  host-routing
    vrf cust102
    ipv4 address 56.78.102.1 255.255.255.0
    ipv6 address 56:78:100::1/64
    ipv6 address 56:78:102::1/64
    mac-address 22.22.22.
!
/* Configure CEF */ [ Required for Dual homing]

cef adjacency route override rib
!
/* Configure EVPN */

evpn
evi 102
bgp
rd 1605:102
route-target import 160102:102
route-target export 160102:102
!
advertise-mac
!
!
interface Bundle-Ether1
ethernet-segment
  identifier type 0 56.56.56.56.56.56.56.56.01
!
!
interface Bundle-Ether2
ethernet-segment
  identifier type 0 56.56.56.56.56.56.56.56.02
!
!
/* Configure L2VPN */

l2vpn
bridge group bg102
bridge-domain bd102
interface Bundle-Ether1.102
!
interface Bundle-Ether2.102
!
interface Bundle-Ether3.102
! interface Bundle-Ether4.102
! interface Bundle-Ether5.102
! routed interface BVI102
! evi 102
!
!

**Verification**

Verify that you have configured EVPN IPv6 Hosts with Mobility feature is configured.

/* 6PE and Static Route Advertisement */
Host route is advertised as EVPN Route Type 2

Router# show bgp ipv6 unicast 56:78:100::2
BGP routing table entry for 56:78:100::2/128
Versions:
  Process bRIB/RIB SendTblVer
  Speaker 212 212
  Local Label: 2
Last Modified: Oct 31 19:13:10.998 for 00:00:19
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
  160.5.5.5 (metric 20) from 160.0.0.1 (160.0.0.5)
  Received Label 2
  Origin IGP, localpref 100, valid, internal, best, group-best, imported
  Received Path ID 0, Local Path ID 0, version 212
  Extended community: Flags 0x20: SoO:160.5.5.5:100 RT:160100:100
  mac: 00:06:01:00:01:02
  Originator: 160.0.0.5, Cluster list: 100.0.0.4
  Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 1605:100

/* Manually configured static route in global VRF */

Router# show bgp ipv6 unicast 56:78:100::2
BGP routing table entry for 30::1/128
Versions:
  Process bRIB/RIB SendTblVer
  Speaker 9 9
  Local Label: 2
Paths: (2 available, best #2)
  Advertised to update-groups (with more than one peer): 0.2
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
  160.0.0.6 (metric 20) from 160.0.0.1 (160.0.0.6)
  Received Label 2
  Origin incomplete, metric 0, localpref 100, valid, internal, labeled-unicast
  Received Path ID 0, Local Path ID 0, version 0
  mac: 10:11:04:64:f2:7f
Originator: 160.0.0.6, Cluster list: 100.0.0.4
Path #2: Received by speaker 0
Advertised to update-groups (with more than one peer):
0.2
Local
56:78:100:2 from :: (160.0.0.5)
Origin incomplete, metric 0, localpref 100, weight 32768, valid, redistributed, best, group-best
Received Path ID 0, Local Path ID 0, version 9
mac: 10:11:04:64:f2:7f

/* Verify Ethernet Segments are peering for Dual homing */

Router# show evpn ethernet-segment int bundle-Ether 1

Ethernet Segment Id Interface Nexthops
---------------------------------- -------------------
0056.5656.5656.5656.5601 BE1 160.5.5.5
----------------------------------
160.6.6.6

/* Verify DF election */

Router# show evpn ethernet-segment int bundle-Ether 1 carving detail

Legend:
A - Load-balancing mode and Access Protection incompatible,
B - No Forwarders EVPN-enabled,
C - Backbone Source MAC missing (PBB-EVPN),
RT - ES-Import Route Target missing,
E - ESI missing,
H - Interface handle missing,
I - Name (Interface or Virtual Access) missing,
M - Interface in Down state,
O - BGP End of Download missing,
P - Interface already Access Protected,
Pf - Interface forced single-homed,
R - BGP RID not received,
S - Interface in redundancy standby state,
X - ESI-extracted MAC Conflict
SHG - No local split-horizon-group label allocated

Ethernet Segment Id Interface Nexthops
---------------------------------- -------------------
0056.5656.5656.5656.5601 BE1 160.5.5.5
----------------------------------
160.6.6.6
ES to BGP Gates : Ready
ES to L2FIB Gates : Ready
Main port :
Interface name : Bundle-Ether1
Interface MAC : 008a.9644.acdd
IFHandle : 0x0800004dc
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
Configure EVPN IRB

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
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 183
- EVPN-VPWS Multi-Homed, on page 185

**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.
Configure EVPN-VPWS Single Homed

This section describes how you can configure single-homed EVPN-VPWS feature.

Running Configuration

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

```
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 22: 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
Configure EVPN-VPWS Multi-Homed

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)# 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)# 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(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)# 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(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)# 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

EVPN Virtual Private Wire Service (VPWS)
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.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.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.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.55.00.14.00
!
Segment Routing (SR) is a flexible and scalable way of performing source routing. The source device selects a path and encodes it in the packet header as an ordered list of segments. Segments are identifiers for any type of instruction.

Segment routing for traffic engineering (SR-TE) takes place through a tunnel between a source and destination pair. SR-TE uses the concept of source routing, where the source calculates the path and encodes it in the packet header as a segment. In SR-TE preferred path, each segment is an end-to-end path from the source to the destination, and instructs the routers in the provider core network to follow the specified path instead of the shortest path calculated by the IGP. The destination is unaware of the presence of the tunnel.

The user can achieve better resilience and convergence for the network traffic, by transporting MPLS L2VPN services using segment routing, instead of MPLS LDP. Segment routing can be directly applied to the MPLS architecture without changing the forwarding plane. In a segment-routing network that uses the MPLS data plane, LDP or other signaling protocol is not required; instead label distribution is performed by IGP. Removing protocols from the network simplifies its operation and makes it more robust and stable by eliminating the need for protocol interaction. Segment routing utilizes the network bandwidth more effectively than traditional MPLS networks and offers lower latency.

Preferred tunnel path functionality allows you map pseudowires to specific traffic-engineering tunnel paths. Attachment circuits are cross-connected to specific SR traffic engineering tunnel interfaces instead of remote PE router IP addresses reachable using IGP or LDP. Using preferred tunnel path, the traffic engineering tunnel transports traffic between the source and destination PE routers. A path is selected for an SR Policy when the path is valid and its preference is the best (highest value) among all the candidate paths of the SR Policy.

The following L2VPN services are supported over SR-TE policy:

- EVPN VPWS Preferred Path over SR-TE Policy
- L2VPN VPWS Preferred Path over SR-TE Policy
- EVPN VPWS On-Demand Next Hop with SR-TE
- EVPN VPWS Preferred Path over SR-TE Policy, on page 190
- L2VPN VPWS Preferred Path over SR-TE Policy, on page 203
- EVPN VPWS On-Demand Next Hop with SR-TE, on page 215
- Overview of Segment Routing, on page 230
- How Segment Routing Works, on page 230
- Segment Routing Global Block, on page 231
EVPN VPWS Preferred Path over SR-TE Policy

EVPN VPWS Preferred Path over SR-TE Policy feature allows you to set the preferred path between the two end-points for EVPN VPWS pseudowire (PW) using SR-TE policy. SR policy allows you to choose the path on a per EVPN instance (EVI) basis. This feature is supported on bundle attachment circuit (AC) and physical AC.

Restrictions

- If EVPN VPWS with On Demand Next Hop (ODN) is configured, and EVPN VPWS with preferred path is also configured for the same PW, then the preferred-path will take precedence.
- EVPN VPWS SR policy is not supported on EVPN VPWS dual homing.
- EVPN validates if the route is for a single home next hop, otherwise it issues an error message about a dangling SR TE policy, and continue to set up EVPN-VPWS without it. EVPN relies on ESI value being zero to determine if this is a single home or not. If the AC is a Bundle-Ether interface running LACP then you need to manually configure the ESI value to zero to overwrite the auto-sense ESI as EVPN VPWS multihoming is not supported.

To disable EVPN dual homing, configure bundle-Ether AC with ESI value set to zero.

evpn
   interface Bundle-Ether12
      ethernet-segment
         identifier type 0 00.00.00.00.00.00.00.00.00
   /* Or globally */
   Evpn
      ethernet-segment type 1 auto-generation-disable

Topology

Figure 23: 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 1.1.1.1
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
```
Configure Prefix-SID in ISIS

Router(config-sr)# global-block 180000 200000
Router(config-sr)# exit
!
Router# configure
Router(config)# router isis core
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 49.0002.0330.2000.0021.00
Router(config-isis)# nsr
Router(config-isis)# nsf ietf
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide level 2
Router(config-isis-af)# mpls traffic-eng level-2-only
Router(config-isis-af)# mpls traffic-eng router-id loopback0
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# segment-routing prefix-sid-map advertise-local
Router(config-isis-af)# exit
!
Router(config-isis)# interface loopback 0
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-af)# prefix-sid index 180015
Router(config-isis-af)# commit
Router(config-isis-af)# exit

/* Configure Prefix-SID on P2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0022.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 180025
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on PE3 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0030.0030.0030.0035.00
Configure Adjacency-SID in ISIS

Configure Adjacency-SID on PE1, P1, P2, and PE3.

/* Configure Adjacency-SID on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15101
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/1/6
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15102
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on P1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15200
Route(config-isis-if-af)# commit
Configure Adjacency-SID in ISIS

! 
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/7
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15202
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on P2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999

! 
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/7
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# metric 20
Route(config-isis-if-af)# adjacency-sid absolute 15201
Route(config-isis-if-af)# exit

! 
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/5
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# metric 20
Route(config-isis-if-af)# adjacency-sid absolute 15204
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on PE3 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999

! 
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/1
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15301
Route(config-isis-if-af)# exit

! 
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/2
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
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 20.20.20.20
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# 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 20.20.20.20
Configure EVPN VPWS over SR-TE Policy

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: 20.20.20.20
Name: srte_c_9001_ep_20.20.20.20

Router# configure
Router(config)# 12vpn
Router(config-12vpn)# pw-class 1001
Router(config-12vpn-pwc)# encapsulation mpls
Router(config-12vpn-pwc-mls)# preferred-path sr-te policy srte_c_9001_ep_20.20.20.20 failback disable
Router(config-12vpn-pwc-mls)# commit
Router(config-12vpn-pwc-mls)# exit
!
Router(config)# 12vpn
Router(config-12vpn)# xconnect group evpn_vpws
Router(config-12vpn-xc)# p2p evpn_vpws_1001
Router(config-12vpn-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 20.20.20.20, PW ID: evi 1001, ac-id 10001, state is up (established)
Preferred path Inactive: SR TE srte_c_9001_ep_20.20.20.20, Statically configured, fallback enabled
Tunnel: Down
LSP: Up

/* If Fallback Disable is configured, and if the SR-policy is down, or if it is misconfigured in dual homed mode, then the L2VPN PW will be down */
show l2vpn xconnect detail
EVPN: neighbor 20.20.20.20, PW ID: evi 1001, ac-id 10001, state is down (local ready)
Preferred path Active: SR TE srte_c_9001_ep_20.20.20.20, 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 1.1.1.1
  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

PE2:

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 TenGigE0/0/0/7
circuit-type level-2-only
point-to-point
hello-padding disable
address-family ipv4 unicast
  metric 20
  adjacency-sid absolute 15202

P2:

configure
  segment-routing
    local-block 15000 15999

router isis core

interface TenGigE0/0/0/5
circuit-type level-2-only
point-to-point
hello-padding disable
address-family ipv4 unicast
  metric 20
  adjacency-sid absolute 15204

interface TenGigE0/0/0/0/7
circuit-type level-2-only
point-to-point
hello-padding disable
address-family ipv4 unicast
  metric 20
  adjacency-sid absolute 15201
PE3:

configure
   segment-routing
       local-block 15000 15999

router isis core
   !
interface TenGigE0/0/0/1
circuit-type level-2-only
point-to-point
hello-padding disable
address-family ipv4 unicast
   adjacency-sid absolute 15301
   !
interface TenGigE0/0/0/2
circuit-type level-2-only
point-to-point
hello-padding disable
address-family ipv4 unicast
   adjacency-sid absolute 15302
/* Configure Segment-list */

PE1:

configure
   segment-routing
       global-block 180000 200000
   traffic-eng
       logging
       policy status
   segment-routing
       traffic-eng
       segment-list name pref_sid_to_PE3
           index 1 mpls label 180020
   !
   !
configure
   segment-routing
       local-block 15000 15999
   traffic-eng
       logging
       policy status
   segment-routing
       traffic-eng
       segment-list name pref_adj_sid_to_PE3
           index 1 mpls label 15101
           index 2 mpls label 15202
   !
   !
/* Configure SR-TE policy */

segment-routing
   traffic-eng
   policy pref_sid_to_PE3
       color 9001 end-point ipv4 20.20.20.20
       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 20.20.20.20
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_20.20.20.20 fallback disable
xconnect group evpn_vpws
p2p evpn_vpws_1001
interface tengi0/1/0/1.1001
neighbor evpn evi 1001 target 10001 source 20001
pw-class 1001
!

Verify EVPN VPWS Preferred Path over SR-TE Policy Configuration

PE1#show segment-routing traffic-eng forwarding policy name pref_sid_to_PE3 detail
Policy Segment Outgoing Outgoing Next Hop Bytes
Name List Label Interface Switched
----------------- --------------- ----------- ------------------- --------------- ------------
pref_sid_to_PE3 15102 TenGigE0/0/1/6 20.20.20.20 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: 20.20.20.20
Name: srte_c_9001_ep_20.20.20.20

PE1#show mpls forwarding tunnels sr-policy name pref_sid_to_PE3

<table>
<thead>
<tr>
<th>Tunnel Name</th>
<th>Label</th>
<th>Interface</th>
<th>Switched</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>pref_sid_to_PE3 (SR)</td>
<td>15102</td>
<td>TenGigE0/0/1/6</td>
<td>20.20.20.20</td>
<td>836516512</td>
</tr>
</tbody>
</table>

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
- EVVPN: 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, 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 230
- How Segment Routing Works, on page 230
- Segment Routing Global Block, on page 231
L2VPN VPWS Preferred Path over SR-TE Policy

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 24: 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

L2VPN Services over Segment Routing for Traffic Engineering Policy

L2VPN VPWS Preferred Path over SR-TE Policy
Configure Prefix-SID in IS-IS

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 level-2-only
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)# is-type level-2-only
Route(config)# net 49.0002.0330.2000.0021.00
Route(config)# nsr
Route(config)# nsf ietf
Route(config)# log adjacency changes
Route(config)# 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)# is-type level-2-only
Route(config)# net 49.0002.0330.2000.3030.0030.0035.00
Route(config)# nsr
Route(config)# nsf ietf
Route(config)# log adjacency changes
Route(config)# 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
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
Configure SR-TE Policy

/* Configure SR-TE policy */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr)# policy 100
Router(config-sr-te-policy)# color 1 end-point ipv4 2.2.2.2
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 400
Configure VPWS over SR-TE Policy

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: 3.3.3.3
Name: srte_c_1300_ep_3.3.3.3
Router# configure
Router(config)# 12vpn
Router(config-12vpn)# pw-class pw1300
Router(config-12vpn-pw)# encapsulation mpls
Router(config-12vpn-pw-mpls)# load-balancing
Router(config-12vpn-pw-mpls-load-bal)# flow-label both
Router(config-12vpn-pw-mpls-load-bal)# exit!

Router(config-12vpn-pw-mpls)# preferred-path sr-te policy srte_c_1300_ep_3.3.3.3 fallback disable
Router(config-12vpn-pw-mpls)# exit!

Router(config)# 12vpn
Router(config-12vpn)# xconnect group xcon1
Router(config-12vpn-xc)# p2p vplw1002
Router(config-12vpn-xc-p2p)# interface TenGigE0/0/1/1.1002
Router(config-12vpn-xc-p2p)# neighbor 3.3.3.3 pw-id 1002
Router(config-12vpn-xc-p2p-pw)# pw-class pw1300
Router(config-12vpn-xc-p2p-pw)# commit
Router(config-12vpn-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 1.1.1.1
  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 TenGigE0/0/0/0/4
    circuit-type level-2-only
    point-to-point
    hello-padding disable
    address-family ipv4 unicast
    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
    adjacency-sid absolute 15201

interface TenGigE0/0/0/0/9
    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 2.2.2.2
    candidate-paths
      preference 400
        explicit segment-list PE1-PE3-PE2
      !
      preference 500
        explicit segment-list PE1-PE2
  !
  policy 1013
    color 1013 end-point ipv4 2.2.2.2
    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 3.3.3.3
    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_3.3.3.3 fallback disable

Xconnect group xcon1
  pdp vplw1002
interface TenGigE0/0/1/1.1002
neighbor 3.3.3.3 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 1.1.1.1, label 16100 (regular)
Prefix SID:
  Prefix 3.3.3.3, label 16300 (regular)
Prefix SID:
  Prefix 2.2.2.2, 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: 46675 (unprotected) 15200 (unprotected)

PE1# show segment-routing traffic-eng policy candidate-path name 100
SR-TE policy database
---------------------
Color: 100, End-point: 2.2.2.2
Name: srte_c_1_ep_2.2.2.2

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: 2.2.2.2)
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, 2.2.2.2]
  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: 2.2.2.2)
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
  Preference 600:
    Explicit: segment-list PE1-PE3-PE2 (inactive)
      Inactive Reason:
      Weight: 0, Metric Type: IGP
  Preference 500:
    Explicit: segment-list PE1-PE2 (inactive)
      Inactive Reason:
      Weight: 0, Metric Type: IGP
  Preference 200:
    Explicit: segment-list PE1-PE3-PE2-t0016 (inactive)
      Inactive Reason: unresolved first label
      Weight: 0, Metric Type: IGP
  Preference 100:
    Explicit: segment-list PE1-PE2_BE121 (inactive)
      Inactive Reason: unresolved first label
      Weight: 0, Metric Type: IGP
Attributes:
  Binding SID: 27525
  Allocation mode: dynamic
  State: Programmed
  Policy selected: yes
  Forward Class: 0

PE1# show segment-routing traffic-eng forwarding policy name 100
Thu Feb 1 23:19:28.951 EST

Policy Segment Outgoing Outgoing Next Hop Bytes
Name List Label Interface Switched
------------- --------------- ----------- ------------------- --------------- ------------
100 PE1-PE2 Pop 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 Segment Outgoing Outgoing Next Hop Bytes
Name List Label Interface 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 Outgoing Outgoing Next Hop Bytes
## 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 230
- How Segment Routing Works, on page 230
- Segment Routing Global Block, on page 231

## 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 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 setup 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.

```plaintext
evpn
interface Bundle-Ether12
ethernet-segment
  identifier type 0 00.00.00.00.00.00.00.00.00
/* Or globally */
evpn
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 25: 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-if)# prefix-sid index 30101
Route(config-isis-af-if)# 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-if)# prefix-sid index 30102
Route(config-isis-af-if)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/0
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
```
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)# pce
Router(config-sr-te-color-dyn-mpls-pce)# exit
!
Router(config-sr-te)# on-demand color 2
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# exit
!

Configure SR-TE for P and PE routers.
Router(config-sr-te)# on-demand color 3
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# 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)# pce
Router(config-sr-te-color-dyn-mpls-pce)# exit

Router(config-sr-te)# on-demand color 12
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# exit

Router(config-sr-te)# on-demand color 13
Router(config-sr-te-color)# dynamic mpls
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# 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
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

Configure BGP

Configure BGP on PE1 and PE2.

/* Configure BGP on PE1 */

Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 205.1.0.1
Router(config-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-bgp-nbr-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-bgp-nbr-af)# exit
!
Router(config-bgp)# commit

/* Configure BGP on PE2 */

Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 205.1.0.3
Router(config-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-bgp-nbr-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-bgp-nbr-af)# exit
!
Router(config-bgp)# 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)# 12vpn
Router(config-12vpn)# xconnect group evpn_vpws
Router(config-12vpn-xc)# p2p el_10
Router(config-12vpn-xc-p2p)# interface GigE0/0/0/2.2
Router(config-12vpn-xc-p2p)# neighbor evpn evi 2 target 10 source 10
Router(config-12vpn-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)# 12vpn
Router(config-12vpn)# xconnect group evpn_vpws
Router(config-12vpn-xc)# p2p e3_30
Router(config-12vpn-xc-p2p)# interface GigE0/0/0/2.4
Router(config-12vpn-xc-p2p)# neighbor evpn evi 2 target 10 source 10
Router(config-12vpn-xc-p2p)# commit

Configure Flexible Cross-connect Service (F XC) 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)# 12vpn
Router(config-12vpn)# flexible-xconnect-service vlan-unaware evpn_vu
Router(config-12vpn-fxs-vu)# interface GigE0/0/0/2.3
Router(config-12vpn-fxs-vu)# neighbor evpn evi 3 target 20
Router(config-12vpn-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)# 12vpn
Router(config-12vpn)# flexible-xconnect-service vlan-unaware evpn_vu
Router(config-12vpn-fxs-vu)# interface GigE0/0/0/2.3
Router(config-12vpn-fxs-vu)# neighbor evpn evi 3 target 20
Router(config-12vpn-fxs-vu)# commit
/* Configure Prefix-SID in ISIS and TI-LFA */

PE1:

configure
c-router isis ring
  net 49.0001.1921.6800.1001.00
  segment-routing global-block 30100 39100
  nsr
distribute link-state
snsf cisco
  address-family ipv4 unicast
    metric-style wide
mpls traffic-eng level-1
mpls traffic-eng router-id Loopback0
  segment-routing mpls
!
interface Loopback0
  address-family ipv4 unicast
    prefix-sid index 30101
    !
interface HundredGigE0/0/0/0
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
    fast-reroute per-prefix
    fast-reroute per-prefix ti-lfa
    !

P:

configure
c-router isis ring
  net 49.0001.1921.6800.1002.00
  segment-routing global-block 30100 39100
  nsr
distribute link-state
snsf 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
ssegment-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:

```plaintext
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) */
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:

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
!

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 12vpn 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 1

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 230
- How Segment Routing Works, on page 230
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 identifiers 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
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
```
Segment Routing Global Block
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.

The BPDU Transparency with MACsec feature is not supported on:

- N540-28Z4C-SYS-A
- N540-28Z4C-SYS-D
- N540X-16Z4G8Q2C-A
- N540X-16Z4G8Q2C-D
- N540-12Z20G-SYS-A
- N540-12Z20G-SYS-D
- N540X-12Z16G-SYS-A
- N540X-12Z16G-SYS-D

- Layer 2 Control Plane Tunneling in MACsec, on page 233
- MACsec and MKA Overview, on page 234
- L2CP Tunneling, on page 234
- L2CP Tunneling in MACsec, on page 234
- Configuration, on page 235

Layer 2 Control Plane Tunneling in MACsec

The punt decision in Layer 2 Control Plane Tunneling depends on the interface that is configured with MACsec. If the main interface is configured with MACsec policy, all the MACsec packets are punted so that MACsec sessions are established between customer edge (CE) device and the provider edge (PE) device. If the main interface is not configured with MACsec, all MACsec packets are tunnelled to the remote CE.
MACsec and MKA Overview

MACsec is an IEEE 802.1AE standards based Layer 2 hop-by-hop encryption that provides data confidentiality and integrity for media access independent protocols.

MACsec provides MAC-layer encryption over wired networks by using out-of-band methods for encryption keying. The MACsec Key Agreement (MKA) Protocol provides the required session keys and manages the required encryption keys. Only host facing links (links between network access devices and endpoint devices such as a PC or IP phone) can be secured using MACsec.

The 802.1AE encryption with MACsec Key Agreement (MKA) is supported on downlink ports for encryption between the host devices.

MACsec encrypts the entire data except for the Source and Destination MAC addresses of an Ethernet packet.

To provide MACsec services over the WAN or Metro Ethernet, service providers offer Layer 2 transparent services such as E-Line or E-LAN using various transport layer protocols such as Ethernet over Multiprotocol Label Switching (EoMPLS) and L2TPv3.

The packet body in an EAP-over-LAN (EAPOL) Protocol Data Unit (PDU) is referred to as a MACSec Key Agreement PDU (MKPDU). When no MKPDU is received from participants after 3 heartbeats (each heartbeat is of 2 seconds), peers are deleted from the live peer list. For example, if a client disconnects, the participant on the switch continues to operate MKA until 3 heartbeats have elapsed after the last MKPDU is received from the client.

The MKA feature support provides tunneling information such as VLAN tag (802.1Q tag) in the clear so that the service provider can provide service multiplexing such that multiple point to point services can co-exist on a single physical interface and differentiated based on the now visible VLAN ID.

In addition to service multiplexing, VLAN tag in the clear also enables service providers to provide quality of service (QoS) to the encrypted Ethernet packet across the SP network based on the 802.1P (CoS) field that is now visible as part of the 802.1Q tag.

L2CP Tunneling

The Layer 2 control plane is divided into many customer and provider control planes. As defined in the IEEE Standard 802.1Q-2011, an L2CP frame is a frame that contains a destination MAC address that is one among the 32 addresses which are reserved for control protocols. You can transport traffic using VPWS or VPLS service.

L2CP Tunneling in MACsec

The decision to punt depends on the interface that is configured with MACsec. If the interface is configured with MACsec policy, all MACsec packets are punted so that MACsec sessions are established between two customer edge (CE) devices. If the interface is not configured with MACsec, all MACsec packets are tunneled to the remote CE. MACsec cannot be configured on a sub-interface.

When CEs are configured with MACsec and PEs are configured with L2VPN VPWS, all MACsec packets are tunneled through VPWS.
When MACsec is configured on PE on any CE connected interface, all MACsec packets on this interface are}
punted. These packets are not forwarded to remote CEs. When MACsec is configured on the PE’s interface,
MACsec session is not established between PE and CE devices.

Configuration

The following sections describes the procedure for configuring BPDU Transparency with MACsec feature.

- Configure an MPLS core
- Configure L2VPN Xconnect
- Configure MACsec on CE device

Configuring L2VPN Xconnect

Configure IPv4 address on an interface connecting to the core.

```
Router# configure
Router(config)# interface tengige 0/1/0/8/2.1
Router(config-subif)# no shut
Router(config-subif)# ipv4 address 192.0.2.1/24
```

Configure an IPv4 loopback interface.

```
Router# configure
Router(config)# interface loopback 0
Router(config)# ipv4 address 10.0.0.1/32
```

Configure OSPF as IGP.

```
Router# configure
Router(config)# router ospf 100 area 0
Router(config-ospf-ar)# interface Tengige 0/1/0/8/3
Router(config-ospf-ar-if)# exit
Router(config-ospf-ar)# interface loopback 1
```

Configure MPLS LDP for the physical core interface.

```
Router(config-ospf-ar)# mpls ldp
Router(config-ldp)# interface Tengige 0/1/0/8/3
```

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 12vpn 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 12vpn 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)# encap dot1q 1

Configure L2VPN Xconnect with a neighbour which is a pseudowire.

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group g1
Router(config-l2vpn-xc)# p2p g1
Router(config-l2vpn-xc-p2p)# interface TenGigE 0/1/0/2.1
Router(config-l2vpn-xc-p2p)# neighbor 172.16.0.1 pw-id 1

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 04795B232C766A6C513A5C4E37582F220F087178116703312465525017A0C7101
  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 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
FW: neighbor 10.10.10.1, FW ID 1, state is up ( established )
  FW class not set, XC ID 0xc0000001
  Encapsulation MPLS, protocol LDP
  Source address 172.16.0.2
  FW type Ethernet, control word disabled, interworking none
  FW backup disable delay 0 sec
  Sequencing not set

!
Configure BPDU Transparency with MACsec

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

• Gigabit Ethernet Protocol Standards, on page 241
• Carrier Ethernet Model References, on page 241
• Default Configuration Values for Gigabit Ethernet and 10-Gigabit Ethernet, on page 243
• References for Configuring Link Bundles, on page 244

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 241.

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 enabled 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.

---

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>
<thead>
<tr>
<th>Parameter</th>
<th>Configuration File Entry</th>
<th>Default Value</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow control</td>
<td>flow-control</td>
<td>egress on ingress off</td>
<td>none</td>
</tr>
<tr>
<td>MTU</td>
<td>mtu</td>
<td>1514 bytes for normal frames</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1518 bytes for 802.1Q tagged frames</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1522 bytes for QinQ frames</td>
<td></td>
</tr>
<tr>
<td>MAC address</td>
<td>mac address</td>
<td>Hardware burned-in address (BIA^x)</td>
<td>L3 only</td>
</tr>
<tr>
<td>L2 port</td>
<td>l2transport</td>
<td>off/L3</td>
<td>L2 subinterfaces must have L3 main parent interface</td>
</tr>
<tr>
<td>Egress filtering</td>
<td>Ethernet egress-filter</td>
<td>off</td>
<td>none</td>
</tr>
<tr>
<td>Link negotiation</td>
<td>negotiation</td>
<td>off</td>
<td>physical main interfaces only</td>
</tr>
<tr>
<td>Tunneling Ethertype</td>
<td>tunneling ethertype</td>
<td>0x8100</td>
<td>configured on main interface only; applied to subinterfaces only</td>
</tr>
</tbody>
</table>

---

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.
### 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 53.

### 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 Etherokinnet 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.

### Table of Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Configuration File Entry</th>
<th>Default Value</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLAN tag matching</td>
<td>encapsulation</td>
<td>all frames for main interface; only ones specified for subinterfaces</td>
<td>encapsulation command only subinterfaces</td>
</tr>
</tbody>
</table>

1. The restrictions are applicable to L2 main interface, L2 subinterface, L3 main interface, interflex L2 interface etc.

2. burned-in address
• 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.