L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.2.x

First Published: 2017-07-14

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883
THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

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

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

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

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

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

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

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

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: 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 Cisco Systems, Inc. All rights reserved.
CONTENTS

PREFACE

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

CHAPTER 1

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

CHAPTER 2

Configure Gigabit Ethernet for Layer 2 VPNs 3
Introduction to Layer 2 Virtual Private Networks 3
Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces 3
Configure Gigabit Ethernet Interfaces for Layer 2 Transport 5

CHAPTER 3

Configure Layer 2 Access Control Lists 7
Layer 2 Access Control Lists 7
Prerequisites for Configuring Layer 2 Access Control Lists 7
Layer 2 Access Control Lists Feature Highlights 8
Purpose of Layer 2 Access Control Lists 8
How a Layer 2 Access Control List Works 8
Layer 2 Access Control List Process and Rules 8
Create Layer 2 Access Control List 9
Restrictions for Configuring Layer 2 Access Control Lists 9
Configuration 9
Running Configuration 10
Verification 10

CHAPTER 4

Configure Virtual LANs in Layer 2 VPNs 11
Preface

This preface contains these sections:

• Changes to This Document, on page ix
• Obtaining Documentation and Submitting a Service Request, on page ix

Changes to This Document

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 2017</td>
<td>Initial release of this document.</td>
</tr>
<tr>
<td>July 2017</td>
<td>Republished for Release 6.2.2.</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.
Obtaining Documentation and Submitting a Service Request
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

### Table 2: VPN Features Added or Modified in IOS XR Release 6.2.x

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Changed in Release</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>No new features introduced</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
Configure Gigabit Ethernet for Layer 2 VPNs

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

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

• Introduction to Layer 2 Virtual Private Networks, on page 3
• Introduction to Layer 2 VPNs on Gigabit Ethernet Interfaces, on page 3
• Configure Gigabit Ethernet Interfaces for Layer 2 Transport, on page 5

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

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

```plaintext
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)
CHAPTER 3

Configure Layer 2 Access Control Lists

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

- Layer 2 Access Control Lists, on page 7
- Prerequisites for Configuring Layer 2 Access Control Lists, on page 7
- Layer 2 Access Control Lists Feature Highlights, on page 8
- Purpose of Layer 2 Access Control Lists, on page 8
- How a Layer 2 Access Control List Works, on page 8
- Layer 2 Access Control List Process and Rules, on page 8
- Create Layer 2 Access Control List, on page 9
- Restrictions for Configuring Layer 2 Access Control Lists, on page 9
- Configuration, on page 9

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.ffff
Router(config-es-acl)# permit host 000a.000b.000c host 00aa.ab99.1122 cos 1 dei
Router(config-es-acl)# deny host 000a.000b.000c host 00aa.dc11.ba99 cos 7 dei
Router(config-es-acl)# commit
Router(config)# interface tengige0/0/0/4
Router(config-if)# l2transport
Router(config-if-l2)# commit
Running Configuration

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

Verification

Verify that you have configured Layer 2 access control lists.

/* Verify the Layer 2 access control lists configuration */
Router# show access-lists ethernet-services es_acl_1 hardware ingress location 0/0/CPU0
Fri Oct 21 09:39:52.904 UTC
ethernet-services access-list es_acl_1
10 deny 00ff.eedd.0000 ff00.0000.00ff 0000.0100.0000 0000.0000.ffff (2051 matches)
20 permit host 000a.000b.000c host 00aa.ab99.1122 cos 1 dei
30 deny host 000a.000b.000c host 00aa.dc11.ba99 cos 7 dei (2050 matches)
CHAPTER 4

Configure Virtual LANs in Layer 2 VPNs

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

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

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

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

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

VLAN ACs support the following modes of L2VPN operation:

- Basic Dot1Q Attachment Circuit—The Attachment Circuit covers all frames that are received and sent with a specific VLAN tag.
- QinQ Attachment Circuit—The Attachment Circuit covers all frames received and sent with a specific outer VLAN tag and a specific inner VLAN tag. QinQ is an extension to Dot1Q that uses a stack of two tags.
- Q-in-Any Attachment Circuit—The Attachment Circuit covers all frames received and sent with a specific outer VLAN tag and any inner VLAN tag, as long as that inner VLAN tag is not Layer 3 terminated. Q-in-Any is an extension to QinQ that uses wildcarding to match any second tag.
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.

**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:

```
encapsulation dot1q vlan-id
```

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

```
encapsulation dot1ad vlan-id
```

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

- ```
  encapsulation dot1q vlan-id second-dot1q vlan-id
  ```
- ```
  encapsulation dot1ad vlan-id dot1q vlan-id
  ```

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.

```
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.

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

```
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 16)
• 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.

---

**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.
  - Tunnelled 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 19
- **Valid Ingress-Egress Rewrite Combinations**, on page 21

**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 symmetric

Running Configuration

/* Configuration without rewrite */

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

/* Configuration with rewrite */

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

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

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

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

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

/* Configuration without rewrite */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10 second-dot1q 11
/* Configuration with rewrite */
/* PUSH 1 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag push dot1q 20 symmetric
/* TRANSLATE 1-1 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 1-to-1 dot1q 20
/* TRANSLATE 1-2 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 1-to-2 dot1q 20 second-dot1q 30
/* TRANSLATE 2-2 */
interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 2-to-2 dot1q 20 second-dot1q 30

Associated Commands

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

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>
</tbody>
</table>
### Interface Configuration | Ingress Rewrite Action
---|---
dot1q | Pop 1

dot1q | Push 1

dot1q | Push 2

dot1q | Translate 1 to 1

dot1q | Translate 1 to 2

dot1q range | No rewrite

dot1q range | Push 1

dot1q range | Push 2

QinQ | No rewrite

QinQ | Pop 1

QinQ | Push 1

QinQ | Translate 1 to 1

QinQ | Translate 1 to 2

QinQ | Translate 2 to 2

QinQ range | No rewrite

QinQ range | Pop 1

QinQ range | Push 1

QinQ range | Translate 1 to 1

QinQ range | Translate 1 to 2

QinAny | No rewrite

QinAny | Pop 1

QinAny | Push 1

QinAny | Translate 1 to 1

QinAny | Translate 1 to 2

Untagged | No rewrite

Untagged | Push 1

Untagged | Push 2

The following notations are used for the rewrite actions mentioned in the table:
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>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>
</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 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>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ</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>
</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>No rewrite / Translate 1-to-1</td>
<td>QinQ range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite / Translate 1-to-1</td>
<td>QinQ range</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>QinQ range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Pop 1</td>
<td>QinQ range</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range</td>
<td>Pop 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 1</td>
<td>QinQ range</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range</td>
<td>Push 1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td>dot1q</td>
<td>Push 2 / Translate 1-to-2</td>
<td>QinQ range</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>QinQ range</td>
<td>No rewrite</td>
</tr>
<tr>
<td>dot1q</td>
<td>No rewrite</td>
<td>Untagged</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>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>Push 2</td>
<td>dot1q range</td>
<td>Push 2</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>
<tr>
<td>dot1q range</td>
<td>Push 1</td>
<td>QinQ</td>
<td>Translate 1-to-2</td>
</tr>
<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>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</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>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>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>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 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>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>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>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>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>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>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td></td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td></td>
<td>No rewrite / push 1 / Translate 1-to-1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td></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></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></td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td></td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Pop 1</td>
</tr>
<tr>
<td></td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td></td>
<td>Pop 1</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td></td>
<td>Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>No rewrite</td>
</tr>
<tr>
<td></td>
<td>Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Push 1</td>
</tr>
<tr>
<td></td>
<td>Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-1</td>
</tr>
<tr>
<td></td>
<td>Translate 1-to-2</td>
<td>QinQ range / QinAny</td>
<td>Translate 1-to-2</td>
</tr>
<tr>
<td></td>
<td>No rewrite</td>
<td>Untagged</td>
<td>No rewrite</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>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
CHAPTER 5

Configure Link Bundles for Layer 2 VPNs

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

The advantages of link bundling are:

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

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

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

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

- Configure Gigabit Ethernet Link Bundle, on page 31
- Configure VLAN Bundle, on page 34
- References for Configuring Link Bundles, on page 35

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 1: Link Bundle Topology*

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

Configuration

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

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

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

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

Running Configuration

configure
interface Bundle-Ether2
   ipv4 address 50.0.0.1 255.255.255.0
   mac-address 1212.1212.1212
   bundle maximum-active links 32 hot-standby
   bundle minimum-active links 1
   bundle minimum-active bandwidth 30000000
!
interface Bundle-Ether2.201
ipv4 address 12.22.1.1 255.255.255.0
encapsulation dot1q 201
!
interface TenGigE0/0/0/14
bundle id 2 mode on
!
interface TenGigE0/0/0/15
bundle id 2 mode on
!
interface TenGigE0/0/0/16
bundle id 2 mode on
!
interface TenGigE0/0/0/17
bundle id 2 mode on
!

Verification
Verify that the VLAN status is UP.

Router# show interfaces bundle-ether 2.201

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

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

References for Configuring Link Bundles

This section provides references to configuring link bundles. For an overview of link bundles and configurations, see Configure Link Bundles for Layer 2 VPNs, on page 31.
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.
CHAPTER 6

Configure Multipoint Layer 2 Services

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

Note

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

---

Prerequisites for Implementing Multipoint Layer 2 Services, on page 39
Information About Implementing Multipoint Layer 2 Services, on page 40
How to Implement Services, on page 44
Configuration Examples for Multipoint Layer 2 Services, on page 68

---

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.

  Note

  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
```
interface TenGigE0/0/0/13.100
vfi 1
neighbor 192.0.2.1 pw-id 12345
   pw-class mpls_csr
!
!

Virtual Forwarding Instance

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

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

VPLS for an MPLS-based Provider Core

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

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

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

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

VPLS for Layer 2 Switching

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

```bash
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
exi-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 syslogs 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.

**How to Implement Services**

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

**Configuring a Bridge Domain**

These topics describe how to configure a bridge domain:

**Creating a Bridge Domain**

Perform this task to create a bridge domain.

**SUMMARY STEPS**

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

Step 1 configure
Example: RP/0/RP0/CPU0:router# configure
Enters the XR Config mode.

Step 2 l2vpn
Example: RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
Enters L2VPN configuration mode.

Step 3 bridge group bridge-group-name
Example: RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
Creates a bridge group that can contain bridge domains, and then assigns network interfaces to the bridge domain.

Step 4 bridge-domain bridge-domain-name
Example: RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 Use the commit or end command.
commit - Saves the configuration changes and remains within the configuration session.
end - Prompts user to take one of these actions:
  • Yes - Saves configuration changes and exits the configuration session.
  • No - Exits the configuration session without committing the configuration changes.
  • Cancel - Remains in the configuration mode, without committing the configuration changes.

Associating Members with a Bridge Domain

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

• Ethernet and VLAN
• VFI
BVI is not supported in a bridge domain that is configured with VFI.

**SUMMARY STEPS**

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

**DETAILED STEPS**

**Step 1** configure

Example:

RP/0/RP0/CPU0:router# configure

Enters the XR Config mode.

**Step 2** l2vpn

Example:

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

Enters L2VPN configuration mode.

**Step 3** bridge group bridge group name

Example:

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

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

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

Example:

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

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

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

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

**Step 6** *(Optional)*

`static-mac-address { MAC-address }

**Example:**

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

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

**Step 7**

Use the `commit` or `end` command.

- **commit** - Saves the configuration changes and remains within the configuration session.
- **end** - Prompts user to take one of these actions:
  - **Yes** - Saves configuration changes and exits the configuration session.
  - **No** - Exits the configuration session without committing the configuration changes.
  - **Cancel** - Remains in the configuration mode, without committing the configuration changes.

---

**Configuring Bridge Domain Parameters**

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

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

**SUMMARY STEPS**

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

**Step 1**  
**configure**  
**Example:**  
RP/0/RP0/CPU0:router# configure  
Enters the XR Config mode.

**Step 2**  
**l2vpn**  
**Example:**  
RP/0/RP0/CPU0:router(config)# l2vpn  
RP/0/RP0/CPU0:router(config-l2vpn)#  
Enters the l2vpn configuration mode.

**Step 3**  
**bridge group bridge-group-name**  
**Example:**  
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco  
RP/0/RP0/CPU0:router(config-l2vpn-bg)#  
Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

**Step 4**  
**bridge-domain bridge-domain-name**  
**Example:**  
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc  
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#  
Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

**Step 5**  
**flooding disable**  
**Example:**  
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# flooding disable  
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#  
Disables flooding.

**Step 6**  
**mtu bytes**  
**Example:**  
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mtu 1000  
Adjusts the maximum packet size or maximum transmission unit (MTU) size for the bridge domain.
  - Use the *bytes* argument to specify the MTU size, in bytes. The range is from 64 to 65535.
Step 7  Use the **commit** or **end** command.

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

**end** - Prompts user to take one of these actions:

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

---

**Disabling a Bridge Domain**

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

**Step 1**  **configure**

*Example:*

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

Enters the XR Config mode.

**Step 2**  **l2vpn**

*Example:*

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

Enters L2VPN configuration mode.

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

*Example:*

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
```
Configuring a Layer 2 Virtual Forwarding Instance

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

Creating the Virtual Forwarding Instance

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

SUMMARY STEPS

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

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

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

Step 4  bridge-domain  bridge-domain name

Example:

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

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

Step 5  shutdown

Example:

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

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

Step 6  Use the commit or end command.

commit - Saves the configuration changes and remains within the configuration session.
end - Prompts user to take one of these actions:

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

Step 1  configure
Example:
RP/0/RP0/CPU0:router# configure
Enters the XR Config mode.

Step 2  l2vpn
Example:
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
Enters L2VPN configuration mode.

Step 3  bridge group bridge group name
Example:
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4  bridge-domain bridge-domain name
Example:
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5  vfi {vfi-name}
Example:
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6  Use the commit or end command.
commit - Saves the configuration changes and remains within the configuration session.
end - Prompts user to take one of these actions:
- Yes - Saves configuration changes and exits the configuration session.
- No - Exits the configuration session without committing the configuration changes.
- Cancel - Remains in the configuration mode, without committing the configuration changes.
Associating Pseudowires with the Virtual Forwarding Instance

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

**Step 1** `configure`

*Example:*

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

Enters the XR Config mode.

**Step 2** `l2vpn`

*Example:*

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

Enters L2VPN configuration mode.

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

*Example:*

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

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

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

*Example:*

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

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

**Step 5** `vfi { vfi-name }`
Example:

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

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

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

Example:

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

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

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

Step 7 Use the commit or end command.

commit - Saves the configuration changes and remains within the configuration session.
end - Prompts user to take one of these actions:

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

Associating a Virtual Forwarding Instance to a Bridge Domain

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 configure
Example:

RP/0/RP0/CPU0:router# configure

Enters the XR Config mode.

Step 2  l2vpn

Example:

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

Enters the L2VPN configuration mode.

Step 3  bridge group bridge group name

Example:

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

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

Step 4  bridge-domain bridge-domain name

Example:

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

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

Step 5  vfi { vfi name }

Example:

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

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

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

Example:

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

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

• Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

• Use the pw-id keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.
Step 7  

**static-mac-address** \{ **MAC-address** \}

**Example:**

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

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

Step 8  

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

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

**end** - Prompts user to take one of these actions:

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

---

**Attaching Pseudowire Classes to Pseudowires**

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

Step 1  

**configure**

**Example:**

RP/0/RP0/CPU0:router# configure

Enters the XR Config mode.

Step 2  

**l2vpn**

**Example:**

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

Enters the L2VPN configuration mode.
Step 3  bridge group  bridge group name
Example:

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

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

Step 4  bridge-domain  bridge-domain name
Example:

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

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

Step 5  vfi  { vfi-name }
Example:

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

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

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

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

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
  • Use the A.B.C.D argument to specify the IP address of the cross-connect peer.
  • Use the pw-id keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7  pw-class  { class-name }
Example:

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

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

Step 8  Use the commit or end command.
  commit - Saves the configuration changes and remains within the configuration session.
  end - Prompts user to take one of these actions:
  • Yes - Saves configuration changes and exits the configuration session.
  • No - Exits the configuration session without committing the configuration changes.
Configure Multipoint Layer 2 Services

Configuring Pseudowires Using Static Labels

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

**Step 1**

`configure`

**Example:**

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

Enters the XR Config mode.

**Step 2**

`l2vpn`

**Example:**

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

Enters the L2VPN configuration mode.

**Step 3**

`bridge group bridge-group-name`

**Example:**

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

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

**Step 4**

`bridge-domain bridge-domain-name`

**Example:**

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

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

**Step 5**

`vfi { vfi-name }

Example:

RP/0/RP0/CPU0:router(config-l2vpn-bd-bg)# vfi 1
RP/0/RP0/CPU0:router(config-l2vpn-bd-bg-vfi)#

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

**Step 6**

`neighbor { A.B.C.D } { pw-id value }

Example:

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

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

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

**Step 7**

`mpls static label { local value } { remote value }

Example:

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

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

**Step 8**

Use the `commit` or `end` command.

- **commit** - Saves the configuration changes and remains within the configuration session.
- **end** - Prompts user to take one of these actions:
  - **Yes** - Saves configuration changes and exits the configuration session.
  - **No** - Exits the configuration session without committing the configuration changes.
  - **Cancel** - Remains in the configuration mode, without committing the configuration changes.

---

**Disabling a Virtual Forwarding Instance**

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

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

DETAILED STEPS

Step 1 configure
Example:
RP/0/RP0/CPU0:router# configure
Enters the XR Config mode.

Step 2 l2vpn
Example:
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
Enters the L2VPN configuration mode.

Step 3 bridge group bridge group name
Example:
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 bridge-domain bridge-domain name
Example:
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 vfi { vfi-name }
Example:
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

**Step 6**

**shutdown**

Example:

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

Disables the virtual forwarding interface (VFI).

**Step 7**

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

- **commit** - Saves the configuration changes and remains within the configuration session.
- **end** - Prompts user to take one of these actions:
  - **Yes** - Saves configuration changes and exits the configuration session.
  - **No** - Exits the configuration session without committing the configuration changes.
  - **Cancel** - Remains in the configuration mode, without committing the configuration changes.

**Step 8**

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

Example:

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

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

---

**Configuring the MAC Address-related Parameters**

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

The MAC table attributes are set for the bridge domains.

**Configuring the MAC Address Source-based Learning**

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

**SUMMARY STEPS**

1. **configure**
2. **l2vpn**
3. **bridge group bridge group name**
4. **bridge-domain bridge-domainname**
5. **mac**
6. **learning disable**
7. Use the **commit** or **end** command.
8. **show l2vpn bridge-domain [ detail ]**
DETAILED STEPS

Step 1  configure
Example:

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

Enters the XR Config mode.

Step 2  l2vpn
Example:

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

Enters the L2VPN configuration mode.

Step 3  bridge group  bridge group name
Example:

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

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

Step 4  bridge-domain  bridge-domainname
Example:

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

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

Step 5  mac
Example:

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

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6  learning disable
Example:

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

Step 7  Use the commit or end command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:
  • Yes - Saves configuration changes and exits the configuration session.
  • No - Exits the configuration session without committing the configuration changes.
  • Cancel - Remains in the configuration mode, without committing the configuration changes.

Step 8  show l2vpn bridge-domain [ detail ]

Example:

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

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

Configuring the MAC Address Limit

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

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1  configure

Example:

RP/0/RP0/CPU0:router# configure

Enters the XR Config mode.
Step 2  l2vpn

Example:

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

Enters the L2VPN configuration mode.

Step 3  bridge group  bridge group name

Example:

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

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

Step 4  bridge-domain  bridge-domain name

Example:

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

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

Step 5  (Optional) interface  type  interface_id

Example:

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

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

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

Step 6  mac

Example:

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

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 7  limit

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# limit
```
Sets the MAC address limit for action, maximum, and notification and enters L2VPN bridge group bridge domain MAC limit configuration mode.

**Step 8**

```plaintext
maximum { value }
```

*Example:*

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

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

**Step 9**

```plaintext
action { flood | no-flood | shutdown }
```

*Example:*

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

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

**Step 10**

```plaintext
notification { both | none | trap }
```

*Example:*

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

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

**Step 11**

```plaintext
mac limit threshold 80
```

*Example:*

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

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

**Step 12**

Use the `commit` or `end` command.

- **commit** - Saves the configuration changes and remains within the configuration session.
- **end** - Prompts user to take one of these actions:
  - **Yes** - Saves configuration changes and exits the configuration session.
  - **No** - Exits the configuration session without committing the configuration changes.
  - **Cancel** - Remains in the configuration mode, without committing the configuration changes.

**Step 13**

```plaintext
show l2vpn bridge-domain [ detail ]
```

*Example:*

```plaintext
RP/0/RP0/CPU0:router# show l2vpn bridge-domain detail
```
Displays the details about the MAC address limit.

---

**Configuring the MAC Address Aging**

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

**Step 1**

configure

**Example:**

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

Enters the XR Config mode.

**Step 2**

`l2vpn`

**Example:**

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

Enters the L2VPN configuration mode.

**Step 3**

`bridge group bridge-group-name`

**Example:**

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

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

**Step 4**

`bridge-domain bridge-domain-name`

**Example:**
Configuring the MAC Address Aging

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

**Step 5**

`mac`

**Example:**

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

Enters L2VPN bridge group bridge domain MAC configuration mode.

**Step 6**

`aging`

**Example:**

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

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

**Step 7**

`time { seconds }`

**Example:**

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

Configures the maximum aging time.

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

**Step 8**

Use the `commit` or `end` command.

- `commit` - Saves the configuration changes and remains within the configuration session.
- `end` - Prompts user to take one of these actions:
  - `Yes` - Saves configuration changes and exits the configuration session.
  - `No` - Exits the configuration session without committing the configuration changes.
  - `Cancel` - Remains in the configuration mode, without committing the configuration changes.

**Step 9**

`show l2vpn bridge-domain [ detail ]`

**Example:**

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

Displays the details about the aging fields.
Disabling MAC Flush at the Bridge Port Level

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

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

SUMMARY STEPS

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

DETAILED STEPS

---

**Step 1**  
`configure`  
**Example:**  
`RP/0/RP0/CPU0:router# configure`

Enters the XR Config mode.

**Step 2**  
`l2vpn`  
**Example:**  
`RP/0/RP0/CPU0:router(config)# l2vpn`

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

Enters L2VPN configuration mode.

**Step 3**  
`bridge group bridge-group-name`  
**Example:**  
`RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco`

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

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

**Step 4**  
`bridge-domain bridge-domain-name`  
**Example:**  
`RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`

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

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

**Step 5**  
`mac`  
**Example:**
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
Enters l2vpn bridge group bridge domain MAC configuration mode.

**Step 6** port-down flush disable

**Example:**

RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
port-down flush disable
Disables MAC flush when the bridge port becomes nonfunctional.

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

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

**end** - Prompts user to take one of these actions:

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

---

**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:

```plaintext
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:

```plaintext
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), PBBs: 0 (0 up)
List of ACs:
  AC: TenGigE0/2/0/1.7, state is up
    Type VLAN; Num Ranges: 1
    Outer Tag: 21
    VLAN ranges: [22, 22]
    MTU 1508; XC ID 0x208000b; interworking none
    MAC learning: enabled
    Flooding:
      Broadcast & Multicast: enabled
      Unknown unicast: enabled
    MAC aging time: 300 s, Type: inactivity
    MAC limit: 4000, Action: none, Notification: syslog
    MAC limit reached: no
    MAC port down flush: enabled
    MAC Secure: disabled, Logging: disabled
    Split Horizon Group: none
    Dynamic ARP Inspection: disabled, Logging: disabled
    IP Source Guard: disabled, Logging: disabled
    DHCPv4 snooping: disabled
    IGMP Snooping: enabled
    IGMP Snooping profile: none
    MLD Snooping profile: none
    Storm Control: bridge-domain policer
    Static MAC addresses:
    Statistics:
      packets: received 714472608 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 97708776
      bytes: received 88594603392 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 12115888224
    MAC move: 0
    Storm control drop counters:
      packets: broadcast 0, multicast 0, unknown unicast 0
      bytes: broadcast 0, multicast 0, unknown unicast 0
    Dynamic ARP inspection drop counters:
      packets: 0, bytes: 0
    IP source guard drop counters:
      packets: 0, bytes: 0
List of VFIs:
  VFI 222 (up)
    FW: neighbor 1.1.1.1, FW ID 222, state is up (established)
    FW class not set, XC ID 0xc000000a
    Encapsulation MPLS, protocol LDP
    Source address 21.21.21.21
    FW type Ethernet, control word disabled, interworking none
    Sequencing not set
    FW Status TLV in use
      MPLS Local Remote
<table>
<thead>
<tr>
<th>Label</th>
<th>24017</th>
<th>24010</th>
</tr>
</thead>
<tbody>
<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-12)#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-12)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-12)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-12)#l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain test-switch
RP/0/RSP0/CPU0:router(config-l2vpn-bd)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-l2vpn-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bd-ac)#commit
RP/0/RSP0/CPU0:router(config-l2vpn-bd-ac)#end

RP/0/RSP0/CPU0:router(config-l2vpn-bd-ac)#end

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.2.x
This table lists the configuration steps (actions) and the corresponding purpose for this example:

**SUMMARY STEPS**

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

**DETAILED STEPS**

**Step 1**

configure

Enters global configuration mode.

**Step 2**

interface Bundle-ether10
Creates a new bundle trunk interface.

**Step 3**  
**l2transport**  
Changes Bundle-ether10 from an L3 interface to an L2 interface.

**Step 4**  
**interface GigabitEthernet0/2/0/5**  
Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/5.

**Step 5**  
**bundle id 10 mode active**  
Establishes GigabitEthernet0/2/0/5 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

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

**Step 7**  
**bundle id 10 mode active**  
Establishes GigabitEthernet0/2/0/6 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

**Step 8**  
**interface GigabitEthernet0/2/0/0**  
Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/0.

**Step 9**  
**l2transport**  
Change GigabitEthernet0/2/0/0 from an L3 interface to an L2 interface.

**Step 10**  
**interface GigabitEthernet0/2/0/1**  
Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/1.

**Step 11**  
**l2transport**  
Change GigabitEthernet0/2/0/1 from an L3 interface to an L2 interface.

**Step 12**  
**interface TenGigE0/1/0/2**  
Enters interface configuration mode. Changes configuration mode to act on TenGigE0/1/0/2.

**Step 13**  
**l2transport**  
Changes TenGigE0/1/0/2 from an L3 interface to an L2 interface.

**Step 14**  
**l2vpn**  
Enter L2VPN configuration mode.

**Step 15**  
**bridge group examples**  
Creates the bridge group **examples**.

**Step 16**  
**bridge-domain test-switch**  
Creates the bridge domain **test-switch**, that is a member of bridge group **examples**.

**Step 17**  
**interface Bundle-ether10**  
Establishes Bundle-ether10 as an AC of bridge domain test-switch.

**Step 18**  
**exit**
Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 19**

interface GigabitEthernet0/2/0/0

Establishes GigabitEthernet0/2/0/0 as an AC of bridge domain test-switch.

**Step 20**

exit

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 21**

interface GigabitEthernet0/2/0/1

Establishes GigabitEthernet0/2/0/1 as an AC of bridge domain test-switch.

**Step 22**

exit

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 23**

interface TenGigE0/1/0/2

Establishes interface TenGigE0/1/0/2 as an AC of bridge domain test-switch.

**Step 24**

Use the commit or end command.

- **commit** - Saves the configuration changes and remains within the configuration session.
- **end** - Prompts user to take one of these actions:
  - **Yes** - Saves configuration changes and exits the configuration session.
  - **No** - Exits the configuration session without committing the configuration changes.
  - **Cancel** - Remains in the configuration mode, without committing the configuration changes.

---

**Bridging on Ethernet Flow Points: Example**

This example shows how to configure a Cisco NCS 5500 Series Router to perform Layer 2 switching on traffic that passes through Ethernet Flow Points (EFPs). EFP traffic typically has one or more VLAN headers. Although both IOS XR trunks and IOS XR EFPs can be combined as attachment circuits in bridge domains, this example uses EFPs exclusively.

**Important notes:**

- An EFP is a Layer 2 subinterface. It is always created under a trunk interface. The trunk interface must exist before the EFP is created.

- In an empty configuration, the bundle interface trunk does not exist, but the physical trunk interfaces are automatically configured. Therefore, only the bundle trunk is created.

- In this example the subinterface number and the VLAN IDs are identical, but this is out of convenience, and is not a necessity. They do not need to be the same values.

- The bridge domain test-efp has three attachment circuits (ACs). All the ACs are EFPs.

- Only frames with a VLAN ID of 999 enter the EFPs. This ensures that all the traffic in this bridge domain has the same VLAN encapsulation.
• The ACs in this example use interfaces that are in the admin down state (unresolved state). Bridge domains that use nonexistent interfaces as ACs are legal, and the commit for such configurations does not fail. In this case, the status of the bridge domain shows unresolved until you configure the missing interface.

Configuration Example

RP/0/RSP1/CPU0:router#configure
RP/0/RSP1/CPU0:router(config)#interface Bundle-ether10
RP/0/RSP1/CPU0:router(config-if)#interface Bundle-ether10.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface GigabitEthernet0/6/0/5
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/6
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/7.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface 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
RP/0/RSP1/CPU0:router(config-l2vpn-bd-ac)#exit
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**
interface TenGigE0/1/0/2.999 l2transport
Step 12  
encapsulation dot1q 999  
Assigns VLAN ID of 999 to this EFP.

Step 13  
l2vpn  
Enter L2VPN configuration mode.

Step 14  
bridge group examples  
Creates the bridge group named examples.

Step 15  
bridge-domain test-efp  
Creates the bridge domain named test-efp, that is a member of bridge group examples.

Step 16  
interface Bundle-ether10.999  
Establishes Bundle-ether10.999 as an AC of the bridge domain named test-efp.

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

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

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

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

Step 21  
Use the commit or end command.

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

end - Prompts user to take one of these actions:

• Yes - Saves configuration changes and exits the configuration session.

• No - Exits the configuration session without committing the configuration changes.

• Cancel - Remains in the configuration mode, without committing the configuration changes.
CHAPTER 7

Configure Point-to-Point Layer 2 Services

This section introduces you to point-to-point Layer 2 services, and also describes the configuration procedures to implement it.

The following point-to-point services are supported:

- Local Switching—A point-to-point internal circuit on a router, also known as local connect.
- Attachment circuit—a connection between a PE-CE router pair.
- Pseudowires—a virtual point-to-point circuit from one PE router to another. Pseudowires are implemented over the MPLS network.

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

- Ethernet over MPLS, on page 79
- Configure Local Switching Between Attachment Circuits, on page 82
- Flexible Cross-Connect Service, on page 86
- Flexible Cross-Connect Service Supported Modes, on page 88
- Configure Preferred Tunnel Path, on page 102
- 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 2: Ethernet Port Mode Packet Flow*

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 3: VLAN Mode Packet Flow*

At the egress VLAN PE, the PE associates a VLAN tag to the frames coming off of the pseudowire and after switching the traffic internally, it sends out the traffic on an Ethernet trunk port.

**Note**

Because the port is in trunk mode, the VLAN PE doesn't remove the VLAN tag and forwards the frames through the port with the added tag.
Inter-AS Mode

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

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

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

Figure 4: EoMPLS over Inter-AS: Basic Double AS Topology

QinQ Mode

QinQ is an extension of 802.1Q for specifying multiple 802.1Q tags (IEEE 802.1QinQ VLAN Tag stacking). Layer 3 VPN service termination and L2VPN service transport are enabled over QinQ sub-interfaces.

Cisco 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 5: 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 6: 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.

```bash
/* 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-if-l2)# 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-if)# no shutdown
Router(config)# exit
Router(config)# interface TenGigE 0/0/0/2
Router(config-if)# bundle id 3 mode on
Router(config-if)# no shutdown
Router(config)# exit
Router(config)# interface TenGigE 0/0/0/9
Router(config-if)# bundle id 2 mode on
Router(config-if)# no shutdown
Router(config)# exit
Router(config)# interface TenGigE 0/0/0/8
Router(config-if)# bundle id 2 mode on
Router(config-if)# no shutdown
Router(config)# exit

/* Configure Layer 2 transport on the ethernet link bundles */
Router(config)# interface Bundle-Ether 3 l2transport
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface Bundle-Ether 2 l2transport
Router(config-if)# no shutdown
Router(config-if)# exit

/* 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
/ * 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 0/0/0/1.1 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 TenGigE0/0/0.1
    interface TenGigE0/0/0/1.1
    !
    !
    !

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
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.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
<th>ST</th>
<th>Description</th>
<th>ST</th>
<th>Description</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>
</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>
</tr>
</tbody>
</table>

Associated Commands

- `interface (p2p)`
- `l2vpn`  
- `p2p`  
- `xconnect group`
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 7: 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 8: 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 (PE3 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(config)# 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)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxs1
Router(config-l2vpn-fxs-vu)# interface GigabitEthernet 0/2/0/3.1
Router(config-l2vpn-fxs-vu)# interface GigabitEthernet 0/2/0/0.1
Router(config-l2vpn-fxs-vu)# neighbor evpn evi 1 target 1
Router(config-l2vpn-fxs-vu)# commit

/* Configure PE2 */
Router(config)# 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

Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface GigabitEthernet 0/0/0/0.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxs1
Router(config-l2vpn-fxs-vu)# interface GigabitEthernet 0/0/0/3.1
Router(config-l2vpn-fxs-vu)# interface GigabitEthernet 0/0/0/0.1
Router(config-l2vpn-fxs-vu)# neighbor evpn evi 1 target 1
Router(config-l2vpn-fxs-vu)# commit

Running Configuration

/* On PE1 */
!
Configure
interface GigabitEthernet 0/2/0/3.1 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 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
!
12vpn
  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
!
12vpn
  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)# 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)# 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)# 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)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether20.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
Router(config-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

Running Configuration

/* On PE1 */

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
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.

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

Running Configuration

/* On PE1 */

Configure
interface GigabitEthernet 0/2/0/7.1 l2transport
encapsulation dot1q 1
rewrite ingress tag translate 1-to-2 dot1q 500 second-dot1q 100 symetric
!

Configure
interface GigabitEthernet 0/2/0/7.2 l2transport
encapsulation dot1q 2
rewrite ingress tag translate 1-to-2 dot1q 600 second-dot1q 200 symetric
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 PE2 */
Router# configure
Router(config)# 12vpn
Router(config-12vpn)# flexible-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)# interface Bundle-Ether2.1 12transport
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-12vpn)# interface Bundle-Ether3.1 12transport
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
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether4
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether5
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 33.44.55.66.77.88.99.aa.bb
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether5
/* Configure PE4 */

Router(config)# configure
Router(config)# 12vpn
Router(config-12vpn)# flexible-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)# 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-12vpn)# 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-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 l2transport
    encapsulation dot1q 1
    rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether3.1 l2transport
    encapsulation dot1q 2
    rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

!

evpn
interface Bundle-Ether2
    ethernet-segment identifier type 0 22.33.44.55.66.77.88.99.aa
interface Bundle-Ether3
ethernet-segment identifier type 0 33.44.55.66.77.88.99.aa.bb

/* On PE2 */

configure
l2vpn
    flexible-xconnect-service vlan-aware evi 6
    interface Bundle-Ether2.1
    interface Bundle-Ether3.1

configure
interface Bundle-Ether2.1 l2transport
    encapsulation dot1q 1
    rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

configure
interface Bundle-Ether3.1 l2transport
    encapsulation dot1q 2
    rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

evpn
    interface Bundle-Ether2
    ethernet-segment identifier type 0 22.33.44.55.66.77.88.99.aa
    interface Bundle-Ether3
    ethernet-segment identifier type 0 33.44.55.66.77.88.99.aa.bb

/* On PE3 */

configure
l2vpn
    flexible-xconnect-service vlan-aware evi 6
    interface Bundle-Ether4.1
    interface Bundle-Ether5.1

configure
interface Bundle-Ether4.1 l2transport
    encapsulation dot1q 1
    rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

configure
interface Bundle-Ether5.1 l2transport
    encapsulation dot1q 2
    rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

evpn
    interface Bundle-Ether4
    ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.14.00
    interface Bundle-Ether5
    ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.15.00
Configure Point-to-Point Layer 2 Services

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 9: Local Switching
Configure Multi-Homed Flexible Cross-Connect Service using Local Switching

This section describes how you can configure multi-homed flexible cross-connect service using local switching.

/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-l2vpn-fxs)# 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-2 dot1q 3 second-dot1q 3 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 22.33.44.55.66.77.88.99.aa
Router(config-evpn-ac-es)# commit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# interface Bundle-Ether3
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 33.44.55.66.77.88.99.aa.bb
Router(config-evpn-ac-es)# commit

/* Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 6
Router(config-l2vpn-fxs)# 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-2 dot1q 3 second-dot1q 3 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether3.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-2 dot1q 3 second-dot1q 3 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 22.33.44.55.66.77.88.99.aa
Router(config-evpn-ac-es)# commit
Configure Point-to-Point Layer 2 Services

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)# l2vpn

/* Configure pseudowire class name */
Router(config-l2vpn)# pw-class path1

/* Configure MPLS encapsulation for the pseudowire */
Router(config-l2vpn-pwc)# encapsulation mpls

/* Configure preferred path tunnel settings. 
If fallback disable configuration is used, and when 
the TE/ tunnel is configured, 
if the preferred path goes down, 
the corresponding pseudowire can also go down. */

Router(config-l2vpn-pwc-encap-mpls)# preferred-path
   interface tunnel-te 11 fallback disable

/* Commit your configuration */
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# commit

Running Configuration

Router# show running-configuration
!
l2vpn
   pw-class path1
       encapsulation mpls
           preferred-path interface tunnel-te 11 fallback disable
!
!
Configure Pseudowire Redundancy

Pseudowire redundancy allows you to configure your network to detect a failure in the network and reroute the Layer 2 service to another endpoint that can continue to provide service. This feature provides the ability to recover from a failure of either the remote provider edge (PE) router or the link between the PE and customer edge (CE) routers.

L2VPNs can provide pseudowire resiliency through their routing protocols. When connectivity between end-to-end PE routers fails, an alternative path to the directed LDP session and the user data takes over. However, there are some parts of the network in which this rerouting mechanism does not protect against interruptions in service.

Pseudowire redundancy enables you to set up backup pseudowires. You can configure the network with redundant pseudowires and redundant network elements.

Prior to the failure of the primary pseudowire, the ability to switch traffic to the backup pseudowire is used to handle a planned pseudowire outage, such as router maintenance.

**Note**

Pseudowire redundancy is provided only for point-to-point Virtual Private Wire Service (VPWS) pseudowires.

**Configuration**

This section describes the configuration for pseudowire redundancy.

```bash
/* Configure a cross-connect group with a static point-to-point cross connect */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group A
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface tengige 0/0/0/0.2
Router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 2

/*Configure the pseudowire segment for the cross-connect group */
Router(config-l2vpn-xc-p2p-pw)# pw-class path1

/*Configure the backup pseudowire segment for the cross-connect group */
Router(config-l2vpn-xc-p2p-pw)# backup neighbor 10.2.2.2 pw-id 5

/*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
```

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.2.x
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
!
...

Configure Pseudowire Redundancy

Configure Point-to-Point Layer 2 Services
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 107
- EVPN Operation, on page 108
- EVPN Route Types, on page 109
- Configure EVPN L2 Bridging Service, on page 110
- EVPN Software MAC Learning, on page 112
- EVPN Out of Service, on page 119
- EVPN Routing Policy, on page 122

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>
<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>Timer</td>
<td>Range</td>
<td>Default Value</td>
<td>Trigger</td>
<td>Applicability</td>
<td>Action</td>
<td>Sequence</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>---------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>recovery</td>
<td>0-3600s</td>
<td>30s</td>
<td>node recovered, interface recovered **</td>
<td>Single-Homed (ESI configured), 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>

**Note**

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

* indicates all required software components are loaded.

** indicates link status is up.

**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.
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.

**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>Route Type</td>
<td>Name</td>
<td>Usage</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------</td>
<td>--------------------------------------------</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.

```c
/* Configure address family session in BGP */
```
configure
router bgp 200
  router-id 209.165.200.227
  address-family l2vpn evpn
  neighbor 10.10.10.10
description MPLSFACING-PEER
  updatesource Loopback 0
  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-bgd)# route-target export 100:6005
Router(config-evpn-evi-bgd)# 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-bgrd)# bridge-domain 1-1
Router(config-l2vpn-bgrd-bd)# interface GigabitEthernet 0/0/0/1.1
Router(config-l2vpn-bgrd-bd-ac)# evi 6005
Router(config-l2vpn-bgrd-bd-ac-evi)# commit
Router(config-l2vpn-bgrd-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
description MPLSFACING-PEER
  updatesource Loopback 0
  address-family l2vpn evpn

configure
evpn
evi 6005
bgp
  rd 200:50
  route-target import 100:6005
  route-target export 100:6005

configure
l2vpn
  bridge group 1
  bridge-domain 1-1
  interface GigabitEthernet 0/0/0/1.1
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 11: EVPN Software MAC Learning

The above figure illustrates the process of software MAC learning. The following are the steps involved in the process:

1. Traffic comes in on one port in the bridge domain.
2. The source MAC address (AA) is learnt on the PE and is stored as a dynamic MAC entry.
3. The MAC address (AA) is converted into a type-2 BGP route and is sent over BGP to all the remote PEs in the same EVI.
4. The MAC address (AA) is updated on the PE as a remote MAC address.

Configure EVPN Software MAC Learning

The following section describes how you can configure EVPN Software MAC Learning:

Note

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

Note

The router does not support flow-aware transport (FAT) pseudowire.

/* Configure bridge domain */
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_SH
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface TenGigE0/4/0/10.2001
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 12: 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)# 12vpn
RP/0/RSP0/CPU0:router(config-12vpn)# bridge group EVPN_ALL_ACTIVE
/* Configure advertisement of MAC routes. */
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac

/* Configure address family session in BGP. */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 09.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn

Running Configuration

l2vpn
bridge group EVPN_ALL_ACTIVE
bridge-domain EVPN_2001
interface BundleEther1.2001
evi 2001
!
evpn
  evi 2001
  advertise-mac
!
router bgp 200 bgp
  router-id 40.40.40.40
  address-family l2vpn evpn
  neighbor 10.10.10.10
description MPLS-FACING-PEER
  updatesource Loopback 0
description MPLS-FACING-PEER
  address-family l2vpn evpn

Verification

Verify EVPN in single home devices.

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

<table>
<thead>
<tr>
<th>Ethernet Segment Id</th>
<th>Interface</th>
<th>Nexthops</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Te0/4/0/10</td>
<td>20.20.20.20</td>
</tr>
</tbody>
</table>

Topology:
Operational : SH
Configured : Single-active (AApS) (default)

Dual Home Device—All-Active Load Balancing Mode

The following section describes how you can configure EVPN Software MAC Learning feature in dual home device (DHD) in all-active load balancing mode:
All-active load-balancing is known as Active/Active per Flow (AApF). In the above figure, identical Ethernet Segment Identifier is used on both EVPN PEs. PEs are attached to Ethernet Segment using bundle interfaces. In the CE, single bundles are configured towards two EVPN PEs. In this mode, the MAC address that is learnt is stored on both PE1 and PE2. Both PE1 and PE2 can forward the traffic within the same EVI.

Configure EVPN Software MAC Learning in Dual Home Device—All-Active Mode

This section describes how you can configure EVPN Software MAC Learning feature in dual home device—all-active mode:

/* Configure bridge domain. */
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface 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
RP/0/RSP0/CPU0:router(config-evpn-evi)# exit
RP/0/RSP0/CPU0:router(config-evpn)# interface bundle-ether1
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# identifier type 0 01.11.00.00.00.00.00.01

/* Configure address family session in BGP. */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 209.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description 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# configure
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# configure
RP/0/RSP0/CPU0:router(config)# interface bundle-Ether1.2001 l2transport
RP/0/RSP0/CPU0:router(config-if)# encapsulation dot1aq 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
        ethernet-segment
          identifier type 0 01.11.00.00.00.00.00.00.01
          !
          !
          router bgp 200
          bgp router-id 209.165.200.227
          address-family l2vpn evpn
            !
          neighbor 10.10.10.10
            remote-as 200
            description MPLS-FACING-PEER
            update-source Loopback0
            address-family l2vpn evpn
            !
            interface Bundle-Ether1
              lACP switchover suppress-flaps 300
              load-interval 30
              !
              interface bundle-Ether1.2001 l2transport
                encapsulation dot1aq 2001
                rewrite ingress tag pop 1 symmetric

Verification

Verify EVPN in dual home devices in All-Active mode.

RP/0/RSP0/CPU0: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>--------</td>
<td>---------</td>
<td>--------</td>
</tr>
</tbody>
</table>
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
evi: 2001
  XC ID 0x80000458
  Statistics:
  packets received 28907734874 (unicast 9697466652), sent 76882059953
  bytes received 5550285095808 (unicast 1861913597184), sent 14799781851396
  MAC move: 0
  List of ACs:
    AC: TenGigE0/4/0/10.2001, state is up
    Type VLAN; Num Ranges: 1

  Statistics:
    packets received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 4557394908
    bytes received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 875013222336
    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
```

<table>
<thead>
<tr>
<th>Neighbor</th>
<th>Spk</th>
<th>AS</th>
<th>MsgRcvd</th>
<th>MsgSent</th>
<th>TblVer</th>
<th>InQ</th>
<th>OutQ</th>
<th>Up/Down</th>
<th>St/PfxRcd</th>
</tr>
</thead>
<tbody>
<tr>
<td>209.165.200.225</td>
<td>0</td>
<td>200</td>
<td>216739</td>
<td>229871</td>
<td>200781341</td>
<td>0</td>
<td>0</td>
<td>3d00h</td>
<td>348032</td>
</tr>
<tr>
<td>209.165.201.30</td>
<td>0</td>
<td>200</td>
<td>6462962</td>
<td>4208831</td>
<td>200781341</td>
<td>10</td>
<td>0</td>
<td>2d22h</td>
<td>35750</td>
</tr>
</tbody>
</table>

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
```

<table>
<thead>
<tr>
<th>Topo ID</th>
<th>Producer</th>
<th>Next Hop(s)</th>
<th>Mac Address</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1112</td>
<td>0/6/CPU0</td>
<td>Te0/6/0/1.36001</td>
<td>00a3.0001.0001</td>
<td></td>
</tr>
</tbody>
</table>

Verify the MAC updates to the L2FIB table in a route switch processor (RSP).

```
RP/0/RSP0/CPU0:router# show l2vpn mac mac all location 0/6/cPU0
```

<table>
<thead>
<tr>
<th>Topo ID</th>
<th>Producer</th>
<th>Next Hop(s)</th>
<th>Mac Address</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1112</td>
<td>0/6/CPU0</td>
<td></td>
<td>00a3.0001.0001</td>
<td></td>
</tr>
</tbody>
</table>

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
```

<table>
<thead>
<tr>
<th>Mac Address</th>
<th>Type</th>
<th>Learned from/Filtered on</th>
<th>LC learned</th>
<th>Resync Age/Last Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000.2001.5555</td>
<td>dynamic</td>
<td>Te0/0/0/2/0.2001</td>
<td>N/A</td>
<td>11 Jan 14:37:22</td>
</tr>
<tr>
<td>N/A</td>
<td>local dynamic</td>
<td></td>
<td>N/A</td>
<td>11 Jan 14:37:22</td>
</tr>
<tr>
<td>00bb.2001.0001</td>
<td>dynamic</td>
<td>Te0/0/0/2/0.2001</td>
<td>N/A</td>
<td>11 Jan 14:37:22</td>
</tr>
<tr>
<td>N/A</td>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>0000.2001.1111</td>
<td>EVPN</td>
<td>BD id: 1110</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>N/A</td>
<td>remote static</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>00a9.2002.0001</td>
<td>EVPN</td>
<td>BD id: 1110</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

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
```

<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>2001</td>
<td>00a9.2002.0001</td>
<td>::</td>
<td>10.10.10.10</td>
<td>34226</td>
</tr>
<tr>
<td>2001</td>
<td>00a9.2002.0001</td>
<td>::</td>
<td>209.165.201.30</td>
<td>34202</td>
</tr>
<tr>
<td>2001</td>
<td>0000.2001.5555</td>
<td>20.1.5.55</td>
<td>TenGigE0/0/0/2/0.2001</td>
<td>34203</td>
</tr>
</tbody>
</table>

```
RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 mac 00a9.2002.0001 detail
```

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.2.x
**EVPN Out of Service**

The EVPN Out of Service feature enables you to control the state of bundle interfaces that are part of an Ethernet segment that have Link Aggregation Control protocol (LACP) configured. This feature enables you to put a node out of service (OOS) without having to manually shutdown all the bundles on their provider edge (PE).

Use the `cost-out` command to bring down all the bundle interfaces belonging to an Ethernet VPN (EVPN) Ethernet segment on a node. The Ethernet A-D Ethernet Segment (ES-EAD) routes are withdrawn before shutting down the bundles. The PE signals to the connected customer edge (CE) device to bring down the corresponding bundle member. This steers away traffic from this PE node without traffic disruption. The traffic that is bound for the Ethernet segment from the CE is directed to the peer PE in a multi-homing environment.

Note: EVPN cost-out is supported only on manually configured ESIs.

In the following topology, the CE is connected to PE1 and PE2. When you configure the `cost-out` command on PE1, all the bundle interfaces on the Ethernet segment are brought down. Also, the corresponding bundle member is brought down on the CE. Hence, the traffic for this Ethernet segment is now sent to PE2 from the CE.

```
EVPN Out of Service

Verify the BGP routes associated with EVPN with bridge-domain filter.

RP/0/RSP0/CPU0:router# show bgp l2vpn evpn bridge-domain EVPN_2001 route-type 2
*> [2][0][48][00bb.2001.0001][0]/104
  0.0.0.0 0 i <------ locally learnt MAC
* i [2][0][48][00a9.2002.00be][0]/104
  10.10.10.10 100 0 i <------ remotely learnt MAC
* i 209.165.201.30 100 0 i

EVPN Out of Service

The EVPN Out of Service feature enables you to control the state of bundle interfaces that are part of an Ethernet segment that have Link Aggregation Control protocol (LACP) configured. This feature enables you to put a node out of service (OOS) without having to manually shutdown all the bundles on their provider edge (PE).

Use the `cost-out` command to bring down all the bundle interfaces belonging to an Ethernet VPN (EVPN) Ethernet segment on a node. The Ethernet A-D Ethernet Segment (ES-EAD) routes are withdrawn before shutting down the bundles. The PE signals to the connected customer edge (CE) device to bring down the corresponding bundle member. This steers away traffic from this PE node without traffic disruption. The traffic that is bound for the Ethernet segment from the CE is directed to the peer PE in a multi-homing environment.

Note: EVPN cost-out is supported only on manually configured ESIs.

In the following topology, the CE is connected to PE1 and PE2. When you configure the `cost-out` command on PE1, all the bundle interfaces on the Ethernet segment are brought down. Also, the corresponding bundle member is brought down on the CE. Hence, the traffic for this Ethernet segment is now sent to PE2 from the CE.

EVPN Out of Service

Verify the BGP routes associated with EVPN with bridge-domain filter.

RP/0/RSP0/CPU0:router# show bgp l2vpn evpn bridge-domain EVPN_2001 route-type 2
*> [2][0][48][00bb.2001.0001][0]/104
  0.0.0.0 0 i <------ locally learnt MAC
* i [2][0][48][00a9.2002.00be][0]/104
  10.10.10.10 100 0 i <------ remotely learnt MAC
* i 209.165.201.30 100 0 i

The EVPN Out of Service feature enables you to control the state of bundle interfaces that are part of an Ethernet segment that have Link Aggregation Control protocol (LACP) configured. This feature enables you to put a node out of service (OOS) without having to manually shutdown all the bundles on their provider edge (PE).

Use the `cost-out` command to bring down all the bundle interfaces belonging to an Ethernet VPN (EVPN) Ethernet segment on a node. The Ethernet A-D Ethernet Segment (ES-EAD) routes are withdrawn before shutting down the bundles. The PE signals to the connected customer edge (CE) device to bring down the corresponding bundle member. This steers away traffic from this PE node without traffic disruption. The traffic that is bound for the Ethernet segment from the CE is directed to the peer PE in a multi-homing environment.

Note: EVPN cost-out is supported only on manually configured ESIs.

In the following topology, the CE is connected to PE1 and PE2. When you configure the `cost-out` command on PE1, all the bundle interfaces on the Ethernet segment are brought down. Also, the corresponding bundle member is brought down on the CE. Hence, the traffic for this Ethernet segment is now sent to PE2 from the CE.

```
EVPN Out of Service

Verify the BGP routes associated with EVPN with bridge-domain filter.

RP/0/RSP0/CPU0:router# show bgp l2vpn evpn bridge-domain EVPN_2001 route-type 2
*> [2][0][48][00bb.2001.0001][0]/104
  0.0.0.0 0 i <------ locally learnt MAC
* i [2][0][48][00a9.2002.00be][0]/104
  10.10.10.10 100 0 i <------ remotely learnt MAC
* i 209.165.201.30 100 0 i

```
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.

```bash
/* 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 : 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
EVVPN 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
EVVPN 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 : 5
  MAC : 5
EVPN Routing Policy

The EVPN Routing Policy feature provides the route policy support for address-family L2VPN EVPN. This feature adds EVPN route filtering capabilities to the routing policy language (RPL). The filtering is based on various EVPN attributes.
A routing policy instructs the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another.

This feature enables you to configure route-policies using EVPN network layer reachability information (NLRI) attributes of EVPN route type 1 to 5 in the route-policy match criteria, which provides more granular definition of route-policy. For example, you can specify a route-policy to be applied to only certain EVPN route-types or any combination of EVPN NLRI attributes. This feature provides flexibility in configuring and deploying solutions by enabling route-policy to filter on EVPN NLRI attributes.

To implement this feature, you need to understand the following concepts:

- Routing Policy Language
- Routing Policy Language Structure
- Routing Policy Language Components
- Routing Policy Language Usage
- Policy Definitions
- Parameterization
- Semantics of Policy Application
- Policy Statements
- Attach Points

For information on these concepts, see Implementing Routing Policy.

Currently, this feature is supported only on BGP neighbor "in" and "out" attach points. The route policy can be applied only on inbound or outbound on a BGP neighbor.

**EVPN Route Types**

The EVPN NLRI has the following different route types:

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

The Ethernet (AD) routes are advertised on per EVI and per Ethernet Segment Identifier (ESI) basis. These routes are sent per Ethernet segment (ES). They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed.

An Ethernet A-D route type specific EVPN NLRI consists of the following fields:
NLRI Format: Route-type 1:

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

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

Path attributes: [MPLS Label]

Example

route-policy evpn-policy
  if rd in (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 (1.1.1:0) [and/or evpn-route-type is 1] [and/or esi in
  (00a1.a2a3.a4a5.a6a7.a8a9)] [and/or etag is 4294967295] then
    set ..
  endif
end-policy

Route Type 2: MAC/IP Advertisement Route

The host's IP and MAC addresses are advertised to the peers within NLRI. The control plane learning of MAC addresses reduces unknown unicast flooding.

A MAC/IP Advertisement Route type specific EVPN NLRI consists of the following fields:
NLRI Format: Route-type 2:

```
[Type][Len][RD][ESI][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr][MPLS Label1][MPLS Label2]
```

Net attributes: [Type][RD][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr]

Path attributes: [ESI], [MPLS Label1], [MPLS Label2]

Example

```bash
route-policy evpn-policy
  if rd in (1.1.1.2:0) [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:
NLRI Format: Route-type 3:

```
[Type][Len][RD][ETag][IP Addr Len][Originating Router's IP Addr]
```

Net attributes: [Type][RD][ETag][IP Addr Len][Originating Router's IP Addr]

Example

```
route-policy evpn-policy
    if rd in (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:
### NLRI Format: Route-type 4:

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

**Net attributes:**

```
[Type][Len][RD][ESI][IP Addr Len][Originating Router's IP Addr]
```

**Example**

```bash
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:
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.

---

**EVPN Features**

**EVPN RPL Attribute**

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

**Net attributes:** [Type] [RD] [ETag] [IP Addr Len] [IP Addr] [GW IP Addr]

**Path attributes:** [ESI], [GW IP Addr], [Label]

---

**Example**
```
route-policy evpn-policy
  if rd in (30.30.30.30:1) [and/or evpn-route-type is 5] [and/or esi in (0000.0000.0000.0000.0000)] [and/or etag is 0] [and/or destination in (12.2.0.0/16)] [and/or evpn-gateway in (0.0.0.0)] then
    set ..
  endif
end-policy
```
Example

evpn-route-type is 3

The following are the various EVPN route types that can be used:
1 - ethernet-ad
2 - mac-advertisement
3 - inclusive-multicast
4 - ethernet-segment
5 - ip-advertisement

IP Prefix
An IP prefix attribute holds IPv4 or IPv6 prefix match specification, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. When IP prefix is specified in EVPN route type 2, it represents either a IPv4 or IPv6 host IP Address (/32 or /128). When IP prefix is specified in EVPN route type 5, it represents either IPv4 or IPv6 subnet. It is a net attribute in EVPN route type 2 and 5.

Example

destination in (128.47.10.2/32)
destination in (128.47.0.0/16)
destination in (128:47::1/128)
destination in (128:47::0/112)

esi
An Ethernet Segment Identifier (ESI) attribute consists of 10 octets. It is a net attribute in EVPN route type 1 and 4, and a path attribute in EVPN route type 2 and 5.

Example

esi in (ffff.ffff.ffff.ffff.fff0)

etag
An Ethernet tag attribute consists of four octets. An Ethernet tag identifies a particular broadcast domain, for example, a VLAN. An EVPN instance consists of one or more broadcast domains. It is a net attribute in EVPN route type 1, 2, 3 and 5.

Example

etag in (10000)
**mac**

The mac attribute consists of six octets. This attribute is a net attribute in EVPN route type 2.

**Example**

```
mac in (0206.acb1.e806)
```

**evpn-originator**

The evpn-originator attribute specifies the originating router's IP address (4 or 16 octets). This is a net attribute in EVPN route type 3 and 4.

**Example**

```
evpn-originator in (1.2.3.4)
```

**evpn-gateway**

The evpn-gateway attribute specifies the gateway IP address. The gateway IP address is a 32-bit or 128-bit field (IPv4 or IPv6), and encodes an overlay next-hop for the IP prefixes. The gateway IP address field can be zero if it is not used as an overlay next-hop. This is a path attribute in EVPN route type 5.

**Example**

```
evpn-gateway in (1.2.3.4)
```

---

### EVPN RPL Attribute Set

In this context, the term set is used in its mathematical sense to mean an unordered collection of unique elements. The policy language provides sets as a container for groups of values for matching purposes. Sets are used in conditional expressions. The elements of the set are separated by commas. Null (empty) sets are allowed.

**prefix-set**

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The prefix-set specifies one or more IP prefixes.

**Example**

```
prefix-set ip_prefix_set
14.2.0.0/16,
54.0.0.0/16,
12.12.12.0/24,
50:50::1:0/112
end-set
```
mac-set
The mac-set specifies one or more MAC addresses.

Example

```
mac-set mac_address_set
1234.2345.6789,
2345.3456.7890
end-set
```

esi-set
The esi-set specifies one or more ESI's.

Example

```
esi-set evpn_esi_set
1234.2345.3456.4567.5678,
1234.2345.3456.4567.5670
end-set
```

etag-set
The etag-set specifies one or more Ethernet tags.

Example

```
etag-set evpn_etag_set
10000,
20000
end-set
```

Configure EVPN RPL Feature

The following section describe how to configure mac-set, esi-set, evpn-gateway, and evpn-originator.

```c
/* 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
```
Running Configuration

/* Configuring a esi-set and referring 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-af)# remote-as 8
Router(config-bgp-af)# address-family ipv4 unicast
Router(config-bgp-af)# route-policy gateway_demo in
Router(config-bgp-af)# route-policy originator_demo out
Router(config-bgp-af)# commit

Running Configuration

/* Configuring a mac-set and referring it in a route-policy (Attach point - neighbor-in) */
mac-set demo_mac_set
1234.ffff.aaa3,
2323.4444.ffff
end-set
route-policy policy_use_pass_mac_set
if mac in demo_mac_set then
    set med 200
else
    set med 1000
endif
end-policy
!
router bgp 100
address-family 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
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
!
CHAPTER 9

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 139
- EVPN Single-Homing Access Gateway, on page 140
- EVPN Multihoming All-Active, on page 141
- Enable Auto-BGP RT with Manual ESI Configuration, on page 142
- Supported EVPN IRB Scenarios, on page 142
- Distributed Anycast Gateway, on page 142
- VM Mobility Support, on page 145

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 15: 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 CEs.
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.

**Note**

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.

**Note**

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

**Intra-subnet Unicast Data Plane**

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

**Inter-subnet Unicast Data Plane**

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

**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

```bash
/* Configure CEF to prefer RIB prefixes over adjacency prefixes.*/
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 3
RP/0/RSP0/CPU0:router(config-if)# lACP system mac 1.1.1
RP/0/RSP0/CPU0:router(config-if)# exit
RP/0/RSP0/CPU0:router(config)# cef adjacency route override rib
```
/* Configure EVPN L3VRF per DC tenant. */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# vrf irb1
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf)# import route-target 1000:1
RP/0/RSP0/CPU0:router(config-vrf)# export route-target 1000:1
RP/0/RSP0/CPU0:router(config-vrf)# 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-int)# 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)# redistribute connected
RP/0/RSP0/CPU0:router(config-bgp-vrf)# redistribute static
RP/0/RSP0/CPU0:router(config-bgp-vrf)# exit
RP/0/RSP0/CPU0:router(config-bgp-vrf)# redistribute connected
RP/0/RSP0/CPU0:router(config-bgp-vrf)# 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 EVPN IRB

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

Verify the adjacency entries, particularly verify newly added information for synced IPv4 and IP ARP entries.
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: 0x77d7bcb380
Last updated: Feb 28 15:58:21.998
Adjacency producer: arp (prod_id: 10)
Flags: incomplete adj,
Additional Adjacency Information (4 bytes long),
Upto first 4 bytes (in hex): 01000000
Netio idb pointer not cached Cached interface type: 78

Adjacency references:
bfd_agent (JID 150, PID 3637), 0 reference
l2fib_mgr (JID 185, PID 4003), 0 reference
fib_mgr (JID 294, PID 3605), 1 reference
aib (JID 314, PID 3590), 1 reference

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 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  BV2 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  BV2 0000.f65a.3570 10:1:2::91
7  0/0/CPU0  BV2 0000.f65a.357d 10:1:2::93
7  0/0/CPU0  BV2 0000.f65a.3570 fe80::200:f6ff:fe5a:3570

Verify sequence ID for VM mobility.
```plaintext
show l2route evpn mac-ip all detail

<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</td>
</tr>
<tr>
<td></td>
<td>0x06000000</td>
<td>0x22000080</td>
<td></td>
<td></td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

show l2route evpn mac all detail

<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>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>0022.5830.0001</td>
<td>L2VPN</td>
<td>Bundle-Ether5.1300</td>
<td>0</td>
<td>BSSpl</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x06000000</td>
<td>0x25000080</td>
<td></td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Verify duplicate detection and recovery parameters.

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

show run evpn mac

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

configure
```
Configure EVPN IRB

RP/0/RP0/CPU0:EVVPN-LF1(config)# evpn
RP/0/RP0/CPU0:EVVPN-LF1(config-evpn)# mac
RP/0/RP0/CPU0:EVVPN-LF1(config-evpn)# secure
RP/0/RP0/CPU0:EVVPN-LF1(config-evpn-mac)# move-count 1000
RP/0/RP0/CPU0:EVVPN-LF1(config-evpn-mac-secure)# end

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

RP/0/RP0/CPU0:router# show run evpn mac

evpn
mac
  secure
    move-count 1000

Verify the entries to obtain details learnt in L2FIB RP when it is an aggregator. Route processor (RP) entries are aggregated entries obtained from the line cards. In some cases of MAC move, there could be different states for the same MAC. This is displayed in RP aggregated entries. RP determines the update to be sent to L2RIB according to MAC-Learning algorithms.

RP/0/RP0/CPU0:router# show l2vpn mac-learning mac-ipv4 all location 0/RP0/CPU0

<table>
<thead>
<tr>
<th>Topo ID</th>
<th>Producer</th>
<th>Next Hop(s)</th>
<th>Mac Address</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0/0/CPU0</td>
<td>BV1</td>
<td>1000.0001.0001</td>
<td>10.1.1.11</td>
</tr>
<tr>
<td>7</td>
<td>0/0/CPU0</td>
<td>BV2</td>
<td>0000.f65a.3570</td>
<td>10.1.2.91</td>
</tr>
<tr>
<td>7</td>
<td>0/0/CPU0</td>
<td>BV2</td>
<td>0000.f65a.357d</td>
<td>10.1.2.93</td>
</tr>
</tbody>
</table>

Verify the entries in L2RIB that are updated by RP L2FIB. Note the following when you verify the entries:

- The entries with producer as L2VPN and NH as remote IP are learnt from the remote peer gateways, which are learnt from BGP, updated to EVPN, and then updated to L2RIB. So these entries are not from local IP-MAC learning.

- The entries with producer as L2VPN and NH as local bundle interfaces are synced entries from MH-AA peer gateway.

- The entries with producer as LOCAL and NH as local bundle interfaces are dynamically learnt local entries.

RP/0/RP0/CPU0:router# show l2route evpn mac-ip evi 6

<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>10.1.1.101</td>
<td>L2VPN</td>
<td>172.16.0.2/24014/ME</td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.3575</td>
<td>10.1.1.97</td>
<td>L2VPN</td>
<td>172.16.0.7/24025/ME</td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.3575</td>
<td>10.1.1.97</td>
<td>L2VPN</td>
<td>172.16.0.7/24025/ME</td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>fe80::200:f6ff:fe5a:3575</td>
<td>L2VPN</td>
<td>172.16.0.7/24025/ME</td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>fe80::200:f6ff:fe5a:357c</td>
<td>LOCAL</td>
<td>Bundle-Ether1.11</td>
</tr>
<tr>
<td>6</td>
<td>0000.f65a.357c</td>
<td>fe80::200:f6ff:fe5a:357c</td>
<td>LOCAL</td>
<td>Bundle-Ether1.11</td>
</tr>
<tr>
<td>6</td>
<td>0010.0001.0012</td>
<td>10.1.1.12</td>
<td>L2VPN</td>
<td>172.16.0.7/24025/ME</td>
</tr>
<tr>
<td>6</td>
<td>1000.0001.0001</td>
<td>10.1.1.11</td>
<td>LOCAL</td>
<td>Bundle-Ether1.11</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

EVI MAC address IP address Nexthop Label
---- --------------- ---------- ---------- ----- 
1 0000.f65a.357c 10.1.1.93 172.16.0.2 24014
```

Ethernet Tag : 0
Multi-paths Resolved : True
Static : No
Local Ethernet Segment : N/A
Remote Ethernet Segment : 0100.6cbc.a77c.c180.0000
Local Sequence Number : N/A
Remote Sequence Number : 0
Local Encapsulation : N/A
Remote Encapsulation : MPLS

Verify local BGP entries with appropriate second label and second IP VRF route-target.

```
RP/0/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):
    172.16.0.2 (metric 101) from 172.16.0.9 (172.16.0.2)
  Received Label 24014, Second Label 24027
  Second Label 24027 >>> Second label when IRB host-routing is enabled.
  Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate, rib-install
  Received Path ID 0, Local Path ID 0, version 3772
  Extended community: SoO:172.16.0.2:1 RT:100:100
  EVPN ESI: 0100.6cbc.a77c.c180.0000
  Path #2: Received by speaker 0
  Not advertised to any peer
  Local
    172.16.0.2 (metric 101) from 172.16.0.9 (172.16.0.2)
  Received Label 24014, Second Label 24031
  Origin IGP, localpref 100, valid, internal, add-path, import-candidate, imported, rib-install
  Received Path ID 0, Local Path ID 2, version 3769
  Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100 >>> Second RT is IP VRF RT for remote to import into IP VRF routing table.
  Originator: 172.16.0.2, Cluster list: 172.16.0.9
```
EVVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, 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 EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

Verify the remote peer gateway BGP entries with correct label and route-target. Particularly verify the local auto-generated RD on a remote EVPN gateway. EVVPN 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 EVPN gateway, but not in the local EVPN 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/CPU0:router# show bgp l2vpn evpn rd 172.16.0.7:1**
[2][0][48][0000.f65a.357c][128][10:1:1::93]/232

BGP routing table entry for [2][0][48][0000.f65a.357c][128][10:1:1::93]/232, Route Distinguisher: 172.16.0.7:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 6059 6059
Last Modified: Feb 28 12:03:22.448 for 2d23h
Paths: (2 available, best #1)
  Not advertised to any peer
Path #1: Received by speaker 0
  Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24013, Second Label 24029
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported, rib-install
Received Path ID 0, Local Path ID 0, version 6043
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbe.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.

```plaintext
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
```

```plaintext
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.
```
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24033
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported
Received Path ID 0, Local Path ID 1, version 22163
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
Source from L2VPN and from dynamic ARP entry.

RP/0/RSP0/CPU0:router# show bgp vpnv6 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 bRIB/RIB SendTblVer
Speaker 22163 22163
Last Modified: Feb 28 12:09:30.447 for 2d23h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24029
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported
Received Path ID 0, Local Path ID 0, version 22163
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24033
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported
Received Path ID 0, Local Path ID 1, version 22163
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1

Verify local forwarding with local adjacency which overwrite the RIB entries, and remote peer that use the IP VRF host route entries for IP VPN forwarding.

RP/0/RSP0/CPU0:router# show bgp vpnv4 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, distance 200, metric 0, type internal
Installed Feb 28 15:57:28.154 for 2d20h
Routing Descriptor Blocks
The following sections describe how to verify the subnet stretching.

Verify the VRF.
Verify the BGP configuration.

```
RP/0/RP0/CPU0:leafW# show run router bgp | begin vrf cust130
vrf cust130
  rd auto
  address-family ipv4 unicast
    import route-target
    130:130 !
    export route-target
    130:130 !

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 !
```

L2VPN and Ethernet Services Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.2.x
CHAPTER 10

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 161
- EVPN-VPWS Multi-Homed, on page 163

EVPN-VPWS Single Homed

The EVPN-VPWS single homed solution requires per EVI Ethernet Auto Discovery route. EVPN defines a new BGP Network Layer Reachability Information (NLRI) used to carry all EVPN routes. BGP Capabilities Advertisement used to ensure that two speakers support EVPN NLRI (AFI 25, SAFI 70) as per RFC 4760.

The architecture for EVPN VPWS is that the PEs run Multi-Protocol BGP in control-plane. The following image describes the EVPN-VPWS configuration:

- The VPWS service on PE1 requires the following three elements to be specified at configuration time:
  - The VPN ID (EVI)
  - The local AC identifier (AC1) that identifies the local end of the emulated service.
  - The remote AC identifier (AC2) that identifies the remote end of the emulated service.
PE1 allocates a MPLS label per local AC for reachability.

- The VPWS service on PE2 is set in the same manner as PE1. The three same elements are required and the service configuration must be symmetric.

PE2 allocates a MPLS label per local AC for reachability.

- PE1 advertise a single EVPN per EVI Ethernet AD route for each local endpoint (AC) to remote PEs with the associated MPLS label.

PE2 performs the same task.

- On reception of EVPN per EVI EAD route from PE2, PE1 adds the entry to its local L2 RIB. PE1 knows the path list to reach AC2, for example, next hop is PE2 IP address and MPLS label for AC2.

PE2 performs the same task.

Configure EVPN-VPWS Single Homed

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

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# neighbor 10.10.10.1
Router(config-bgp-af)# commit
Router(config-bgp-af)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/1/0/2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 100 target 12 source 10
Router(config-l2vpn-xc-p2p)# commit
Router(config-l2vpn-xc-p2p)# exit
```

Running Configuration

```
configure
router bgp 100
 address-family l2vpn evpn
 neighbor 10.10.10.1
!
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 18: 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 (PE3 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 1_5_6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether10.2
```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
Router(config-evpn-ac-es)# commit

/* Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether10.2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
Router(config-evpn-ac-es)# commit

/* Configure PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether20.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit

/* Configure PE4 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether20.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit
Running Configuration

/* On PE1 */
!
configure
dl2vpn xconnect group evpn_vpws
  p2p e1_5-6
    interface Bundle-Ether10.2
    neighbor evpn evi 1 target 5 source 6
! evpn
interface Bundle-Ether10
  ethernet-segment
    identifier type 0 00.01.00.ac.ce.55.00.0a.00
!

/* On PE2 */
!
configure
dl2vpn xconnect group evpn_vpws
  p2p e1_5-6
    interface Bundle-Ether10.2
    neighbor evpn evi 1 target 5 source 6
! evpn
interface Bundle-Ether10
  ethernet-segment
    identifier type 0 00.01.00.ac.ce.55.00.0a.00
!

/* On PE3 */
!
configure
dl2vpn xconnect group evpn_vpws
  p2p e1_5-6
    interface Bundle-Ether20.1
    neighbor evpn evi 1 target 6 source 5
! evpn
interface Bundle-Ether20
  ethernet-segment
    identifier type 0 00.01.00.ac.ce.55.00.14.00
!

/* On PE4 */
!
configure
dl2vpn xconnect group evpn_vpws
  p2p e1_5-6
    interface Bundle-Ether20.1
    neighbor evpn evi 1 target 6 source 5
! evpn
interface Bundle-Ether20
  ethernet-segment
    identifier type 0 00.01.00.ac.ce.55.00.14.00
!
CHAPTER 11

References

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

- Gigabit Ethernet Protocol Standards, on page 167
- Carrier Ethernet Model References, on page 167
- Default Configuration Values for Gigabit Ethernet and 10-Gigabit Ethernet, on page 169
- References for Configuring Link Bundles, on page 170

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 167.

Carrier Ethernet Model References

This topic covers the references for Gigabit Ethernet Protocol Standards.
IEEE 802.3 Physical Ethernet Infrastructure

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

IEEE 802.3ae 10 Gbps Ethernet

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

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

General Ethernet Standards

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

Ethernet MTU

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

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

- Fragmentation for IPV4 packets—In this process, IPv4 packets are fragmented as necessary to fit within the MTU of the next-hop physical network.

  Note: IPv6 does not support fragmentation.

- MTU discovery process determines largest packet size—This process is available for all IPV6 devices, and for originating IPv4 devices. In this process, the originating IP device determines the size of the largest IPv6 or IPV4 packet that can be sent without being fragmented. The largest packet is equal to the smallest MTU of any network between the IP source and the IP destination devices. If a packet is larger than the smallest MTU of all the networks in its path, that packet will be fragmented as necessary. This process ensures that the originating device does not send an IP packet that is too large.

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

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

Default Configuration Values for Gigabit Ethernet and 10-Gigabit Ethernet

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

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

<table>
<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 (BIA²)</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>
1. The restrictions are applicable to L2 main interface, L2 subinterface, L3 main interface, interflex L2 interface etc.

2. burned-in address

References for Configuring Link Bundles

This section provides references to configuring link bundles. For an overview of link bundles and configurations, see Configure Link Bundles for Layer 2 VPNs, on page 31.

Characteristics of Link Bundles

• Any type of Ethernet interfaces can be bundled, with or without the use of LACP (Link Aggregation Control Protocol).

• Physical layer and link layer configuration are performed on individual member links of a bundle.

• Configuration of network layer protocols and higher layer applications is performed on the bundle itself.

• A bundle can be administratively enabled or disabled.

• Each individual link within a bundle can be administratively enabled or disabled.

• Ethernet link bundles are created in the same way as Etherokinet channels, where the user enters the same configuration on both end systems.

• The MAC address that is set on the bundle becomes the MAC address of the links within that bundle.

• When LACP configured, each link within a bundle can be configured to allow different keepalive periods on different members.

• Load balancing is done by flow instead of by packet. Data is distributed to a link in proportion to the bandwidth of the link in relation to its bundle.

• QoS is supported and is applied proportionally on each bundle member.

• Link layer protocols, such as CDP, work independently on each link within a bundle.

• Upper layer protocols, such as routing updates and hello messages, are sent over any member link of an interface bundle.

• Bundled interfaces are point to point.

• A link must be in the UP state before it can be in distributing state in a bundle.

• Access Control List (ACL) configuration on link bundles is identical to ACL configuration on regular interfaces.
Multicast traffic is load balanced over the members of a bundle. For a given flow, internal processes select the member link and all traffic for that flow is sent over that member.

Methods of Forming Bundles of Ethernet Interfaces

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

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

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

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

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

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

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

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

Note

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

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

Link Aggregation Through LACP

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

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