Virtual Private Network Configuration Guide for Cisco NCS 6000 Series Routers, IOS XR Release 7.4.x

First Published: 2021-07-01

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

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

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

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

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

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

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

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

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

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

© 2021 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface

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

This guide describes the Cisco NCS 6000 Series Router configurations. The preface for the contains these sections:

Changes to This Document

The following table lists the technical changes made to this document since it was first published.

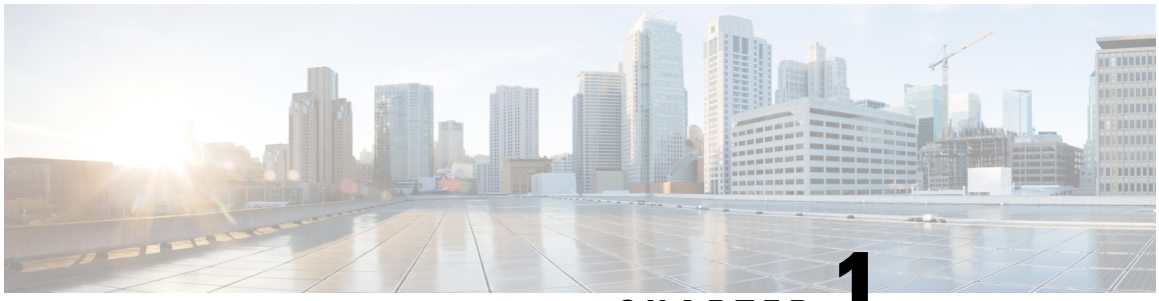
Date	Change Summary
July 2021	Initial release of this document.

Obtaining Documentation and Submitting a Service Request

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

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

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



CHAPTER 1

New and Changed VPN Features

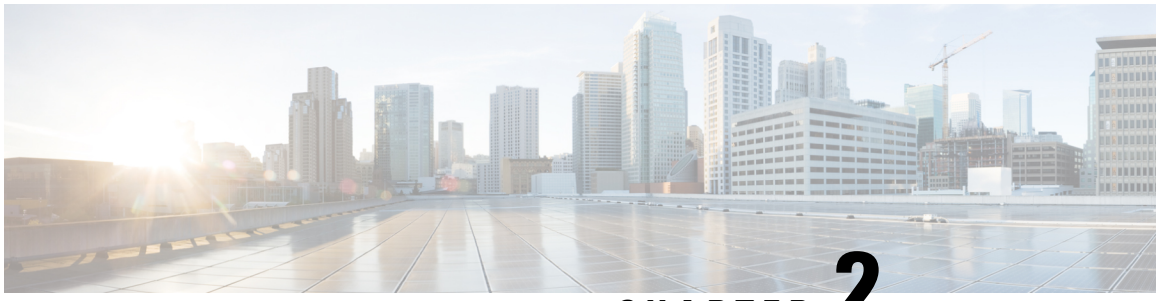
This table summarizes the new and changed feature information for the Virtual Private Network Configuration Guide for Cisco NCS 6000 Series Routers, and tells you where they are documented.

New and Changed VPN Features

Table 1: VPN Features Added or Modified in IOS XR Release 7.4.x

Feature	Description	Changed in Release	Where Documented
PWHE Load Balancing using FAT Label	This feature was introduced.	Release 7.4.1	PWHE Load Balancing using FAT Label, on page 99

New and Changed VPN Features



CHAPTER 2

The Carrier Ethernet Model

This module provides the conceptual information for Implementing Ethernet Flow Points (EFPs).

Release	Modification
Release 5.2.1	This feature was introduced.

Ethernet Flow Point

An Ethernet Flow Point (EFP) is a Layer 2 logical subinterface used to classify traffic under a physical or a bundle interface. A physical interface can be an Ethernet interface and has ports on the line card.

A bundle interface is a virtual interface, created by grouping physical interfaces together. For example, physical interfaces such as 10 Gigabit Ethernet 0/0/0/1 and 10 Gigabit Ethernet 0/0/0/0 can be configured as members of a bundle interface.

Grouping physical interfaces together can:

- Reduce the routing entries.
- Increase the bandwidth of the bundle interface.
- Balance the traffic on the bundle members.

EFP has the following characteristics:

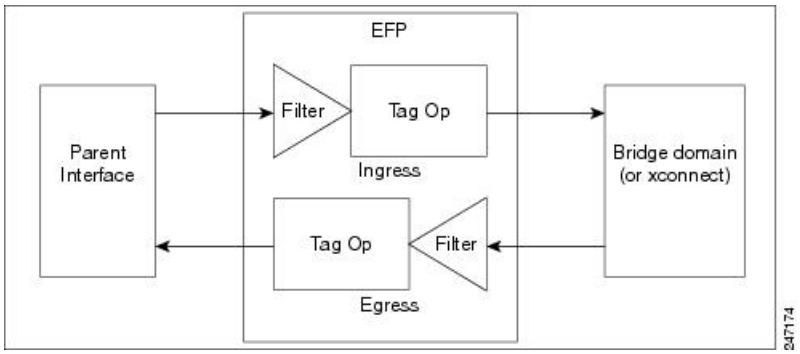
- An EFP represents a logical demarcation point of an Ethernet Virtual Connection (EVC) on an interface. For an EVC associating two or more UNIs, there is a flow point on each interface of every device, through which that EVC passes.
- An EFP can be regarded as an instantiation of a particular service. An EFP is defined by a set of filters. These filters are applied to all the ingress traffic to classify the frames that belong to a particular EFP. An EFP filter is a set of entries, where each entry looks similar to the start of a packet (ignoring source/destination MAC address). Each entry usually contains 0, 1 or 2 VLAN tags. 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.

- An EFP serves four purposes:
  - Identifies all frames that belong to a particular flow on a given interface
  - Performs ingress and egress Ethernet header manipulations
  - Adds features to the identified frames
  - Optionally define how to forward those frames in the data path.

You can perform a variety of operations on the traffic flows when a router is configured with EFPs on various interfaces. Also, you can bridge or tunnel the traffic by many ways from one or more of the router's ingress EFPs to one or more egress EFPs. This traffic is a mixture of VLAN IDs, single or double (QinQ) encapsulation, and ethertypes.

The following figure shows the EFP model.

Figure 1: EFP Model



An EFP subinterface is configured to specify which traffic on ingress is vectored to that EFP. This is done by specifying a VLAN ID or QinQ tagging to match against on ingress. All traffic on ingress is compared to each EFP's matching criterion, and processed by that EFP if a match occurs. The processing performed by an EFP can change VLAN IDs, add or remove VLAN tags, and change ethertypes.

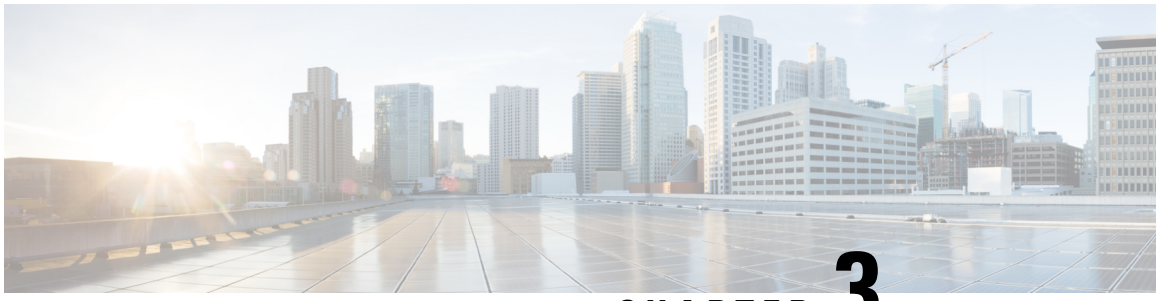
EFP CLI Overview

The following commands are typically used to configure an EFP:

- **l2transport** command - This command identifies a subinterface (or a physical port or bundle-port parent interface) as an EFP.
- **encapsulation** command - This command is used to specify matching criteria.
- **rewrite** command - This command is used to specify the VLAN tag rewrite criteria.

Ethernet CFM

For Ethernet Connectivity Fault Management (CFM) feature, see the *Ethernet OAM* chapter in the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide*.



CHAPTER 3

Configuring Link Bundles

A bundle is a group of one or more ports that are aggregated together and treated as a single link. The different links within a single bundle can have varying speeds, where the fastest link can be a maximum of ten times greater than the slowest link. Each bundle has a single MAC, a single IP address, and a single configuration set (such as ACLs or QoS).

The router supports bundling for these types of interfaces:

- Ethernet interfaces

Note Bundles do not have a one-to-one modular services card association.

Feature History for Configuring Link Bundles

Release	Modification
Release 5.2.1	This feature was introduced.

Prerequisites for Configuring Link Bundles

Before configuring Link Bundling, be sure 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.

- You know the interface IP address.

- You know which links should be included in the bundle you are configuring.
- If you are configuring an Ethernet link bundle, you have at least one of these Ethernet line cards installed in the router:
  - NCS 6000 10x100G Multi-Service CXPv
  - NCS 6000 10x100G Multi-Service CPAK
  - Cisco PANINI 60-ports 10GE SFP Optics LC

Note For more information about physical interfaces, PLIMs, and modular services cards, refer to the *Cisco Network Convergence System 6000 Series Routers Hardware Installation Guide* .

Information About Configuring Link Bundles

To implement the Link Bundling feature, you must understand these concepts:

Link Bundling Overview

A link bundle is simply a group of ports that are bundled together and act as a single link. The advantages of link bundles are these:

- Multiple links can span several line cards to form a single interface. Thus, the failure of a single link does not cause a loss of connectivity.
- Bundled interfaces increase bandwidth availability, because traffic is forwarded over all available members of the bundle. Therefore, traffic can flow on the available links if one of the links within a bundle fails. Bandwidth can be added without interrupting packet flow.

Although the individual links within a single bundle can have varying speeds, all links within a bundle must be of the same type.

Cisco IOS XR software supports these 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.
- 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.

Characteristics of Link Bundles

This list describes the properties and limitations of link bundles:

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

- Bundle membership can span across several line cards that are installed in a single router.
- The Cisco NCS 6000 Series Router supports a maximum of 256 ethernet link bundles. Each link bundle can have a maximum of 64 member links.
- A single bundle supports maximum of 64 physical links.
- Different link speeds are allowed within a single bundle, with a maximum of ten times the speed difference between the members of the bundle.
- 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 Ethernet 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 (the distribution of data between member links) 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 and HDLC keepalives, work independently on each link within a bundle.
- Upper layer protocols, such as routing updates and hellos, 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.
- All links within a single bundle must be configured either to run 802.3ad (LACP) or Etherchannel (non-LACP). Mixed links within a single bundle are not supported.
- A bundle interface can contain physical links only.
- 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.

IEEE 802.3ad Standard

The IEEE 802.3ad standard typically defines a method of forming Ethernet link bundles.

For each link configured as bundle member, this 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 We recommend that you avoid modifying the MAC address, because changes in the MAC address can affect packet forwarding.

VLANs on an Ethernet Link Bundle

802.1Q VLAN subinterfaces can be configured on 802.3ad Ethernet link bundles. Keep this information in mind when adding VLANs on an Ethernet link bundle:

- The maximum number of VLANs allowed per bundle is 4000.
- The maximum number of bundled VLANs allowed per router is 128000.

Note The memory requirement for bundle VLANs is slightly higher than standard physical interfaces.

To create a VLAN subinterface on a bundle, include the VLAN subinterface instance with the **interface Bundle-Ether** command:

interface Bundle-Ether `instance.subinterface`

After you create a VLAN on an Ethernet link bundle, all physical VLAN subinterface configuration is supported on that link bundle.

Link Bundle Configuration Overview

These steps provide a general overview of the link bundle configuration process. Keep in mind that a link must be cleared of all previous network layer configuration before it can be added to a bundle:

1. In global configuration mode, create a link bundle. To create an Ethernet link bundle, enter the **interface Bundle-Ether** command.
2. Assign an IP address and subnet mask to the virtual interface using the **ipv4 address** command.
3. Add interfaces to the bundle you created in Step 1 with the **bundle id** command in the interface configuration submode. You can add up to 32 links to a single bundle.

Note A link is configured to be a member of a bundle from the interface configuration submode for that link.

Nonstop Forwarding During Card Failover

The Cisco IOS XR software supports nonstop forwarding during failover between active and standby paired RSP cards. Nonstop forwarding ensures that there is no change in the state of the link bundles when a failover occurs.

For example, if an active RSP fails, the standby RSP becomes operational. The configuration, node state, and checkpoint data of the failed RSP are replicated to the standby RSP. The bundled interfaces will all be present when the standby RSP becomes the active RSP.

Note Failover is always onto the standby RSP.

Note You do not need to configure anything to guarantee that the standby interface configurations are maintained.

Link Failover

When one member link in a bundle fails, traffic is redirected to the remaining operational member links and traffic flow remains uninterrupted.

Bundle Interfaces: Redundancy, Load Sharing, Aggregation

A bundle is a group of one or more ports that are aggregated together and treated as a single link. The different links within a single bundle can have varying speeds, where the fastest link can be a maximum of four times greater than the slowest link. Each bundle has a single MAC, a single IP address, and a single configuration set (such as ACLs or QoS).

The router supports bundling for these types of interfaces:

- Ethernet interfaces

- VLAN subinterfaces

How to Configure Link Bundling

Configuring Ethernet Link Bundles

This section describes how to configure a Ethernet link bundle.

Note MAC accounting is not supported on Ethernet link bundles.

Note In order for an Ethernet bundle to be active, you must perform the same configuration on both connection endpoints of the bundle.

The creation of an Ethernet link bundle involves creating a bundle and adding member interfaces to that bundle, as shown in the steps that follow.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address-address mask*
4. **bundle minimum-active bandwidth** *kbps* (optional)
5. **bundle minimum-active links** *links* (optional)
6. **bundle maximum-active links** *links* (optional)
7. **bundle maximum-active links** *links* **hot-standby**
8. **exit**
9. **interface TenGigE** *instance*
10. **bundle id** *bundle-id* [**mode** { **active** | **on** | **passive**}]
11. **no shutdown**(optional)
12. **exit**
13. Repeat Step 8 through Step 11 to add more links to the bundle you created in Step 2.
14. Use the **commit** or **end** command.
15. **exit**
16. **exit**
17. Perform Step 1 through Step 15 on the remote end of the connection.
18. **show bundle Bundle-Ether** *bundle-id* [**reasons**] (optional)
19. **show lacp Bundle-Ether** *bundle-id* (optional)

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the XR Config mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface specific configuration commands are entered. Use the **exit** command to exit from the interface configuration submode back to the normal XR Config mode.

Step 3 **ipv4 address** *ipv4-address-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Step 4 **bundle minimum-active bandwidth** *kbps* (optional)

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000
```

Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links* (optional)

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2
```

Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links* (optional)

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1
```

Designates one active link and one link in standby mode that can take over immediately for a bundle if the active link fails (1:1 protection).

The default number of active links allowed in a single bundle is 8.

Note If the **bundle maximum-active** command is issued, then only the highest-priority link within the bundle is active. The priority is based on the value from the **bundle port-priority** command, where a lower value is a higher priority. Therefore, we recommend that you configure a higher priority on the link that you want to be the active link.

Step 7 **bundle maximum-active links** *links* **hot-standby**

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1 hot-standby
```

The **hot-standby** keyword helps to avoid bundle flaps on a switchover or switchback event during which the bundle temporarily falls below the minimum links or bandwidth threshold.

It sets default values for the wait-while timer and suppress-flaps timer to achieve this.

Step 8 **exit**

Example:

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

Exits interface configuration submode for the Ethernet link bundle.

Step 9 **interface TenGigE** *instance*

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 1/0/0/0
```

Enters the interface configuration mode for the specified interface.

Enter the **TenGigE** keyword to specify the interface type. Replace the instance argument with the node-id in the *rack/slot/module* format.

Note Mixed link bundle mode is supported only when active-standby operation is configured (usually with the lower speed link in standby mode).

Step 10 **bundle id** *bundle-id* [**mode** { **active** | **on** | **passive**}]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle-id 3
```

Adds the link to the specified bundle.

To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string.

To add the link to the bundle without LACP support, include the optional **mode on** keywords with the command string.

Note If you do not specify the **mode** keyword, the default mode is **on** (LACP is not run over the port).

Step 11 **no shutdown**(optional)

Example:

```
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

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

Exits interface configuration submode for the Ethernet link bundle.

Step 13 Repeat Step 8 through Step 11 to add more links to the bundle you created in Step 2.

Step 14 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 15 **exit**

Example:

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

Exits interface configuration mode.

Step 16 **exit**

Example:

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

Exits the XR Config mode.

Step 17 Perform Step 1 through Step 15 on the remote end of the connection.

Brings up the other end of the link bundle.

Step 18 **show bundle Bundle-Ether** *bundle-id* [**reasons**] (optional)

Example:

```
RP/0/RP0/CPU0:router# show bundle Bundle-Ether 3 reasons
```

Shows information about the specified Ethernet link bundle

Step 19 **show lacp Bundle-Ether** *bundle-id* (optional)

Example:

```
RP/0/RP0/CPU0:router # show lacp Bundle-Ether 3
```

Shows detailed information about LACP ports and their peers.

Configuring VLAN Bundles

This section describes how to configure a VLAN bundle. The creation of a VLAN bundle involves three main tasks:

1. Create an Ethernet bundle.
2. Create VLAN subinterfaces and assign them to the Ethernet bundle.
3. Assign Ethernet links to the Ethernet bundle.

These tasks are describe in detail in the procedure that follows.

Note In order for a VLAN bundle to be active, you must perform the same configuration on both ends of the bundle connection.

The creation of a VLAN link bundle is described in the steps that follow.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **bundle minimum-active bandwidth** *kbps* (optional)
5. **bundle minimum-active links** *links* (optional)
6. **bundle maximum-active links** *links* (optional)
7. **exit**
8. **interface Bundle-Ether** *bundle-id.vlan-id*
9. **encapsulation dot1q** *vlan-id*
10. **ipv4 address** *ip-address mask*
11. **no shutdown**
12. **exit**
13. Repeat Step 7 through Step 12 to add more VLANs to the bundle you created in Step 2.
14. Use the **commit** or **end** command.
15. **exit**
16. **exit**
17. **show ethernet trunk bundle-Ether** *instance*
18. **configure**
19. **interface TenGigE** *instance*
20. **bundle id** `bundle-id` **[mode {active | on | passive}]**
21. **no shutdown**
22. Repeat Step 19 through Step 21 to add more Ethernet interfaces to the bundle you created in Step 2 .
23. Use the **commit** or **end** command.
24. Perform Step 1 through Step 23 on the remote end of the connection.
25. **show bundle Bundle-Ether** *bundle-id* [**reasons**]
26. **show ethernet trunk bundle-Ether** *instance*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router#(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface specific configuration commands are entered. Use the **exit** command to exit from the interface configuration submode back to the normal XR Config mode

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Step 4 **bundle minimum-active bandwidth** *kbps* (optional)

Example:

```
RP/0/RP0/CPU0:router(config-if) # bundle minimum-active bandwidth 580000
```

(Optional) Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links* (optional)

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2
```

(Optional) Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links* (optional)

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1
```

(Optional) Designates one active link and one link in standby mode that can take over immediately for a bundle if the active link fails (1:1 protection).

Note The default number of active links allowed in a single bundle is 8.

Note If the **bundle maximum-active** command is issued, then only the highest-priority link within the bundle is active. The priority is based on the value from the **bundle port-priority** command, where a lower value is a higher priority. Therefore, we recommend that you configure a higher priority on the link that you want to be the active link.

Step 7 **exit**

Example:

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

Exits interface configuration submode.

Step 8 **interface Bundle-Ether** *bundle-id.vlan-id*

Example:

```
RP/0/RP0/CPU0:router#(config)#interface Bundle-Ether 3.1
```

Creates a new VLAN, and assigns the VLAN to the Ethernet bundle you created in Step 2.

Replace the *bundle-id* argument with the *bundle-id* you created in Step 2.

Replace the *vlan-id* with a subinterface identifier. Range is from 1 to 4094 inclusive (0 and 4095 are reserved).

Note When you include the *vlan-id* argument with the **interface Bundle-Ether** *bundle-id* command, you enter subinterface configuration mode.

Step 9 **encapsulation dot1q** *vlan-id*

Example:

```
RP/0/RP0/CPU0:router#(config-subif)# encapsulation dot1q 10
```

Assigns a VLAN to the subinterface.

Replace the *vlan-id* argument with a subinterface identifier. Range is from 1 to 4094 inclusive (0 and 4095 are reserved).

Step 10 **ipv4 address** *ip-address mask*

Example:

```
RP/0/RP0/CPU0:router#(config-subif)# ipv4 address 10.1.2.3/24
```

Assigns an IP address and subnet mask to the subinterface.

Step 11 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-subif) # no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

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

Exits subinterface configuration mode for the VLAN subinterface.

Step 13 Repeat Step 7 through Step 12 to add more VLANs to the bundle you created in Step 2.

(Optional) Adds more subinterfaces to the bundle.

Step 14 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 15 **exit**

Example:

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

Exits interface configuration mode.

Step 16 **exit**

Example:

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

Exits XR Config mode.

Step 17 **show ethernet trunk bundle-Ether** *instance*

Example:

```
RP/0/RP0/CPU0:routerrouter# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Step 18 **configure**

Example:

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

Enters XR Config mode.

Step 19 **interface TenGigE** *instance*

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 1/0/0/0
```

Enters the interface configuration mode for the specified interface.

Replace the *instance* argument with the node-id in the *rack/slot/module* format.

Note A VLAN bundle is not active until you add an Ethernet interface on both ends of the link bundle.

Step 20 **bundle id `bundle-id` [mode {active | on | passive}]**

Example:

```
RP/0/RP0/CPU0:router(config-if)#  bundle-id 3
```

Adds an Ethernet interface to the bundle you configured in Step 2 through Step 13.

To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string.

To add the interface to the bundle without LACP support, include the optional **mode on** keywords with the command string.

Note If you do not specify the **mode** keyword, the default mode is **on** (LACP is not run over the port).

Step 21 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 22 Repeat Step 19 through Step 21 to add more Ethernet interfaces to the bundle you created in Step 2 .

Step 23 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Step 24 Perform Step 1 through Step 23 on the remote end of the connection.

Brings up the other end of the link bundle.

Step 25 **show bundle Bundle-Ether** *bundle-id* [**reasons**]

Example:

```
RP/0/RP0/CPU0:router#show bundle Bundle-Ether 3 reasons
```

(Optional) Shows information about the specified Ethernet link bundle.

The **show bundle Bundle-Ether** command displays information about the specified bundle. If your bundle has been configured properly and is carrying traffic, the State field in the **show bundle Bundle-Ether** command output will show the number “4,” which means the specified VLAN bundle port is “distributing.”

Step 26 **show ethernet trunk bundle-Ether** *instance*

Example:

```
RP/0/RP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Configuration Examples for Link Bundles

EtherChannel Bundle running LACP: Example

This example shows how to join two ports to form an EtherChannel bundle running LACP:

```
RP/0/RSP0/CPU0:Router# config
RP/0/RSP0/CPU0:Router(config)# interface Bundle-Ether 3
RP/0/RSP0/CPU0:Router(config-if)# ipv4 address 1.2.3.4/24
RP/0/RSP0/CPU0:Router(config-if)# bundle minimum-active bandwidth 620000
RP/0/RSP0/CPU0:Router(config-if)# bundle minimum-active links 1
RP/0/RSP0/CPU0:Router(config-if)# exit
RP/0/RSP0/CPU0:Router(config)# interface TenGigE 0/3/0/0
RP/0/RSP0/CPU0:Router(config-if)# bundle id 3 mode active
RP/0/RSP0/CPU0:Router(config-if)# no shutdown
RP/0/RSP0/CPU0:Router(config)# exit
RP/0/RSP0/CPU0:Router(config)# interface TenGigE 0/3/0/1
RP/0/RSP0/CPU0:Router(config-if)# bundle id 3 mode active
RP/0/RSP0/CPU0:Router(config-if)# no shutdown
RP/0/RSP0/CPU0:Router(config-if)# exit
```

Creating VLANs on a Ethernet Bundle: Example

This example shows how to create and bring up two VLANs on an Ethernet bundle:

```
RP/0/RSP0/CPU0:Router# config
RP/0/RSP0/CPU0:Router(config)# interface Bundle-Ether 1
RP/0/RSP0/CPU0:Router(config-if)# ipv4 address 1.2.3.4/24
RP/0/RSP0/CPU0:Router(config-if)# bundle minimum-active bandwidth 620000
RP/0/RSP0/CPU0:Router(config-if)# bundle minimum-active links 1
RP/0/RSP0/CPU0:Router(config-if)# exit
RP/0/RSP0/CPU0:Router(config)# interface Bundle-Ether 1.1
RP/0/RSP0/CPU0:Router(config-subif)# encapsulation dot1q 10
RP/0/RSP0/CPU0:Router(config-subif)# ip addr 10.2.3.4/24
RP/0/RSP0/CPU0:Router(config-subif)# no shutdown
RP/0/RSP0/CPU0:Router(config-subif)# exit
RP/0/RSP0/CPU0:Router(config)# interface Bundle-Ether 1.2
RP/0/RSP0/CPU0:Router(config-subif)# encapsulation dot1q 20
RP/0/RSP0/CPU0:Router(config-subif)# ip addr 20.2.3.4/24
RP/0/RSP0/CPU0:Router(config-subif)# no shutdown
RP/0/RSP0/CPU0:Router(config-subif)# exit
RP/0/RSP0/CPU0:Router(config)#interface tengige 0/1/5/7
RP/0/RSP0/CPU0:Router(config-if)# bundle-id 1 mode act
RP/0/RSP0/CPU0:Router(config-if)# commit
RP/0/RSP0/CPU0:Router(config-if)# exit
RP/0/RSP0/CPU0:Router(config)# exit
RP/0/RSP0/CPU0:Router # show ethernet trunk bundle-ether 1
```

NCS 6000 Link Bundles connected to a Cisco 7600 EtherChannel: Example

This example is an end-to-end example of a bundle between NCS 6000 Series router and a Cisco 7600 Series Router (P19_C7609-S) in the Metro Ethernet network that supports both L2 and L3 services.

On the Cisco NCS 6000 Series Routers, the bundle is configured with LACP, 1:1 link protection, two L2 subinterfaces, and two layer 3 subinterfaces.

IOS XR side:

```
hostname PE44_IOS-XR_Router

interface Bundle-Ether16
 description Connect to P19_C7609-S Port-Ch 16
 mtu 9216
 no ipv4 address
 bundle maximum-active links 1
!
interface Bundle-Ether16.160 l2transport
 description Connect to P19_C7609-S Port-Ch 16 EFP 160
 encapsulation dot1q 160
!
interface Bundle-Ether16.161 l2transport
 description Connect to P19_C7609-S Port-Ch 16 EFP 161
 encapsulation dot1q 161
!
interface Bundle-Ether16.162
 description Connect to P19_C7609-S Port-Ch 16.162
 ipv4 address 10.194.8.44 255.255.255.0
 encapsulation dot1q 162
!
interface Bundle-Ether16.163
 description Connect to P19_C7609-S Port-Ch 16.163
 ipv4 address 10.194.12.44 255.255.255.0
 encapsulation dot1q 163
!


interface TenGigE 0/1/0/16
 description Connected to P19_C7609-S GE 8/0/16
 bundle id 16 mode active
 bundle port-priority 1
!
interface TenGigE 0/1/0/17
 description Connected to P19_C7609-S GE 8/0/17
 bundle id 16 mode active
 bundle port-priority 2
!
```

IOS XR side - connections to CE devices:

```
hostname PE44_IOS-XR_Router

interface TenGigE 0/1/0/3.160 l2transport
 description VLAN 160 over BE 16.160
 encapsulation dot1q 100 second-dot1q 160
 rewrite ingress tag pop 1 symmetric
!
interface TenGigE 0/1/0/3.161 l2transport
 description VLAN 161 over BE 16.161
 encapsulation dot1q 161
!
l2vpn
 !
 xconnect group 160
  p2p 160
   interface Bundle-Ether16.160
   interface TenGigE 0/1/0/3.160
```

```
   description VLAN_160_over_BE_16.160
  !
!
 xconnect group 161
  p2p 161
   interface Bundle-Ether16.161
   interface TenGigE 0/1/0/3.161
   description VLAN_161_over_BE_16.161
  !
 !
```

IOS XR side - CE devices:

```
hostname PE64_C3750-ME
!
vlan 161
!
interface TenGigE 1/0/1
 description Connected to PE65_ME-C3400 GE 0/1
 switchport access vlan 100
 switchport mode dot1q-tunnel
!
interface TenGigE 1/0/2
 description Connected to PE44_IOS-XR_Router GE 0/1/0/3
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 100,161
 switchport mode trunk
!
interface Vlan161
 description VLAN 161 over BE 16.161 on PE44
 ip address 161.0.0.64 255.255.255.0
!



hostname PE65_ME-C3400
!
vlan 160
!
interface TenGigE 0/1
 description Connected to PE64_C3750-ME GE 1/0/1
 port-type nni
 switchport trunk allowed vlan 160
 switchport mode trunk
!
interface Vlan160
 description VLAN 160 over BE 16.160 on PE44
 ip address 160.0.0.65 255.255.255.0
!
```

IOS side:

```
hostname P19_C7609-S

port-channel load-balance src-dst-port
!
interface Port-channel16
 description Connected to PE44_IOS-XR_Router BE 16
 mtu 9202
 no ip address
 logging event link-status
 logging event status
 speed nonegotiate
```

```
 mls qos trust dscp
 lacp fast-switchover
 lacp max-bundle 1
 service instance 160 ethernet
  description Connected to PE44_IOS-XR_Router BE 16.160
  encapsulation dot1q 160
 !
 service instance 161 ethernet
  description Connected to PE44_IOS-XR_Router BE 16.161
  encapsulation dot1q 161
 !
!
interface Port-channel16.162
 description Connected to PE44_IOS-XR_Router BE 16.162
 encapsulation dot1Q 162
 ip address 10.194.8.19 255.255.255.0
!
interface Port-channel16.163
 description Connected to PE44_IOS-XR_Router BE 16.163
 encapsulation dot1Q 163
 ip address 10.194.12.19 255.255.255.0
!

interface TenGigE 8/0/16
 no shut
 description Connected to PE44_IOS-XR_Router GE 0/1/0/16
 mtu 9202
 no ip address
 logging event link-status
 logging event status
 speed nonegotiate
 no mls qos trust dscp
 lacp port-priority 1
 channel-protocol lacp
 channel-group 16 mode active
!
interface TenGigE 8/0/17
 no shut
 description Connected to PE44_IOS-XR_Router GE 0/1/0/17
 mtu 9202
 no ip address
 logging event link-status
 logging event status
 speed nonegotiate
 no mls qos trust dscp
 lacp port-priority 2
 channel-protocol lacp
 channel-group 16 mode active
!
```

IOS side - connections to CE devices:

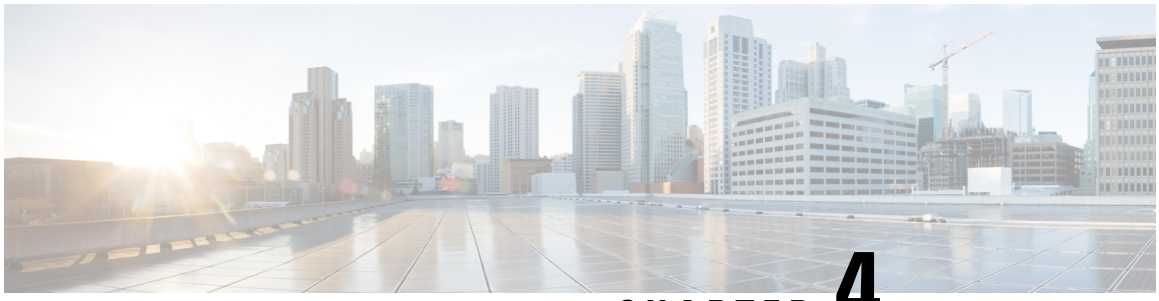
```
hostname P19_C7609-S

interface TenGigE 8/0/7
 description Connected to PE62_C3750-ME GE 1/0/2
 mtu 9000
 no ip address
 speed nonegotiate
 mls qos trust dscp
 service instance 160 ethernet
  description VLAN 160 over Port-Ch 16
  encapsulation dot1q 100 second-dot1q 160
  rewrite ingress tag pop 1 symmetric
```

```
 !
 service instance 161 ethernet
  description VLAN 161 over Port-Ch 16
  encapsulation dot1q 161
 !
!
connect eline-161 Port-channel16 161 TenGigE 8/0/7 161
 !
!
connect eline-160 Port-channel16 160 TenGigE 8/0/7 160
 !
!
```

IOS side - CE devices:

```
hostname PE62_C3750-ME
!
vlan 161
!
interface TenGigE 1/0/1
 description Connected to PE63_ME-C3400 GE 0/1
 switchport access vlan 100
 switchport mode dot1q-tunnel
!
interface TenGigE 1/0/2
 description Connected to P19_C7609-S GE 8/0/7
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 100,161
 switchport mode trunk
!
interface Vlan161
 description VLAN 161 over Port-Chan 16 on P19
 ip address 161.0.0.62 255.255.255.0
!
```

```
hostname PE63_ME-C3400
!
vlan 160
!
interface TenGigE 0/1
 description Connected to PE62_C3750-ME GE 1/0/1
 port-type nni
 switchport trunk allowed vlan 160
 switchport mode trunk
!
interface Vlan160
 description VLAN 160 over Port-Chan 16 on P19
 ip address 160.0.0.63 255.255.255.0
!
```



CHAPTER 4

Implementing Point to Point Layer 2 services

This module provides the conceptual and configuration information for Point to Point Layer 2 services on Cisco IOS XR software.

Note The Point to Point Layer 2 services are also called as MPLS Layer 2 VPNs.

Feature History for Implementing Point to Point Layer 2 services Configuration Module

Release	Modification
Release 5.2.1	The point to point layer 2 services were introduced.
Release 5.2.3	Support was added for QinQ mode, Inter-AS mode, and GTP load balancing.

Prerequisites for Implementing Point to Point Layer 2 Services

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.

Information About Implementing Point to Point Layer 2 Services

To implement Point to Point Layer 2 Services, you should understand these concepts:

L2VPN Overview

Layer 2 VPN (L2VPN) emulates the behavior of a LAN across an IP or MPLS-enabled IP network allowing Ethernet devices to communicate with each other as they would when connected to a common LAN segment.

As Internet service providers (ISPs) look to replace their Asynchronous Transfer Mode (ATM) infrastructures with an IP infrastructure, there is a need for to provide standard methods of using an IP infrastructure to provide a serviceable L2 interface to customers; specifically, to provide standard ways of using an IP infrastructure to provide virtual circuits between pairs of customer sites.

Building a L2VPN system requires coordination between the ISP and the customer. The ISP provides L2 connectivity; the customer builds a network using data link resources obtained from the ISP. In an L2VPN service, the ISP does not require information about a the customer's network topology, policies, routing information, point-to-point links, or network point-to-point links from other ISPs.

The ISP 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.
- Emulation of L2 quality-of-service (QoS) over a packet switch network.
- Ease of configuration of the L2 service.
- Support for different types of tunneling mechanisms (MPLS, L2TPv3, IPSec, GRE, and others).
- L2VPN process databases include all information related to circuits and their connections.

Virtual Circuit Connection Verification on L2VPN

Virtual Circuit Connection Verification (VCCV) is an L2VPN Operations, Administration, and Maintenance (OAM) feature that allows network operators to run IP-based provider edge-to-provider edge (PE-to-PE) keepalive protocol across a specified pseudowire to ensure that the pseudowire data path forwarding does not contain any faults. The disposition PE receives VCCV packets on a control channel, which is associated with the specified pseudowire. The control channel type and connectivity verification type, which are used for VCCV, are negotiated when the pseudowire is established between the PEs for each direction.

Two types of packets can arrive at the disposition egress:

- Type 1—Specifies normal Ethernet-over-MPLS (EoMPLS) data packets.
- Type 2—Specifies VCCV packets.

Cisco NCS 6000 Series Router supports Label Switched Path (LSP) VCCV Type 1, which uses an inband control word if enabled during signaling. The VCCV echo reply is sent as IPv4 that is the reply mode in IPv4. The reply is forwarded as IP, MPLS, or a combination of both.

VCCV pings counters that are counted in MPLS forwarding on the egress side. However, on the ingress side, they are sourced by the route processor and do not count as MPLS forwarding counters.

Ethernet over MPLS

Ethernet-over-MPLS (EoMPLS) provides a tunneling mechanism for Ethernet traffic through an MPLS-enabled L3 core and encapsulates Ethernet protocol data units (PDUs) inside MPLS packets (using label stacking) to forward them across the MPLS network.

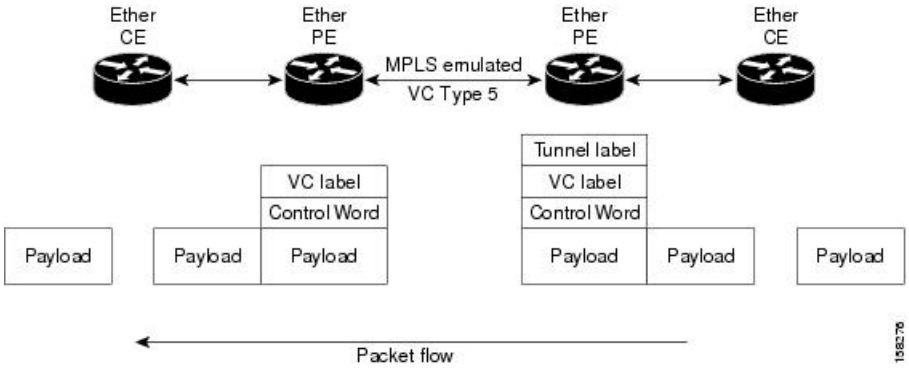
EoMPLS features are described in these subsections:

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.

The following figure provides an example of Ethernet port mode.

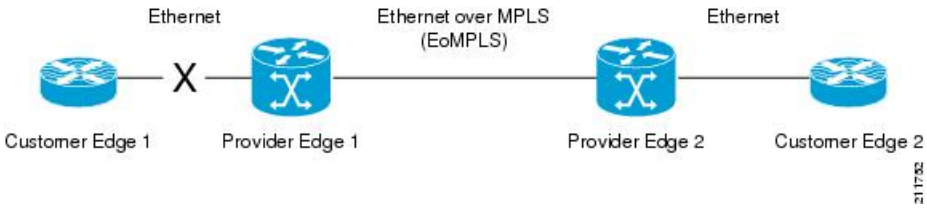
Figure 2: Ethernet Port Mode Packet Flow



Ethernet Remote Port Shutdown

Ethernet remote port shutdown provides a mechanism for the detection and propagation of remote link failure for port mode EoMPLS on a Cisco NCS 6000 Series Router line card. This lets a service provider edge router on the local end of an Ethernet-over-MPLS (EoMPLS) pseudowire detect a cross-connect or remote link failure and cause the shutdown of the Ethernet port on the local customer edge router. Shutting down the Ethernet port on the local customer edge router prevents or mitigates a condition where that router would otherwise lose data by forwarding traffic continuously to the remote failed link, especially if the link were configured as a static IP route (see following figure)..

Figure 3: Remote Link Outage in EoMPLS Wide Area Network



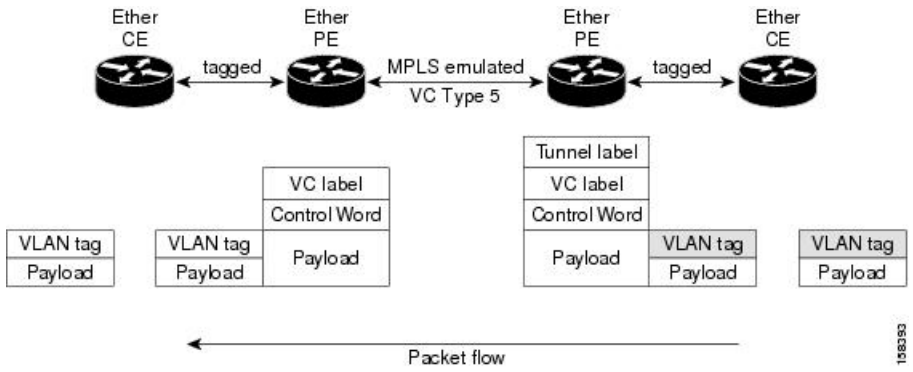
To enable this functionality, see the **l2transport propagate** command in *MPLS Command Reference for Cisco NCS 6000 Series Routers*.

VLAN Mode

In VLAN mode, each VLAN on a customer-end to provider-end link can be configured as a separate L2VPN connection using virtual connection (VC) type 4 or VC type 5. VC type 5 is the default mode.

As illustrated in the following figure, the Ethernet PE associates an internal VLAN-tag to the Ethernet port for switching the traffic internally from the ingress port to the pseudowire; however, before moving traffic into the pseudowire, it removes the internal VLAN tag.

Figure 4: 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 5: 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.

The Cisco NCS 6000 Series Routers implement the Layer 2 tunneling or Layer 3 forwarding depending on the subinterface configuration at provider edge routers. This function only supports up to two QinQ tags on the SPA and fixed PLIM:

- Layer 2 QinQ VLANs in L2VPN attachment circuit: QinQ L2VPN attachment circuits are configured under the Layer 2 transport subinterfaces for point-to-point EoMPLS based cross-connects using virtual circuit type 4 pseudowires and point-to-point local-switching-based cross-connects including full interworking 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:

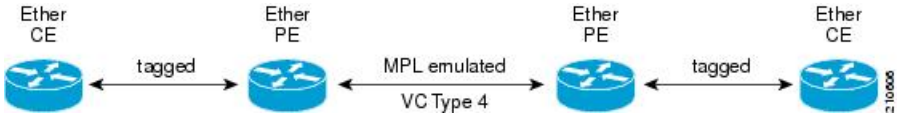
- IPv4 unicast and multicast
- IPv6 unicast and multicast
- MPLS

Note The Cisco NCS 6000 Series Router does not support: bundle attachment circuits and Hot Standby Router Protocol (HSRP) or Virtual Router Redundancy Protocol (VRRP) on QinQ subinterfaces.

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 6: EoMPLS over QinQ Mode



High Availability

L2VPN uses control planes in both route processors and line cards, as well as forwarding plane elements in the line cards.

Note The l2tp_mgr process does not support high availability.

The availability of L2VPN meets these requirements:

- A control plane failure in either the route processor or the line card will not affect the circuit forwarding path.
- The router processor control plane supports failover without affecting the line card control and forwarding planes.
- L2VPN integrates with existing Label Distribution Protocol (LDP) graceful restart mechanism.

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). Using preferred tunnel path, it is always assumed that the traffic engineering tunnel that transports the L2 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).

Note

- Currently, preferred tunnel path configuration applies only to MPLS encapsulation.
- The fallback enable option is supported.

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.

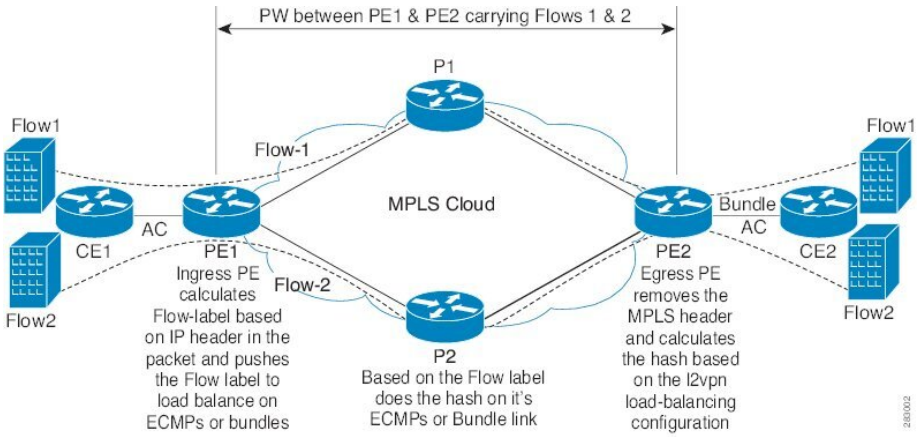
Note Pseudowire redundancy is provided only for point-to-point Virtual Private Wire Service (VPWS) pseudowires.

Flow Aware Transport Pseudowire (FAT PW)

Flow Aware Transport Pseudowires (FAT PW) are used to load-balance traffic in the core when equal cost multipaths (ECMP) are used. The flow, in this context, refers to a sequence of packets that have the same source and destination pair. The packets are transported from a source provider edge (PE) to a destination PE.

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

Figure 7: FAT PW with two flows distributing over ECMPs and Bundle-Links



The MPLS labels add an additional label to the stack, called the flow label, which contains the flow information of a virtual circuit (VC). A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from the IP payload of a packet. The flow label contains the end of label stack (EOS) bit set and inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The FAT PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are taken based on that label.

Core routers perform load balancing using the flow-label in the FAT PW with other information like MAC address and IP address. The flow-label adds greater entropy to improve traffic load balancing. Therefore, it's possible to distribute flows over ECMPs and link bundles.

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

L2VPN Nonstop Routing

The L2VPN Nonstop Routing (NSR) feature avoids label distribution path (LDP) sessions from flapping on events such as process failures (crash) and route processor failover (RP FO). NSR on process failure (crash) is supported by performing RP FO, if you have enabled NSR using NSR process failure switchover.

NSR enables the router (where failure has occurred) to maintain the control plane states without a graceful restart (GR). NSR, by definition, does not require any protocol extension and typically uses Stateful Switch Over (SSO) to maintain it's control plane states.

Note NSR is enabled by default for L2VPN on Cisco IOS XR 64 bit operating system. You cannot configure the **nsr** command under L2VPN configuration submode.

How to Implement Point to Point Layer 2 Services

This section describes the tasks required to implement Point to Point Layer 2 Services:

Configuring an Interface or Connection for Point to Point Layer 2 Services

Perform this task to configure an interface or a connection for Point to Point Layer 2 Services.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **l2transport**
4. **exit**
5. **interface** *type interface-path-id.subinterface* **l2transport**
6. **encapsulation dot1q** *vlan-id*
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 **interface** *type interface-path-id*

Example:

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

Enters interface configuration mode and configures an interface.

Step 3 **l2transport**

Example:

```
RP/0/RP0/CPU0:router(config-if)# l2transport
```

Enables L2 transport on the selected interface.

Step 4 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-if-l2)# exit
```

Exits the current configuration mode.

Step 5 **interface** *type interface-path-id.subinterface* **l2transport**

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/0.1 l2transport
```

Enters subinterface configuration mode and configures the subinterface as a layer 2 interface.

Step 6 **encapsulation dot1q** *vlan-id*

Example:

```
RP/0/RP0/CPU0:router(config-if)# encapsulation dot1q vln1
```

Assigns native VLAN ID to an interface trunking 802.1Q VLAN traffic.

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 Static Point-to-Point Cross-Connects

Note Consider this information about cross-connects when you configure static point-to-point cross-connects:

- An cross-connect is uniquely identified with the pair; the cross-connect name must be unique within a group.
- A segment (an attachment circuit or pseudowire) is unique and can belong only to a single cross-connect.
- A static VC local label is globally unique and can be used in one pseudowire only.
- No more than 16,000 cross-connects can be configured per router.

Note Static pseudowire connections do not use LDP for signaling.

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface** *type interface-path-id*
6. **neighbor** *ip-address* **pw-id** *pseudowire-id*
7. **mpls static label local** { *value* } **remote** { *value* }
8. Use the **commit** or **end** command.
9. *show l2vpn xconnect group group name*

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 **xconnect group** *group-name*

Example:

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

Enters the name of the cross-connect group.

Step 4 **p2p** *xconnect-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

Enters a name for the point-to-point cross-connect.

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

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# interface TenGigE 0/0/0/0.1
```

Specifies the interface type and instance.

Step 6 **neighbor** *ip-address* **pw-id** *pseudowire-id*

Example:

```
RP/0/RP0/CPU0:routerr(config-l2vpn-xc-p2p)# neighbor 2.2.2.2 pw-id 2000
```

Configures the pseudowire segment for the cross-connect.

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN.

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

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# mpls static label local 699 remote 890
```

Configures local and remote label ID values.

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 xconnect group group name*

Example:

```
RP/0/RP0/CPU0:show l2vpn xconnect group vlan_grp_1
```

Displays the name of the Point-to-Point cross-connect group you created.

Configuring Dynamic Point-to-Point Cross-Connects

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

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interworking ipv4**
6. **interface** *type interface-path-id*
7. **neighbor** *ip-address* **pw-id** *pseudowire-id*
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
```

Enters L2VPN configuration mode.

Step 3 **xconnect group** *group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

Step 4 **p2p** *xconnect-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

Enters a name for the point-to-point cross-connect.

Step 5 **interworking ipv4**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc)# interworking ipv4
```

Configure the interworking for IPv4.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/0.1
```

Specifies the interface type ID. The choices are:

- GigabitEthernet: GigabitEthernet/IEEE 802.3 interfaces.
- TenGigE: TenGigabitEthernet/IEEE 802.3 interfaces.
- CEM: Circuit Emulation interface

Step 7 **neighbor** *ip-address* **pw-id** *pseudowire-id*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 2.2.2.2 pw-id 2000
```

Configures the pseudowire segment for the cross-connect.

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN.

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

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Inter-AS

The Inter-AS configuration procedure is identical to the L2VPN cross-connect configuration tasks (see "Configuring Static Point-to-Point Cross-Connects" section and "Configuring Dynamic Point-to-Point Cross-Connects" section) except that the remote PE IP address used by the cross-connect configuration is now reachable through iBGP peering.

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

Configuring Preferred Tunnel Path

This procedure describes how to configure a preferred tunnel path.

Note The tunnel used for the preferred path configuration is an MPLS Traffic Engineering (MPLS-TE) tunnel.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** {*name*}
4. **encapsulation mpls**
5. **preferred-path** {**interface**} {**tunnel-ip** *value* | **tunnel-te** *value* | **tunnel-tp** *value*} [**fallback disable**]
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the configuration mode.

Step 2 **l2vpn**

Example:

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

Enters L2VPN configuration mode.

Step 3 **pw-class** {*name*}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# pw-class path1
```

Configures the pseudowire class name.

Step 4 **encapsulation mpls**

Example:

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

Configures the pseudowire encapsulation to MPLS.

Step 5 **preferred-path** {**interface**} {**tunnel-ip** *value* | **tunnel-te** *value* | **tunnel-tp** *value*} [**fallback disable**]

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-mpls)# preferred-path interface tunnel-te 11 fallback
disable
```

Configures preferred path tunnel settings. If the fallback disable configuration is used and once the TE/TP tunnel is configured as the preferred path goes down, the corresponding pseudowire can also go down.

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.

Enabling Load Balancing with ECMP and FAT PW

Perform this task to enable load balancing with ECMP and FAT PW.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *{name}*
4. **encapsulation mpls**
5. **load-balancing flow-label both**
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
```

Enters the L2VPN configuration mode.

Step 3 **pw-class** *{name}*

Example:

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

Configures the pseudowire class name.

Step 4 **encapsulation mpls**

Example:

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

Configures the pseudowire encapsulation to MPLS.

Step 5 **load-balancing flow-label both**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-
mpls)# load-balancing flow-label both
```

Enables load-balancing on ECMPs. Also, enables the imposition and disposition of flow labels for the pseudowire.

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.

Configuring L2VPN Nonstop Routing

Perform this task to configure L2VPN Nonstop Routing.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **nsr**
4. **logging nsr**
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **l2vpn**

Example:

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

Enters the XR Config mode.

Step 3 **nsr**

Example:

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

Enables L2VPN nonstop routing.

Step 4 **logging nsr**

Example:

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

Enables logging of NSR events.

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.

Configure MPLS LDP Nonstop Routing

Perform this task to enable Label Distribution Protocol (LDP) Nonstop Routing (NSR) for synchronizing label information between active and standby LDPs. From Release 6.1.1 onwards, with the introduction of stateful LDP feature, you must explicitly configure LDP NSR to synchronize label information between active and standby LDPs.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **nsr**
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **mpls ldp**

Example:

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

Enters MPLS LDP configuration mode.

Step 3 **nsr**

Example:

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

Enables LDP nonstop routing.

Step 4 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 Flow Aware Transport Pseudowire

This section provides information on

Enabling Load Balancing with ECMP and FAT PW for VPWS

Perform this task to enable load balancing with ECMP and FAT PW for VPWS. Creating a PW-Class in L2VPN configuration leads to load-balancing.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** { *name* }
4. **encapsulation mpls**
5. **load-balancing flow-label** { **both** | **code** | **receive** | **transmit** } [**static**]
6. **exit**
7. **exit**
8. **xconnect group** *group-name*
9. **p2p** *xconnect-name*
10. **interface type** *interface-path-id*
11. **neighbor** *A.B.C.D* **pw-id** pseudowire-id
12. **pw-class** *class-name*
13. 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 **pw-class** { *name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class path1
```

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

Step 4 **encapsulation mpls**

Example:

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

Configures the pseudowire encapsulation to MPLS.

Step 5 **load-balancing flow-label** { **both** | **code** | **receive** | **transmit** } [**static**]

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-pwc-mpls)# load-balancing flow-label both
```

Enables load-balancing on ECMPs. Also, enables the imposition and disposition of flow labels for the pseudowire.

Note If the static keyword is not specified, end to end negotiation of the FAT PW is enabled.

Step 6 **exit**

Example:

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

Exits the pseudowire encapsulation submode and returns the router to the parent configuration mode.

Step 7 **exit**

Example:

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

Exits the pseudowire submode and returns the router to the l2vpn configuration mode.

Step 8 **xconnect group** *group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group grp1
RP/0/RP0/CPU0:router(config-l2vpn-xc)#
```

Specifies the name of the cross-connect group.

Step 9 **p2p** *xconnect-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)#
```

Specifies the name of the point-to-point cross-connect.

Step 10 **interface type** *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# interface TenGigE 0/0/0/0.1
```

Specifies the interface type and instance.

Step 11 **neighbor** *A.B.C.D* **pw-id** pseudowire-id

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.2.2.2 pw-id 2000
```

Configures the pseudowire segment for the cross-connect.

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

Note A.B.C.D can be a recursive or non-recursive prefix.

Step 12 **pw-class** *class-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class path1
```

Associates the pseudowire class with this pseudowire.

Step 13 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 Point to Point Layer 2 Services

In the following example, two traffic classes are created and their match criteria are defined. For the first traffic class called class1, ACL 101 is used as the match criterion. For the second traffic class called class2, ACL 102 is used as the match criterion. Packets are checked against the contents of these ACLs to determine if they belong to the class.

This section includes the following configuration examples:

L2VPN Interface Configuration: Example

This example shows how to configure an L2VPN interface:

```
configure
 interface TenGigE 0/0/0/0.1 l2transport
 encapsulation dot1q 1
 end
```

Point-to-Point Cross-connect Configuration: Examples

This section includes configuration examples for both static and dynamic p2p cross-connects.

Static Configuration

This example shows how to configure a static point-to-point cross-connect:

```
configure
 l2vpn
  xconnect group vlan_grp_1
   p2p vlan1
   interworking ipv4
   interface TenGigE 0/0/0/0.1
   neighbor 2.2.2.2 pw-id 2000
    mpls static label local 699 remote 890
    commit
```

Dynamic Configuration

This example shows how to configure a dynamic point-to-point cross-connect:

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

Inter-AS: Example

This example shows how to set up an AC to AC cross-connect from AC1 to AC2:

Preferred Path: Example

This example shows how to configure preferred tunnel path:

```
configure
l2vpn
pw-class path1
encapsulation mpls
preferred-path interface tunnel-ip value  fallback disable
```

Enabling Load Balancing with FAT PW: Example

This sample configuration shows how to enable load balancing with FAT PW for VPWS.

```
l2vpn
pw-class class1
    encapsulation mpls
        load-balancing flow-label transmit
    !
 !
pw-class class2
    encapsulation mpls
        load-balancing flow-label both
```

```
    !

xconnect group group1
    p2p p1
        interface TenGigE 0/0/0/0.1
        neighbor 10.0.0.1 pw-id 1
            pw-class class1
        !
    !
!
```

Configuring L2VPN Nonstop Routing: Example

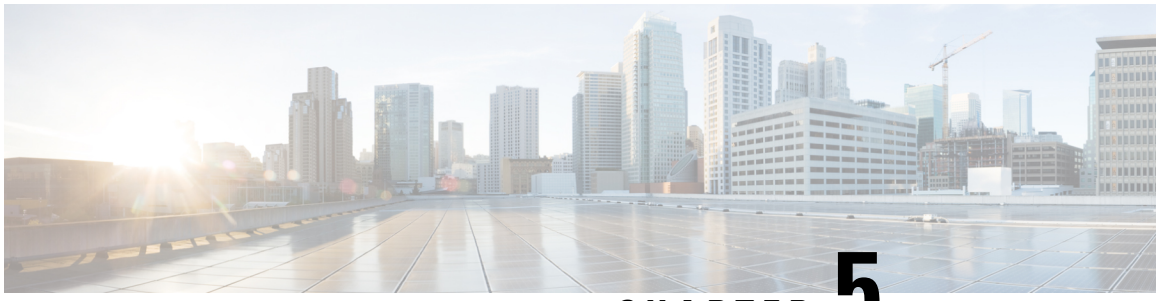
This example shows how to configure L2VPN Nonstop Routing.

```
config
l2vpn
 nsr
 logging nsr
```

Configuring Flow Aware Transport Pseudowire: Example

This sample configuration shows how to enable load balancing with FAT PW for VPWS.

```
l2vpn
pw-class class1
    encapsulation mpls
        load-balancing flow-label transmit
    !
 !
pw-class class2
    encapsulation mpls
        load-balancing flow-label both
    !

xconnect group group1
    p2p p1
        interface TenGigE 0/0/0/0.1
        neighbor 1.1.1.1 pw-id 1
            pw-class class1
        !
    !
!
```



CHAPTER 5

Implementing IPv6 VPN Provider Edge Transport over MPLS

IPv6 Provider Edge or IPv6 VPN Provider Edge (6PE/VPE) uses the existing MPLS IPv4 core infrastructure for IPv6 transport. 6PE/VPE enables IPv6 sites to communicate with each other over an MPLS IPv4 core network using MPLS label switched paths (LSPs).

This feature relies heavily on multiprotocol Border Gateway Protocol (BGP) extensions in the IPv4 network configuration on the provider edge (PE) router to exchange IPv6 reachability information (in addition to an MPLS label) for each IPv6 address prefix. Edge routers are configured as dual-stack, running both IPv4 and IPv6, and use the IPv4 mapped IPv6 address for IPv6 prefix reachability exchange.

For detailed information about the commands used to configure 6PE/VPE, see the *Cisco IOS XR Virtual Private Network Command Reference for Cisco NCS 6000 Series Routers* .

Feature History for Implementing 6PE/VPE Transport over MPLS

Release	Modification
Release 5.2.1	The IPv6 Provider Edge (6PE) feature was introduced.
Release 5.2.3	Support was added for IPv6 VPN Provider Edge (6VPE).

Prerequisites for Implementing 6PE/VPE

The following prerequisites are required to implement 6PE/VPE:

- 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.

- Familiarity with MPLS and BGP4 configuration and troubleshooting.

Information About 6PE/VPE

To configure the 6PE/VPE feature, you should understand the concepts that are described in these sections:

Overview of 6PE/VPE

Multiple techniques are available to integrate IPv6 services over service provider core backbones:

- Dedicated IPv6 network running over various data link layers
- Dual-stack IPv4-IPv6 backbone
- Existing MPLS backbone leverage

These solutions are deployed on service providers' backbones when the amount of IPv6 traffic and the revenue generated are in line with the necessary investments and the agreed-upon risks. Conditions are favorable for the introduction of native IPv6 services, from the edge, in a scalable way, without any IPv6 addressing restrictions and without putting a well-controlled IPv4 backbone in jeopardy. Backbone stability is essential for service providers that have recently stabilized their IPv4 infrastructure.

Service providers running an MPLS/IPv4 infrastructure follow similar trends because several integration scenarios that offer IPv6 services on an MPLS network are possible. Cisco Systems has specially developed Cisco 6PE or IPv6 Provider Edge Router over MPLS, to meet all those requirements.

Inter-AS support for 6PE requires support of Border Gateway Protocol (BGP) to enable the address families and to allocate and distribute PE and ASBR labels.

Note Cisco IOS XR displays actual IPv4 next-hop addresses for IPv6 labeled-unicast and VPNv6 prefixes. IPv4-mapped-to-IPv6 format is not supported.

Benefits of 6PE/VPE

Service providers who currently deploy MPLS experience these benefits of Cisco 6PE/VPE:

- Minimal operational cost and risk—No impact on existing IPv4 and MPLS services.
- Provider edge routers upgrade only—A 6PE/VPE router can be an existing PE router or a new one dedicated to IPv6 traffic.
- No impact on IPv6 customer edge routers—The ISP can connect to any customer CE running Static, IGP or EGP.
- Production services ready—An ISP can delegate IPv6 prefixes.
- IPv6 introduction into an existing MPLS service—6PE/VPE routers can be added at any time.

IPv6 on the Provider Edge and Customer Edge Routers

Service Provider Edge Routers

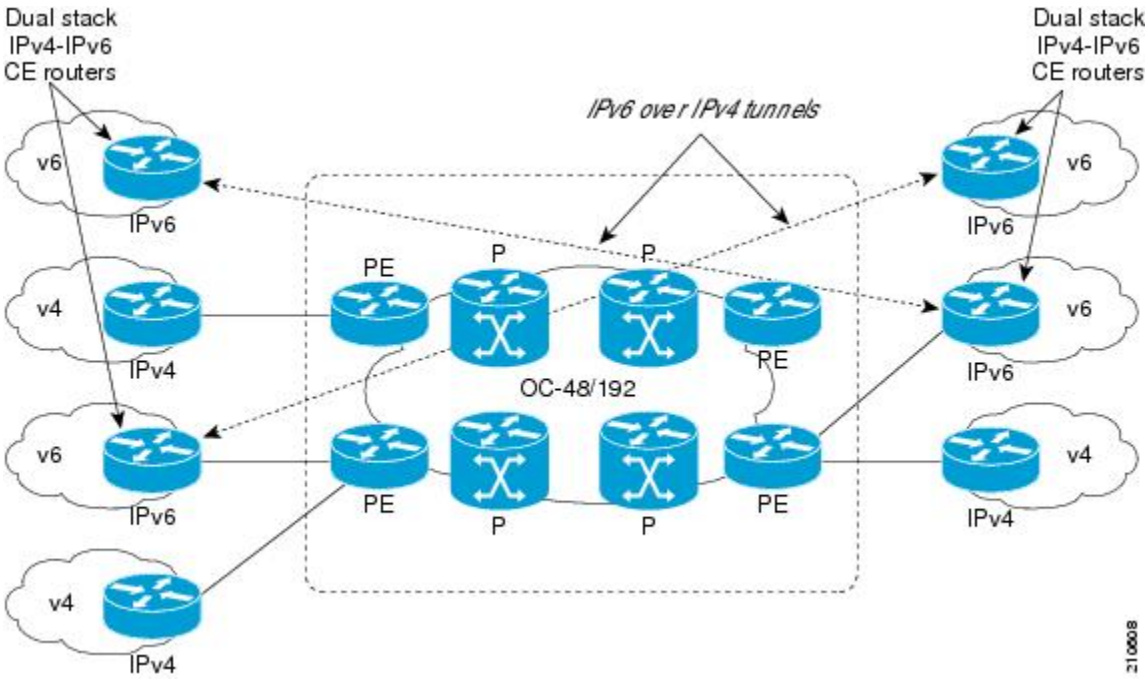
6PE is particularly applicable to service providers who currently run an MPLS network. One of its advantages is that there is no need to upgrade the hardware, software, or configuration of the core network, and it eliminates the impact on the operations and the revenues generated by the existing IPv4 traffic. MPLS is used by many service providers to deliver services to customers. MPLS as a multiservice infrastructure technology is able to provide layer 3 VPN, QoS, traffic engineering, fast re-routing and integration of ATM and IP switching.

Customer Edge Routers

Using tunnels on the CE routers is the simplest way to deploy IPv6 over MPLS networks. It has no impact on the operation or infrastructure of MPLS and requires no changes to the P routers in the core or to the PE routers. However, tunnel meshing is required as the number of CEs to connect increases, and it is difficult to delegate a global IPv6 prefix for an ISP.

The following figure illustrates the network architecture using tunnels on the CE routers.

Figure 8: IPv6 Using Tunnels on the CE Routers



IPv6 Provider Edge Multipath

Internal and external BGP multipath for IPv6 allows the IPv6 router to load balance between several paths (for example, same neighboring autonomous system (AS) or sub-AS, or the same metric) to reach its destination. The 6PE multipath feature uses multiprotocol internal BGP (MP-IBGP) to distribute IPv6 routes over the MPLS IPv4 core network and to attach an MPLS label to each route.

When MP-IBGP multipath is enabled on the 6PE router, all labeled paths are installed in the forwarding table with MPLS information (label stack) when MPLS information is available. This functionality enables 6PE to perform load balancing.

How to Implement 6PE/VPE

This section includes these implementation procedures:

Configuring 6PE/VPE

This task describes how to configure 6PE/VPE on PE routers to transport the IPv6 prefixes across the IPv4 cloud.

Ensure that you configure 6PE/VPE on PE routers participating in both the IPv4 cloud and IPv6 clouds.

Note For 6PE, you can use all routing protocols supported on Cisco IOS XR software such as BGP, OSPF, IS-IS, EIGRP, RIP, and Static to learn routes from both clouds.

BGP uses the **per-vrf** label mode for transporting local and redistributed IP prefixes. Before IOS XR Release 7.5.3, BGP assigned a random label for the prefixes. Starting from Release 7.5.3, BGP assigns a label value of **2**, the IPv6 Explicit NULL Label, for the same prefixes.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family ipv6 labeled-unicast**
6. **exit**
7. **exit**
8. **address-family ipv6 unicast**
9. **allocate-label** [**all** | **route-policy** *policy_name*]
10. 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 **router bgp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 1
```

Enters the number that identifies the autonomous system (AS) in which the router resides.

Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.

Step 3 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
```

Enters neighbor configuration mode for configuring Border Gateway Protocol (BGP) routing sessions.

Step 4 **remote-as** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 100
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 5 **address-family ipv6 labeled-unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 labeled-unicast
```

Specifies IPv6 labeled-unicast address prefixes.

Note This option is also available in IPv6 neighbor configuration mode and VRF neighbor configuration mode.

Step 6 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit
```

Exits BGP address-family submode.

Step 7 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# exit
```

Exits BGP neighbor submode.

Step 8 **address-family ipv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv6 unicast
```

Specifies IPv6 unicast address prefixes.

Step 9 **allocate-label** [**all** | **route-policy** *policy_name*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-af)# allocate-label all
```

Allocates MPLS labels for specified IPv4 unicast routes.

Note The **route-policy** keyword provides finer control to filter out certain routes from being advertised to the neighbor.

Step 10 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 PE to PE Core

This task describes how to configure a Provider Edge (PE) to PE Core.

For information on configuring VPN Routing and Forwarding (VRF), refer to the *Implementing BGP* module of the *Routing Configuration Guide for Cisco NCS 6000 Series Routers*.

SUMMARY STEPS

1. **configure**
2. **router bgp**
3. **address-family vpnv6 unicast**
4. **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*]
5. **bgp client-to-client reflection { cluster-id | disable }**
6. **neighbor** *ip-address*
7. **remote-as** *as-number*
8. **description** *text*
9. **password** { **clear** | **encrypted** } *password*
10. **shutdown**
11. **timers** *keepalive hold-time*
12. **update-source type** *interface-id*
13. **address-family vpnv6 unicast**
14. **route-policy** *route-policy-name* { **in** | **out** }
15. **exit**
16. **vrf** *vrf-name*
17. **rd** { *as-number : nn* | *ip-address : nn* | **auto** }
18. 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 **router bgp**

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 10
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family vpnv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv6 unicast
```

Specifies the vpnv6 address family and enters address family configuration submode.

Step 4 **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-af)# bgp dampening 30 1500 10000 120
```

Configures BGP dampening for the specified address family.

Step 5 **bgp client-to-client reflection { cluster-id | disable }**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-af)# bgp client-to-client
reflection disable
```

Configures client to client route reflection.

Step 6 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 7 **remote-as** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 100
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 8 **description** *text*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# description neighbor 172.16.1.1
```

Provides a description of the neighbor. The description is used to save comments and does not affect software function.

Step 9 **password** { **clear** | **encrypted** } *password*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# password encrypted 123abc
```

Enables Message Digest 5 (MD5) authentication on the TCP connection between the two BGP neighbors.

Step 10 **shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# router bgp 1
```

Terminates any active sessions for the specified neighbor and removes all associated routing information.

Step 11 **timers** *keepalive hold-time*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# timers 12000 200
```

Set the timers for the BGP neighbor.

Step 12 **update-source type** *interface-id*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source TenGigE 0/1/5/0
```

Allows iBGP sessions to use the primary IP address from a specific interface as the local address when forming an iBGP session with a neighbor.

Step 13 **address-family vpnv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpvn6 unicast
```

Enters VPN neighbor address family configuration mode.

Step 14 **route-policy** *route-policy-name* { **in** | **out** }

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pe-pe-vpn-out out
```

Specifies a routing policy for an outbound route. The policy can be used to filter routes or modify route attributes.

Step 15 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit
```

Exits address family configuration and neighbor submode.

Step 16 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf-pe
```

Configures a VRF instance.

Step 17 **rd** { *as-number : nn* | *ip-address : nn* | **auto** }

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# rd 345:567
```

Configures the route distinguisher.

Use the auto keyword if you want the router to automatically assign a unique RD to the VRF.

Step 18 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.

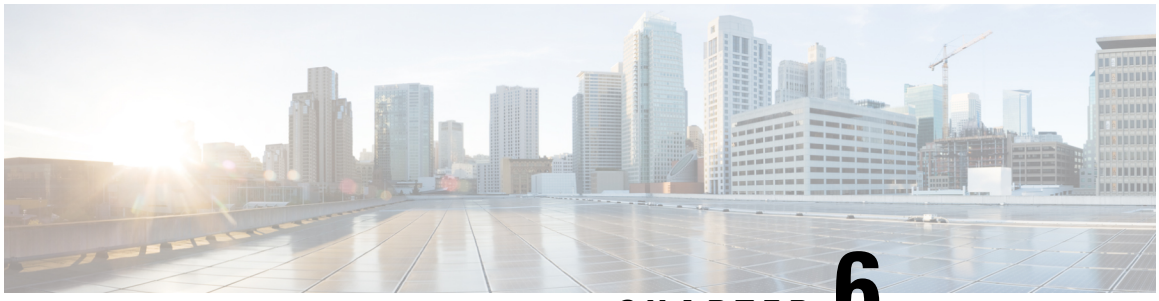
Configuration Examples for 6PE/VPE

This section includes the following configuration example:

Configuring 6PE on a PE Router: Example

This sample configuration shows the configuration of 6PE on a PE router:

```
interface TenGigE0/3/0/0
 ipv6 address 2001::1/64
!
router isis ipv6-cloud
 net 49.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
 interface TenGigE0/3/0/0
  address-family ipv6 unicast
  !
!
router bgp 55400
 bgp router-id 54.6.1.1
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
  network 55:5::/64
  redistribute connected
  redistribute isis ipv6-cloud
   allocate-label all

 !
 neighbor 34.4.3.3
  remote-as 55400
  address-family ipv4 unicast
  !
  address-family ipv6 labeled-unicast
```



CHAPTER 6

Implementing MPLS Layer 3 VPNs

A Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN) consists of a set of sites that are interconnected by means of an MPLS provider core network. At each customer site, one or more customer edge (CE) routers attach to one or more provider edge (PE) routers.

This module provides the conceptual and configuration information for MPLS Layer 3 VPNs on Cisco IOS XR software.

Prerequisites for Implementing MPLS L3VPN

The following prerequisites are required to configure MPLS Layer 3 VPN:

- To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide.*
- If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You must be in a user group associated with a task group that includes the proper task IDs for:
  - BGP commands
  - MPLS commands (generally)
  - MPLS Layer 3 VPN commands
- To configure MPLS Layer 3 VPNs, routers must support MPLS forwarding and Forwarding Information Base (FIB).

MPLS L3VPN Restrictions

The following restrictions apply when configuring MPLS VPN Inter-AS with ASBRs exchanging IPv4 routes and MPLS labels:

- For networks configured with eBGP multihop, a label switched path (LSP) must be configured between non adjacent routers.
- Inter-AS supports IPv4 routes only. IPv6 is not supported.

Note The physical interfaces that connect the BGP speakers must support FIB and MPLS.

Information About MPLS Layer 3 VPNs

To implement MPLS Layer 3 VPNs, you need to understand the following concepts:

MPLS L3VPN Overview

Before defining an MPLS VPN, VPN in general must be defined. A VPN is:

- An IP-based network delivering private network services over a public infrastructure
- A set of sites that are allowed to communicate with each other privately over the Internet or other public or private networks

Conventional VPNs are created by configuring a full mesh of tunnels or permanent virtual circuits (PVCs) to all sites in a VPN. This type of VPN is not easy to maintain or expand, as adding a new site requires changing each edge device in the VPN.

MPLS-based VPNs are created in Layer 3 and are based on the peer model. The peer model enables the service provider and the customer to exchange Layer 3 routing information. The service provider relays the data between the customer sites without customer involvement.

MPLS VPNs are easier to manage and expand than conventional VPNs. When a new site is added to an MPLS VPN, only the edge router of the service provider that provides services to the customer site needs to be updated.

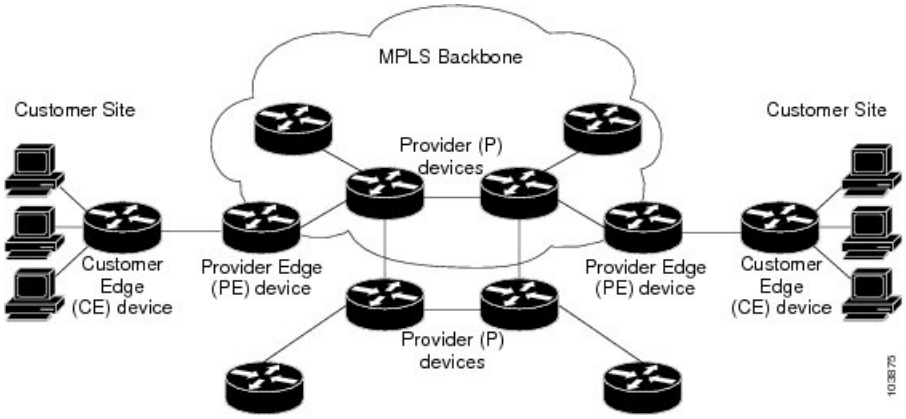
The components of the MPLS VPN are described as follows:

- Provider (P) router—Router in the core of the provider network. P routers run MPLS switching and do not attach VPN labels to routed packets. VPN labels are used to direct data packets to the correct private network or customer edge router.
- PE router—Router that attaches the VPN label to incoming packets based on the interface or subinterface on which they are received, and also attaches the MPLS core labels. A PE router attaches directly to a CE router.
- Customer (C) router—Router in the Internet service provider (ISP) or enterprise network.

- Customer edge (CE) router—Edge router on the network of the ISP that connects to the PE router on the network. A CE router must interface with a PE router.

This following figures shows a basic MPLS VPN topology.

Figure 9: Basic MPLS VPN Topology



MPLS L3VPN Benefits

MPLS L3VPN provides the following benefits:

- Service providers can deploy scalable VPNs and deliver value-added services.
- Connectionless service guarantees that no prior action is necessary to establish communication between hosts.
- Centralized Service: Building VPNs in Layer 3 permits delivery of targeted services to a group of users represented by a VPN.
- Scalability: Create scalable VPNs using connection-oriented, point-to-point overlays, Frame Relay, or ATM virtual connections.
- Security: Security is provided at the edge of a provider network (ensuring that packets received from a customer are placed on the correct VPN) and in the backbone.
- Integrated Quality of Service (QoS) support: QoS provides the ability to address predictable performance and policy implementation and support for multiple levels of service in an MPLS VPN.
- Straightforward Migration: Service providers can deploy VPN services using a straightforward migration path.
- Migration for the end customer is simplified. There is no requirement to support MPLS on the CE router and no modifications are required for a customer intranet.

How MPLS L3VPN Works

MPLS VPN functionality is enabled at the edge of an MPLS network. The PE router performs the following tasks:

- Exchanges routing updates with the CE router

- Translates the CE routing information into VPN version 4 (VPNv4) routes.
- Exchanges VPNv4 and VPNv6 routes with other PE routers through the Multiprotocol Border Gateway Protocol (MP-BGP)

Virtual Routing and Forwarding Tables

Each VPN is associated with one or more VPN routing and forwarding (VRF) instances. A VRF defines the VPN membership of a customer site attached to a PE router. A VRF consists of the following components:

- An IP version 4 (IPv4) unicast routing table
- A derived FIB table
- A set of interfaces that use the forwarding table
- A set of rules and routing protocol parameters that control the information that is included in the routing table

These components are collectively called a VRF instance.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A site can be a member of multiple VPNs. However, a site can associate with only one VRF. A VRF contains all the routes available to the site from the VPNs of which it is a member.

Packet forwarding information is stored in the IP routing table and the FIB table for each VRF. A separate set of routing and FIB tables is maintained for each VRF. These tables prevent information from being forwarded outside a VPN and also prevent packets that are outside a VPN from being forwarded to a router within the VPN.

VPN Routing Information: Distribution

The distribution of VPN routing information is controlled through the use of VPN route target communities, implemented by BGP extended communities. VPN routing information is distributed as follows:

- When a VPN route that is learned from a CE router is injected into a BGP, a list of VPN route target extended community attributes is associated with it. Typically, the list of route target community extended values is set from an export list of route targets associated with the VRF from which the route was learned.
- An import list of route target extended communities is associated with each VRF. The import list defines route target extended community attributes that a route must have for the route to be imported into the VRF. For example, if the import list for a particular VRF includes route target extended communities A, B, and C, then any VPN route that carries any of those route target extended communities—A, B, or C—is imported into the VRF.

BGP Distribution of VPN Routing Information

A PE router can learn an IP prefix from the following sources:

- A CE router by static configuration
- An eBGP session with the CE router
- A Routing Information Protocol (RIP) exchange with the CE router

- Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and RIP as Interior Gateway Protocols (IGPs)

The IP prefix is a member of the IPv4 address family. After the PE router learns the IP prefix, the PE converts it into the VPN-IPv4 prefix by combining it with a 64-bit route distinguisher. The generated prefix is a member of the VPN-IPv4 address family. It uniquely identifies the customer address, even if the customer site is using globally nonunique (unregistered private) IP addresses. The route distinguisher used to generate the VPN-IPv4 prefix is specified by the **rd** command associated with the VRF on the PE router.

BGP distributes reachability information for VPN-IPv4 prefixes for each VPN. BGP communication takes place at two levels:

- Within the IP domain, known as an autonomous system.
- Between autonomous systems.

PE to PE or PE to route reflector (RR) sessions are iBGP sessions, and PE to CE sessions are eBGP sessions. PE to CE eBGP sessions can be directly or indirectly connected (eBGP multihop).

BGP propagates reachability information for VPN-IPv4 prefixes among PE routers by the BGP protocol extensions (see RFC 2283, Multiprotocol Extensions for BGP-4), which define support for address families other than IPv4. Using the extensions ensures that the routes for a given VPN are learned only by other members of that VPN, enabling members of the VPN to communicate with each other.

MPLS Forwarding

Based on routing information stored in the VRF IP routing table and the VRF FIB table, packets are forwarded to their destination using MPLS.

A PE router binds a label to each customer prefix learned from a CE router and includes the label in the network reachability information for the prefix that it advertises to other PE routers. When a PE router forwards a packet received from a CE router across the provider network, it labels the packet with the label learned from the destination PE router. When the destination PE router receives the labeled packet, it pops the label and uses it to direct the packet to the correct CE router. Label forwarding across the provider backbone is based on either dynamic label switching or traffic engineered paths. A customer data packet carries two levels of labels when traversing the backbone:

- The top label directs the packet to the correct PE router.
- The second label indicates how that PE router should forward the packet to the CE router.

More labels can be stacked if other features are enabled. For example, if traffic engineering (TE) tunnels with fast reroute (FRR) are enabled, the total number of labels imposed in the PE is four (Layer 3 VPN, Label Distribution Protocol (LDP), TE, and FRR).

Automatic Route Distinguisher Assignment

To take advantage of iBGP load balancing, every network VRF must be assigned a unique route distinguisher. VRF requires a route distinguisher for BGP to distinguish between potentially identical prefixes received from different VPNs.

With thousands of routers in a network each supporting multiple VRFs, configuration and management of route distinguishers across the network can present a problem. Cisco IOS XR software simplifies this process by assigning unique route distinguisher to VRFs using the **rd auto** command.

To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across theVRFs.

Finally, route distinguisher values are checkpointed so that route distinguisher assignment to VRF is persistent across failover or process restart. If an route distinguisher is explicitely configured for a VRF, this value is not overridden by the autoroute distinguisher.

MPLS L3VPN Major Components

An MPLS-based VPN network has three major components:

- VPN route target communities—A VPN route target community is a list of all members of a VPN community. VPN route targets need to be configured for each VPN community member.
- Multiprotocol BGP (MP-BGP) peering of the VPN community PE routers—MP-BGP propagates VRF reachability information to all members of a VPN community. MP-BGP peering needs to be configured in all PE routers within a VPN community.
- MPLS forwarding—MPLS transports all traffic between all VPN community members across a VPN service-provider network.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A given site can be a member of multiple VPNs. However, a site can associate with only one VRF. A customer-site VRF contains all the routes available to the site from the VPNs of which it is a member

How to Implement MPLS Layer 3 VPNs

This section contains instructions for the following tasks:

Configuring the Core Network

Configuring the core network includes the following tasks:

Assessing the Needs of MPLS VPN Customers

Before configuring an MPLS VPN, the core network topology must be identified so that it can best serve MPLS VPN customers. Perform this task to identify the core network topology.

SUMMARY STEPS

1. Identify the size of the network.
2. Identify the routing protocols in the core.
3. Determine if MPLS High Availability support is required.
4. Determine if BGP load sharing and redundant paths are required.

DETAILED STEPS

Step 1 Identify the size of the network.

Identify the following to determine the number of routers and ports required:

- How many customers will be supported?
- How many VPNs are required for each customer?
- How many virtual routing and forwarding (VRF) instances are there for each VPN?

Step 2 Identify the routing protocols in the core.

Determine which routing protocols are required in the core network.

Step 3 Determine if MPLS High Availability support is required.

MPLS VPN nonstop forwarding and graceful restart are supported on select routers and Cisco IOS XR software releases.

Step 4 Determine if BGP load sharing and redundant paths are required.

Determine if BGP load sharing and redundant paths in the MPLS VPN core are required.

Configuring Routing Protocols in the Core

To configure a routing protocol, see the .

Configuring MPLS in the Core

To enable MPLS on all routers in the core, you must configure a Label Distribution Protocol (LDP). You can use either of the following as an LDP:

- MPLS LDP—See the *Implementing MPLS Label Distribution Protocol* chapter in the *MPLS Configuration Guide for Cisco NCS 6000 Series Routers* for configuration information.
- MPLS Traffic Engineering Resource Reservation Protocol (RSVP)—See module in the *MPLS Configuration Guide for Cisco NCS 6000 Series Routers* for configuration information.

Determining if FIB Is Enabled in the Core

Forwarding Information Base (FIB) must be enabled on all routers in the core, including the provider edge (PE) routers. For information on how to determine if FIB is enabled, see the *Implementing Cisco Express Forwarding* module in the *IP Addresses and Services Configuration Guide for Cisco NCS 6000 Series Routers*.

Configuring Multiprotocol BGP on the PE Routers and Route Reflectors

Perform this task to configure multiprotocol BGP (MP-BGP) connectivity on the PE routers and route reflectors.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*

3. **address-family vpnv4 unicast** or **address-family vpnv6 unicast**
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **address-family vpnv4 unicast** or **address-family vpnv6 unicast**
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 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family vpnv4 unicast** or **address-family vpnv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
```

Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

Step 4 **neighbor** *ip-address* **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24 remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 5 **address-family vpnv4 unicast** or **address-family vpnv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
```

Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

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.

Connecting MPLS VPN Customers

To connect MPLS VPN customers to the VPN, perform the following tasks:

Defining VRFs on the PE Routers to Enable Customer Connectivity

Perform this task to define VPN routing and forwarding (VRF) instances.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family ipv4 unicast**
4. **import route-policy** *policy-name*
5. **import route-target** [*as-number:nn* | *ip-address:nn*]
6. **export route-policy** *policy-name*
7. **export route-target** [*as-number:nn* | *ip-address:nn*]
8. **exit**
9. **exit**
10. **router bgp** *autonomous-system-number*
11. **vrf** *vrf-name*
12. **rd** { *as-number* | *ip-address* | **auto** }
13. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 4 **import route-policy** *policy-name*

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# import route-policy policy_A
```

Specifies a route policy that can be imported into the local VPN.

Step 5 **import route-target** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 120:1
```

Allows exported VPN routes to be imported into the VPN if one of the route targets of the exported route matches one of the local VPN import route targets.

Step 6 **export route-policy** *policy-name*

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# export route-policy policy_B
```

Specifies a route policy that can be exported from the local VPN.

Step 7 **export route-target** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# export route-target 120:2
```

Associates the local VPN with a route target. When the route is advertised to other provider edge (PE) routers, the export route target is sent along with the route as an extended community.

Step 8 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# exit
```

Exits VRF address family configuration mode and returns the router to VRF configuration mode.

Step 9 **exit**

Example:

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

Exits VRF configuration mode and returns the router to XR Config mode.

Step 10 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 11 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode for BGP routing.

Step 12 **rd** { *as-number* | *ip-address* | **auto** }

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# rd auto
```

Automatically assigns a unique route distinguisher (RD) to vrf_1.

Step 13 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 VRF Interfaces on PE Routers for Each VPN Customer

Perform this task to associate a VPN routing and forwarding (VRF) instance with an interface or a subinterface on the PE routers.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *ipv4-address mask*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/3/0/0
```

Enters interface configuration mode.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-if)# vrf vrf_A
```

Configures a VRF instance and enters VRF configuration mode.

Step 4 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0
```

Configures a primary IPv4 address for the specified interface.

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.

Configuring BGP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure PE-to-CE routing sessions using BGP.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*

3. **bgp router-id** {*ip-address*}
4. **vrf** *vrf-name*
5. **address-family ipv4 unicast**
6. **label mode per-ce**
7. Do one of the following:
   - **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
8. **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
9. **network** {*ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]
10. **exit**
11. **neighbor** *ip-address*
12. **remote-as** *autonomous-system-number*
13. **password** { **clear** | **encrypted** } *password*
14. **ebgp-multihop** [*ttl-value*]
15. **address-family ipv4 unicast**
16. **allowas-in** [*as-occurrence-number*]
17. **route-policy** *route-policy-name* **in**
18. **route-policy** *route-policy-name* **out**
19. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **bgp router-id** {*ip-address*}

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24
```

Configures the local router with a router ID of 192.168.70.24.

Step 4 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for BGP routing.

Step 5 **address-family ipv4 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 6 **label mode per-ce**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# label mode per-ce
```

Sets the MPLS VPN label allocation mode for each customer edge (CE) label mode allowing the provider edge (PE) router to allocate one label for every immediate next-hop.

Step 7 Do one of the following:

- **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# redistribute connected
```

Causes routes to be redistributed into BGP. The routes that can be redistributed into BGP are:

- Connected
- Intermediate System-to-Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Static

Step 8 **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# aggregate-address 10.0.0.0/8 as-set
```

Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized.

- The **as-set** keyword generates autonomous system set path information and community information from contributing paths.
- The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths.
- The **summary-only** keyword filters all more specific routes from updates.
- The **route-policy** *route-policy-name* keyword and argument specify the route policy used to set the attributes of the aggregate route.

Step 9 **network** {*ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# network 172.20.0.0/16
```

Configures the local router to originate and advertise the specified network.

Step 10 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# exit
```

Exits VRF address family configuration mode and returns the router to VRF configuration mode for BGP routing.

Step 11 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24
```

Places the router in VRF neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 12 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 13 **password** { **clear** | **encrypted** } *password*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# password clear pswd123
```

Configures neighbor 172.168.40.24 to use MD5 authentication with the password pswd123.

Step 14 **ebgp-multihop** [*ttl-value*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop
```

Allows a BGP connection to neighbor 172.168.40.24.

Step 15 **address-family ipv4 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
```

Enters VRF neighbor address family configuration mode for BGP routing.

Step 16 **allowas-in** [*as-occurrence-number*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 3
```

Replaces the neighbor autonomous system number (ASN) with the PE ASN in the AS path three times.

Step 17 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in
```

Applies the In-Ipv4 policy to inbound IPv4 unicast routes.

Step 18 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in
```

Applies the In-Ipv4 policy to outbound IPv4 unicast routes.

Step 19 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 RIPv2 as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions using Routing Information Protocol version 2 (RIPv2).

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **vrf** *vrf-name*
4. **interface** *type instance*
5. **site-of-origin** { *as-number : number* | *ip-address : number* }
6. **exit**
7. Do one of the following:
   - **redistribute bgp** *as-number* [[**external** | **internal** | **local**] [**route-policy** *name*]
   - **redistribute connected** [**route-policy** *name*]
   - **redistribute isis** *process-id* [**level-1** | **level-1-2** | **level-2**] [**route-policy** *name*]
   - **redistribute eigrp** *as-number* [**route-policy** *name*]
   - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**route-policy** *name*]
   - **redistribute static** [**route-policy** *name*]
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **router rip**

Example:

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

Enters the Routing Information Protocol (RIP) configuration mode allowing you to configure the RIP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-rip)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for RIP routing.

Step 4 **interface** *type instance*

Example:

```
RP/0/RP0/CPU0:router(config-rip-vrf)# interface TenGigE 0/3/0/0
```

Enters VRF interface configuration mode.

Step 5 **site-of-origin** { *as-number : number* | *ip-address : number* }

Example:

```
RP/0/RP0/CPU0:router(config-rip-vrf-if)# site-of-origin 200:1
```

Identifies routes that have originated from a site so that the re-advertisement of that prefix back to the source site can be prevented. Uniquely identifies the site from which a PE router has learned a route.

Step 6 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-rip-vrf-if)# exit
```

Exits VRF interface configuration mode, and returns the router to VRF configuration mode for RIP routing.

Step 7 Do one of the following:

- **redistribute bgp** *as-number* [[**external** | **internal** | **local**] [**route-policy** *name*]
- **redistribute connected** [**route-policy** *name*]
- **redistribute isis** *process-id* [**level-1** | **level-1-2** | **level-2**] [**route-policy** *name*]
- **redistribute eigrp** *as-number* [**route-policy** *name*]
- **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**route-policy** *name*]
- **redistribute static** [**route-policy** *name*]

Example:

```
RP/0/RP0/CPU0:router(config-rip-vrf)# redistribute connected
```

Causes routes to be redistributed into RIP. The routes that can be redistributed into RIP are:

- Border Gateway Protocol (BGP)
- Connected
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- Intermediate System-to-Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Static

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

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

end - Prompts user to take one of these actions:

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

Configuring Static Routes Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use static routes.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **vrf** *vrf-name*
4. **address-family ipv4 unicast**
5. *prefix/mask* [**vrf** *vrf-name*] { *ip-address* | *type interface-path-id* }
6. *prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect**
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **router static**

Example:

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

Enters static routing configuration mode allowing you to configure the static routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-static)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for static routing.

Step 4 **address-family ipv4 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf)# address-family ipv4 unicast
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 *prefix/mask* [**vrf** *vrf-name*] { *ip-address* | *type interface-path-id* }

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 10.1.1.1
```

Assigns the static route to vrf_1.

Step 6 *prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect**

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 bfd fast-detect
```

Enables bidirectional forwarding detection (BFD) to detect failures in the path between adjacent forwarding engines.

This option is available is when the forwarding router address is specified in Step 5 .

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 OSPF as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Open Shortest Path First (OSPF).

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **vrf** *vrf-name*
4. **router-id** {*router-id* | type interface-path-id}
5. Do one of the following:
   - **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute static** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

- **redistribute eigrp** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute rip** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

6. **area** *area-id*
7. **interface** type interface-path-id
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **router ospf** *process-name*

Example:

```
RP/0/RP0/CPU0:router(config)# router ospf 109
```

Enters OSPF configuration mode allowing you to configure the OSPF routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.

Step 4 **router-id** {*router-id* | type interface-path-id}

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# router-id 172.20.10.10
```

Configures the router ID for the OSPF routing process.

Step 5 Do one of the following:

- **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute static** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute eigrp** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

- **redistribute rip** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# redistribute connected
```

Causes routes to be redistributed into OSPF. The routes that can be redistributed into OSPF are:

- Border Gateway Protocol (BGP)
- Connected
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- OSPF
- Static
- Routing Information Protocol (RIP)

Step 6 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# area 0
```

Configures the OSPF area as area 0.

Step 7 **interface** type interface-path-id

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf-ar)# interface TenGigE 0/3/0/0
```

Associates interface TenGigE 0/3/0/0 with area 0.

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

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

end - Prompts user to take one of these actions:

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

Configuring EIGRP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Enhanced Interior Gateway Routing Protocol (EIGRP).

Using EIGRP between the PE and CE routers allows you to transparently connect EIGRP customer networks through an MPLS-enable Border Gateway Protocol (BGP) core network so that EIGRP routes are redistributed through the VPN across the BGP network as internal BGP (iBGP) routes.

Before you begin

BGP is configured in the network. See the *Implementing BGP* module in the *Routing Configuration Guide for Cisco NCS 6000 Series Routers*

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **vrf** *vrf-name*
4. **address-family ipv4**
5. **router-id** *router-id*
6. **autonomous-system** *as-number*
7. **default-metric** *bandwidth delay reliability loading mtu*
8. **redistribute** { { **bgp** | **connected** | **isis** | **ospf** | **rip** | **static** } [*as-number* | *instance-name*] } [**route-policy** *name*]
9. **interface** *type interface-path-id*
10. **site-of-origin** { *as-number:number* | *ip-address : number* }
11. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **router eigrp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router eigrp 24
```

Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for EIGRP routing.

Step 4 **address-family ipv4**

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf)# address family ipv4
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 **router-id** *router-id*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# router-id 172.20.0.0
```

Configures the router ID for the Enhanced Interior Gateway Routing Protocol (EIGRP) routing process.

Step 6 **autonomous-system** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# autonomous-system 6
```

Configures the EIGRP routing process to run within a VRF.

Step 7 **default-metric** *bandwidth delay reliability loading mtu*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# default-metric 100000 4000 200 45 4470
```

Sets the metrics for an EIGRP.

Step 8 **redistribute** { { **bgp** | **connected** | **isis** | **ospf** | **rip** | **static** } [*as-number* | *instance-name*] } [**route-policy** *name*]

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# redistribute connected
```

Causes connected routes to be redistributed into EIGRP.

Step 9 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# interface TenGigE 0/3/0/0
```

Associates interface TenGigE 0/3/0/0 with the EIGRP routing process.

Step 10 **site-of-origin** { *as-number:number* | *ip-address : number* }

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# site-of-origin 201:1
```

Configures site of origin (SoO) on interface TenGigE 0/3/0/0.

Step 11 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 EIGRP Redistribution in the MPLS VPN

Perform this task for every provider edge (PE) router that provides VPN services to enable Enhanced Interior Gateway Routing Protocol (EIGRP) redistribution in the MPLS VPN.

Before you begin

The metric can be configured in the route-policy configuring using the **redistribute** command (or configured with the **default-metric** command). If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route is not installed in the EIGRP database. If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route is not advertised to the CE router. See the *Implementing EIGRP*module in the *Routing Configuration Guide for Cisco NCS 6000 Series Routers*.

Restriction Redistribution between native EIGRP VPN routing and forwarding (VRF) instances is not supported. This behavior is designed.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **vrf** *vrf-name*
4. **address-family ipv4**
5. **redistribute bgp** [*as-number*] [**route-policy** *policy-name*]
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **router eigrp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router eigrp 24
```

Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode for EIGRP routing.

Step 4 **address-family ipv4**

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf)# address family ipv4
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 **redistribute bgp** [*as-number*] [**route-policy** *policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 24 route-policy policy_A
```

Causes Border Gateway Protocol (BGP) routes to be redistributed into EIGRP.

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.

Verifying the MPLS Layer 3 VPN Configuration

Perform this task to verify the MPLS Layer 3 VPN configuration.

SUMMARY STEPS

1. **show running-config router bgp** *as-number* **vrf** *vrf-name*
2. **show running-config routes**
3. **show ospf vrf** *vrf-name* **database**
4. **show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address*
5. **show bgp vrf** *vrf-name* **summary**

6. **show bgp vrf** *vrf-name* **neighbors** *ip-address*
7. **show bgp vrf** *vrf-name*
8. **show route vrf** *vrf-name ip-address*
9. **show bgp vpn unicast summary**
10. **show running-config router isis**
11. **show running-config mpls**
12. **show isis adjacency**
13. **show mpls ldp forwarding**
14. **show bgp vpnv4 unicast** or **show bgp vrf** *vrf-name*
15. **show bgp vrf** *vrf-name* **imported-routes**
16. **show route vrf** *vrf-name ip-address*
17. **show cef vrf** *vrf-name ip-address*
18. **show cef vrf** *vrf-name ip-address* **location** *node-id*
19. **show bgp vrf** *vrf-name ip-address*
20. **show ospf vrf** *vrf-name* **database**

DETAILED STEPS

Step 1 **show running-config router bgp** *as-number* **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router# show running-config router bgp 3 vrf vrf_A
```

Displays the specified VPN routing and forwarding (VRF) content of the currently running configuration.

Step 2 **show running-config routes**

Example:

```
RP/0/RP0/CPU0:router# show running-config routes
```

Displays the Open Shortest Path First (OSPF) routes table in the currently running configuration.

Step 3 **show ospf vrf** *vrf-name* **database**

Example:

```
RP/0/RP0/CPU0:router# show ospf vrf vrf_A database
```

Displays lists of information related to the OSPF database for a specified VRF.

Step 4 **show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router# show running-config router bgp 3 vrf vrf_A neighbor 172.168.40.24
```

Displays the Border Gateway Protocol (BGP) VRF neighbor content of the currently running configuration.

Step 5 **show bgp vrf** *vrf-name* **summary**

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A summary
```

Displays the status of the specified BGP VRF connections.

Step 6 **show bgp vrf** *vrf-name* **neighbors** *ip-address*

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A neighbors 172.168.40.24
```

Displays information about BGP VRF connections to the specified neighbors.

Step 7 **show bgp vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A
```

Displays information about a specified BGP VRF.

Step 8 **show route vrf** *vrf-name ip-address*

Example:

```
RP/0/RP0/CPU0:router# show route vrf vrf_A 10.0.0.0
```

Displays the current routes in the Routing Information Base (RIB) for a specified VRF.

Step 9 **show bgp vpn unicast summary**

Example:

```
RP/0/RP0/CPU0:router# show bgp vpn unicast summary
```

Displays the status of all BGP VPN unicast connections.

Step 10 **show running-config router isis**

Example:

```
RP/0/RP0/CPU0:router# show running-config router isis
```

Displays the Intermediate System-to-Intermediate System (IS-IS) content of the currently running configuration.

Step 11 **show running-config mpls**

Example:

```
RP/0/RP0/CPU0:router# show running-config mpls
```

Displays the MPLS content of the currently running-configuration.

Step 12 **show isis adjacency**

Example:

```
RP/0/RP0/CPU0:router# show isis adjacency
```

Displays IS-IS adjacency information.

Step 13 **show mpls ldp forwarding**

Example:

```
RP/0/RP0/CPU0:router# show mpls ldp forwarding
```

Displays the Label Distribution Protocol (LDP) forwarding state installed in MPLS forwarding.

Step 14 **show bgp vpnv4 unicast** or **show bgp vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router# show bgp vpnv4 unicast
```

Displays entries in the BGP routing table for VPNv4 or VPNv6 unicast addresses.

Step 15 **show bgp vrf** *vrf-name* **imported-routes**

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A imported-routes
```

Displays BGP information for routes imported into specified VRF instances.

Step 16 **show route vrf** *vrf-name ip-address*

Example:

```
RP/0/RP0/CPU0:router# show route vrf vrf_A 10.0.0.0
```

Displays the current specified VRF routes in the RIB.

Step 17 **show cef vrf** *vrf-name ip-address*

Example:

```
RP/0/RP0/CPU0:router# show cef vrf vrf_A 10.0.0.1
```

Displays the IPv4 Cisco Express Forwarding (CEF) table for a specified VRF.

Step 18 **show cef vrf** *vrf-name ip-address* **location** *node-id*

Example:

```
RP/0/RP0/CPU0:router# show cef vrf vrf_A 10.0.0.1 location 0/1/cpu0
```

Displays the IPv4 CEF table for a specified VRF and location.

Step 19 **show bgp vrf** *vrf-name ip-address*

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A 10.0.0.0
```

Displays entries in the BGP routing table for VRF vrf_A.

Step 20 **show ospf vrf** *vrf-name* **database**

Example:

```
RP/0/RP0/CPU0:router# show ospf vrf vrf_A database
```

Displays lists of information related to the OSPF database for a specified VRF.

Configuration Examples for Implementing MPLS Layer 3 VPNs

The following section provides sample configurations for MPLS L3VPN features:

Configuring an MPLS VPN Using BGP: Example

The following example shows the configuration for an MPLS VPN using BGP on “vrf vpn1”:

```
address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!
interface Loopback0
  ipv4 address 10.0.0.1 255.255.255.255
!
interface TenGigE 0/1/0/0
  vrf vpn1
  ipv4 address 10.0.0.2 255.0.0.0
!
interface TenGigE 0/1/0/1
  ipv4 address 10.0.0.1 255.0.0.0
!
router ospf 100
  area 100
    interface loopback0
    interface TenGigE 0/1/0/1
  !
!
router bgp 100
  address-family vpnv4 unicast
```

```
  retain route-target route-policy policy1
  neighbor 10.0.0.3
    remote-as 100
    update-source Loopback0
    address-family vpnv4 unicast
  !
  vrf vpn1
    rd 100:1
    address-family ipv4 unicast
      redistribute connected
    !
    neighbor 10.0.0.1
      remote-as 200
      address-family ipv4 unicast
        as-override
        route-policy pass-all in
        route-policy pass-all out
      !
      advertisement-interval 5
    !
  !
!
mpls ldp
  route-id looback0
  interface TenGigE 0/1/0/1
!
```

Configuring the Routing Information Protocol on the PE Router: Example

The following example shows the configuration for the RIP on the PE router:

```
vrf vpn1
  address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!

interface TenGigE 0/1/0/0
  vrf vpn1
  ipv4 address 10.0.0.2 255.0.0.0
!

router rip
 vrf vpn1
  interface TenGigE 0/1/0/0
  !
  timers basic 30 90 90 120
  redistribute bgp 100
  default-metric 3
  route-policy pass-all in
 !
```

Configuring the PE Router Using EIGRP: Example

The following example shows the configuration for the Enhanced Interior Gateway Routing Protocol (EIGRP) on the PE router:

```
Router eigrp 10
 vrf VRF1
  address-family ipv4
   router-id 10.1.1.2
   default-metric 100000 2000 255 1 1500
   as 62
   redistribute bgp 2000
   interface Loopback0
   !
   interface TenGigE 0/6/0/0
```

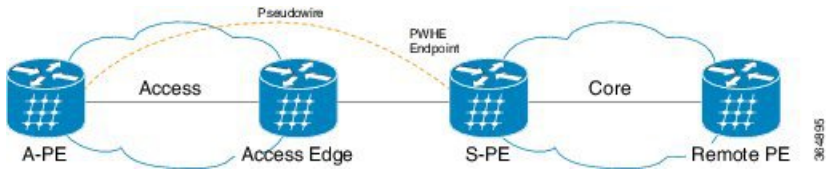
Pseudowire Headend

Pseudowire Headend (PWHE) feature allows termination of access pseudowires (PWs) into a Layer 3 (VRF or global) domain or into a Layer 2 domain. PWs provide an easy and scalable mechanism for tunneling customer traffic into a common IP/MPLS network infrastructure. PWHE allows customers to provision features such as quality of service (QoS), access lists (ACL), lawful intercept (LI), bidirectional forwarding detection (BFD), unicast reverse path forwarding (uRPF), NetFlow, and L3VPN on a per PWHE interface basis, on a service Provider Edge (PE) router.

Pseudowires (PWs) enable payloads to be transparently carried across IP/MPLS packet-switched networks (PSNs). Service providers are now extending PW connectivity into the access and aggregation regions of their networks. PWs are regarded as simple and manageable lightweight tunnels for returning customer traffic into core networks.

PWHE cross-connects to a pseudowire neighbor, which is reachable through recursive as well as non-recursive prefix. The reachability through recursive prefix is through introduction of BGP RFC3107 support on the Cisco NCS 6000 Series Router. Consider the following network topology for an example scenario.

Figure 10: Pseudowire Network



For PWHE cross-connect configuration, interconnectivity between A-PE (Access Provider Edge) and S-PE (Service Provider Edge) is through BGP RFC3107 that distributes MPLS labels along with IP prefixes. The customer network can avoid using an IGP to provide connectivity to the S-PE device, which is outside the customer's autonomous system.

For all practical purposes, the PWHE interface is treated like any other existing L3 interface. PWs operate in one of the following modes:

- Bridged interworking (VC type 4 or VC type 5) mode
- IP interworking mode (VC type 11)

With VC type 4 and VC type 5, PWs carry customer Ethernet frames (tagged or untagged) with IP payload. Thus, an S-PE device must perform ARP resolution for customer IP addresses learned over the PWHE. With VC type 4 (VLAN tagged) and VC type 5 (Ethernet port/raw), PWHE acts as a broadcast interface. Whereas with VC type 11 (IP Interworking), PWHE acts as a point-to-point interface. Therefore there are two types of PWHE interface:

- PW-Ether (for VC type 4 and 5)
- PW-IW (for VC type 11)

These PWs can terminate into a VRF or the IP global table on S-PE.

Benefits of PWHE

Some of the benefits of implementing PWHE are:

- Dissociates the customer facing interface (CFI) of the S-PE from the underlying physical transport media of the access or aggregation network.
- Reduces capex in the access or aggregation network and S-PE.
- Distributes and scales the customer facing Layer 2 UNI interface set.
- Implements a uniform method of OAM functionality.
- Enables providers to extend or expand the Layer 3 service footprints.
- Provides a method of terminating customer traffic into a next generation network (NGN).

Generic Interface List

A generic interface list contains a list of physical or bundle interfaces that is used in a PW-HE connection.

The generic interface list supports only main interfaces, and not sub-interfaces. The generic interface list is bi-directional and restricts both receive and transmit interfaces on access-facing line cards. The generic interface list has no impact on the core-facing side.

A generic interface list is used to limit the resources allocated for a PWHE interface to the set of interfaces specified in the list.

Only the S-PE is aware of the generic interface list and expects that the PWHE packets arrive on only line cards with generic interface list members on it. If packets arrive at the line card without generic interface list members on it, they are dropped.

Configure Pseudowire Headend

The Pseudowire Headend (PWHE) is created by configuring the pw-ether main interface, pw-ether subinterface, or pw-iw interface. The available PWHE types are pw-ether main interfaces, subinterfaces, and pw-iw interfaces. Unless specified otherwise, the term interface is applicable for pw-ether main interfaces, subinterfaces, and pw-iw interfaces.

For the PWHE to be functional, the cross-connect has to be configured completely. Configuring other Layer 3 (L3) parameters, such as VRF and IP addresses, are optional for the PWHE to be functional. However, the L3 features are required for the Layer 3 services to be operational; that is, for PW L3 termination. PWHE supports both IPv4 and IPv6 addresses.

PWHE Configuration Restrictions

These configuration restrictions are applicable for PWHE:

- The generic interface list members must be the superset of the equal-cost multi-path routing (ECMP) path list to the A-PE.
- Only eight generic interface lists are supported per A-PE neighbor address.
- Eight Layer 3 links per generic interface list are supported.
- Each generic interface list can have eight members in it including bundles.
- Only PW-Ether interfaces can be configured as L3 subinterfaces.
- Cross-connects that contain PW-Ether main interfaces can be configured as either VC type 4 or VC type 5. By default, the cross-connects are configured as VC type 5.
- Cross-connects that contain PW-Ether main interfaces that have L3 PW-Ether subinterfaces associated with them, are supported with only VC type 5.
- Cross-connects that contain PW-IW interfaces are only supported with IPv4 and VC type 11. PW-IW interfaces are the L3 virtual interfaces used for IP interworking. To configure the cross-connect as VC type 11, use the interworking ipv4 command.
- VC type 4 configuration is not supported on sub-interfaces. But if you try to configure, the system rejects the configuration. Though the system rejects the configuration, you must remove the configuration manually to avoid any issues that may occur later.
- PW-Ether interfaces and subinterfaces can be configured with both IPv4 and IPv6.
- QoS, ACL, LI, BFD, uRPF, NetFlow features are not supported on PW-Ether subinterface.
- PW-IW interfaces can be configured only with IPv4.
- Interface lists can accept 10-Gigabit Ethernet and 100-Gigabit Ethernet; other interfaces are rejected.
- Pseudowire redundancy, preferred path, local switching or L2TP for cross-connects configured with PWHE are not supported.
- Address family, Cisco Discovery Protocol (CDP) and MPLS configurations are not allowed on PWHE interfaces.
- Applications such as TE and LDP have checks for interface type and therefore do not allow PWHE to be configured.
- Only eBGP and static routes are supported.
- MAC address is not supported for a pw-iw interface.
- By default, control-word is enabled for PW-IW cross-connects.
- Members of Generic Interface List (GIL) can be physical or bundles links.
- GIL members must include all the ECMP paths.
- PWHE main interface and its sub-interfaces flap when you attempt to remove the PWHE main interface that has sub-interfaces. This happens even though the removal of the main interface configuration is rejected.

Configure Pseudowire Headend

This section describes how you can configure Pseudowire Headend feature at both S-PE and A-PE.

S-PE Configuration

Configuring PWHE involves these steps:

- Configure generic interface list
- Configure PWHE Ethernet and interworking interfaces, attach the generic interface list with a PWHE Ethernet and interworking interfaces
- Configure PW class for Ethernet and interworking interfaces
- Configure cross-connect using PWHE Ether and PWIW interfaces

```
/* S-PE Configuration */

/* Configure generic interface list for PWHE Ethernet interface */

Router# configure
Router(config)# generic-interface-list pwhe-list-APE2-1
Router(config-gen-if-list)# interface Bundle-Ether200
Router(config-gen-if-list)# interface TenGigE0/0/0/0/4
Router(config-gen-if-list)# interface TenGigE0/0/0/0/5


/* Configure PWHE Ethernet interface and attach the generic interface list with a PWHE
Ethernet interface */

Router# configure
Router(config)# interface pw-ether 5001
Router(config-if)# ipv4 address 103.7.7.1 255.255.255.252
Router(config-if)# ipv6 address 103:107:1::1/126
Router(config-if)# attach generic-interface-list pwhe-list-APE2-1


/* Configure generic interface list for PW interworking interface */

Router# configure
Router(config)# generic-interface-list pwhe-list-APE-2
Router(config-gen-if-list)# interface Bundle-Ether201
Router(config-gen-if-list)# interface TenGigE0/0/0/0/6
Router(config-gen-if-list)# interface TenGigE0/0/0/0/7


/* Configure interworking interface and attach the generic interface list with an interworking
 interface */

Router# configure
Router(config)#  interface pw-iw 4001
Router(config-if)# ipv4 address 103.107.47.225 255.255.255.252
Router(config-if)# attach generic-interface-list pwhe-list-APE-2


/* Configure PW class for Ethernet interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  pw-class APE2-PE1-PORT
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word
```

```
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet


/* Configure cross-connect for PW Ethernet interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group APE2-PE1-PORT
Router(config-l2vpn-xc)# p2p APE2-PE1-5001
Router(config-l2vpn-xc-p2p)# interface PW-Ether5001
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 100.1.8.1 pw-id 5001
Router(config-l2vpn-xc-p2p-pw)# pw-class APE2-PE1-PORT


/* Configure PW class for interworking interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  pw-class APE-InetRI-PWIW-4001
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word


/* Configure cross-connect for PW interworking interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group APE-InetRI-PWIW-4001
Router(config-l2vpn-xc)# p2p APE-InetRI-PWIW-4001
Router(config-l2vpn-xc-p2p)# interface PW-IW4001
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 100.1.9.1 pw-id 4001
Router(config-l2vpn-xc-p2p-pw)# pw-class APE-InetRI-PWIW-4001
Router(config-l2vpn-xc-p2p-pw)# interworking ipv4
```

A-PE Configuration

Configuring PWHE involves these steps:

- Configure PWHE Ethernet and interworking interfaces, attach the generic interface list with a PWHE Ethernet and interworking interfaces
- Configure PW class for Ethernet and interworking interfaces
- Configure cross-connect using PWHE Ether and PWIW interfaces

```
/* A-PE Configuration */

/* Configure PWHE  Ethernet interface */

Router# configure
Router(config)# interface TenGigE0/0/0/0 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config-subif)# rewrite ingress tag pop 1 symmetric


/* Configure interworking interface */

Router# configure
Router(config)# interface Serial0/5/0/0/8 l2transport
```

```
/* Configure PW Class for Ethernet interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class APE2-PE1-PORT
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet


/* Configure Cross-connect for Ethernet interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group APE2-PE1-PORT
Router(config-l2vpn-xc)# p2p APE2-PE1-5001
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/0
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 100.1.1.1 pw-id 5001
Router(config-l2vpn-xc-p2p-pw)# pw-class APE2-PE1-PORT


/* Configure PW Class for interworking interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class APE-InetRI-PWIW-4001
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word



/* Configure Cross-connect for interworking interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group APE-InetRI-PWIW-4001
Router(config-l2vpn-xc)# p2p APE-InetRI-PWIW-4001
Router(config-l2vpn-xc-p2p)# interface Serial0/5/0/0/8
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 100.1.1.1 pw-id 4001
Router(config-l2vpn-xc-p2p-pw)# pw-class APE-InetRI-PWIW-4001
Router(config-l2vpn-xc-p2p-pw)# interworking ipv4
```

Configure PWHE subinterface

```
/* Configure PWHE subinterface */
Router# configure
Router(config)# interface PW-Ether5001.1001
Router(config-subif)# ipv4 address 105.1.1.1 255.255.255.252
Router(config-subif)# ipv6 address 105:1:1::1/126
Router(config-subif)# encapsulation dot1q 1001
```

Running Configuration

This section shows Pseudowire Headend running configuration.

```
/* On S-PE */

/* Ethernet interface */

configure
```

```
 generic-interface-list pwhe-list-APE2-1
  interface Bundle-Ether200
  interface TenGigE0/0/0/0/4
  interface TenGigE0/0/0/0/5
  !
!

configure
 interface PW-Ether5001
  ipv4 address 103.107.1.1 255.255.255.252
  ipv6 address 103:107:1::1/126
  attach generic-interface-list pwhe-list-APE2-1

!

l2vpn
 pw-class APE2-PE1-PORT
  encapsulation mpls
   control-word
   transport-mode ethernet
  !

!

l2vpn
 xconnect group APE2-PE1-PORT
  p2p APE2-PE1-5001
   interface PW-Ether5001
   neighbor ipv4 100.1.8.1 pw-id 5001
    pw-class APE2-PE1-PORT



/* Interworking interface */

configure
 generic-interface-list pwhe-list-APE-2
  interface Bundle-Ether200
  interface TenGigE0/0/0/0/6
  interface TenGigE0/0/0/0/7
  !
!


configure
 interface PW-IW4001
  ipv4 address 103.107.47.225 255.255.255.252
  attach generic-interface-list pwhe-APE-2

!

l2vpn
 pw-class APE-InetRI-PWIW-4001
  encapsulation mpls
   control-word
   !
!

l2vpn
 xconnect group APE-InetRI-PWIW-4001
  p2p APE-InetRI-PWIW-4001
   interface PW-IW4001
   neighbor ipv4 100.1.9.1 pw-id 4001
    pw-class APE-InetRI-PWIW-4001
```

```
    !
    interworking ipv4
    !
!

!
```

```
/* On A-PE */

/* Ethernet interface */

configure
 interface TenGigE0/0/0/0 l2transport
  encapsulation dot1q 1001
  rewrite ingress tag pop 1 symmetric
!

l2vpn
 pw-class APE2-PE1-PORT
  encapsulation mpls
   control-word
   transport-mode ethernet
  !
!

l2vpn
 xconnect group APE2-PE1-PORT
  p2p APE2-PE1-5001
   interface TenGigE0/0/0/0
   neighbor ipv4 100.1.1.1 pw-id 5001
    pw-class APE2-PE1-PORT
   !
!


/* Interworking interface */

configure
 interface Serial0/5/0/0/8 l2transport
 !
!

l2vpn
 pw-class APE-InetRI-PWIW-4001
  encapsulation mpls
  control-word
  !
!

l2vpn
 xconnect group APE-InetRI-PWIW-4001
  p2p APE-InetRI-PWIW-4001
   interface Serial0/5/0/0/8
   neighbor ipv4 100.1.1.1 pw-id 4001
   pw-class APE-InetRI-PWIW-4001
   !
   interworking ipv4
   !
!
```

```
/* Configure PWHE subinterface */

configure
 interface PW-Ether5001.1001
  ipv4 address 105.1.1.1 255.255.255.252
  ipv6 address 105:1:1::1/126
  encapsulation dot1q 1001
```

Verification

The show outputs given in the following section display the details of the configuration of PW Ethernet interface and cross-connect, and the status of their configuration on S-PE and A-PE.

```
/* S-PE Configuration */

Router-S-PE#  show l2vpn xconnect interface pw-ether 5001 detail
Group APE2-PE1-PORT, XC APE2-PE1-5001, state is up; Interworking none
  AC: PW-Ether5001, state is up
    Type PW-Ether
    Interface-list: pwhe-list-APE2-1
    Replicate status:
    BE616: success
    BE617: success
    MTU 8986; interworking none
    Internal label: 25002
    Statistics:
      packets: received 96860, sent 101636
      bytes: received 7285334, sent 8703696
  PW: neighbor 100.1.8.1, PW ID 5001, state is up ( established )
    PW class APE2-PE1-PORT, XC ID 0xfffe07d0
    Encapsulation MPLS, protocol LDP
    Source address 100.1.1.1
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        29012                          24001
      Group ID     0x4607d3c                      0x40000c0
      Interface    PW-Ether5001                   TenGigE0/0/0/0
      MTU          8986                           8986
      Control word disabled                       disabled
      PW type      Ethernet                       Ethernet
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x6                            0x6
                   (router alert label)           (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------
    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    MIB cpwVcIndex: 4294838224
    Create time: 26/09/2017 11:08:57 (18:23:34 ago)
    Last time status changed: 26/09/2017 11:28:59 (18:03:32 ago)
    Statistics:
      packets: received 96860, sent 101636
      bytes: received 7285334, sent 8703696
```

```
/* A-PE configuration details */

Router-A-PE# show l2vpn xconnect interface te0/0/0/0 detail
Group APE2-PE1-PORT, XC APE2-PE1-5001, state is up; Interworking none
  AC: TenGigE0/0/0/0, state is up
    Type Ethernet
    MTU 8986; XC ID 0x1084455; interworking none
    Statistics:
      packets: received 399484457, sent 1073874787256
      bytes: received 42812068782, sent 81549786107821
  PW: neighbor 100.1.1.1, PW ID 5001, state is up ( established )
    PW class APE2-PE1-PORT, XC ID 0xc0000fa1
    Encapsulation MPLS, protocol LDP
    Source address 100.1.8.1
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        24001                          29012
      Group ID     0x40000c0                      0x4607d3c
      Interface    TenGigE0/0/0/0                 PW-Ether5001
      MTU          8986                           8986
      Control word disabled                       disabled
      PW type      Ethernet                       Ethernet
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x6                            0x6
                   (router alert label)           (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------
    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    MIB cpwVcIndex: 3221229473
    Create time: 04/09/2017 17:48:35 (3w1d ago)
    Last time status changed: 26/09/2017 23:37:11 (18:01:16 ago)
    Last time PW went down: 26/09/2017 23:21:53 (18:16:34 ago)
    Statistics:
      packets: received 1073874787256, sent 399484457
      bytes: received 81549786107821, sent 42812068782
```

The show outputs given in the following section display the details of the configuration of PW interworking interface and cross-connect, and the status of their configuration on S-PE and A-PE.

```
/* S-PE Configuration */

Router-S-PE#  show l2vpn xconnect interface pw-iw 4001 detail
Group APE-InetRI-PWIW-4001, XC APE-InetRI-PWIW-4001, state is up; Interworking IPv4
  AC: PW-IW4001, state is up
    Type PW-IW
    Interface-list: pwhe-APE-2
    Replicate status:
```

```
    BE616: success
    MTU 4470; interworking IPv4
    Internal label: 35423
    Statistics:
      packets: received 185986, sent 185985
      bytes: received 134084287, sent 134654943
  PW: neighbor 100.1.9.1, PW ID 4001, state is up ( established )
    PW class APE-InetRI-PWIW-4001, XC ID 0xfffe1058
    Encapsulation MPLS, protocol LDP
    Source address 100.1.1.1
    PW type IP, control word enabled, interworking IPv4
    PW backup disable delay 0 sec
    Sequencing not set

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        152284                         24018
      Group ID     0x84003ed4                     0xe004040
      Interface    PW-IW4001                      Serial0/5/0/0/8
      MTU          4470                           4470
      Control word enabled                        enabled
      PW type      IP                             IP
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x7                            0x7
                   (control word)                 (control word)
                   (router alert label)           (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------
    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    MIB cpwVcIndex: 4294840408
    Create time: 04/10/2017 09:55:04 (00:21:36 ago)
    Last time status changed: 04/10/2017 10:02:45 (00:13:56 ago)
    Statistics:
      packets: received 185986, sent 185985
      bytes: received 134084287, sent 134654943
```

```
/* A-PE configuration details */

Router-A-PE# show interface pw-iw 4001
PW-IW4001 is up, line protocol is up
  Interface state transitions: 1
  Hardware is PWHE VC11 IP Interworking Interface
  Internet address is 103.107.47.225/30
  MTU 4470 bytes, BW 10000 Kbit (Max: 10000 Kbit)
     reliability 255/255, txload 7/255, rxload 7/255
  Encapsulation PW-IW,  loopback not set,
  Last link flapped 00:14:44
    L2Overhead: 0
    Generic-Interface-List: pwhe-APE-2
  Last input 00:11:04, output 00:11:04
  Last clearing of "show interface" counters never
  5 minute input rate 277000 bits/sec, 48 packets/sec
  5 minute output rate 293000 bits/sec, 51 packets/sec
     185986 packets input, 134084287 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     185985 packets output, 134654943 bytes, 0 total output drops
```

```
     Output 0 broadcast packets, 0 multicast packets
```

The show output given in the following section display the details of the configuration of PWHE subinterface.

```
Router# show interface pw-ether 5001.1001
PW-Ether5001.1001 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is ac19.7200.0001
  Internet address is 105.1.1.1/30
  MTU 9004 bytes, BW 10000 Kbit (Max: 10000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation 802.1Q Virtual LAN, VLAN Id 1001,  loopback not set,
  Last link flapped 00:14:56
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 1000 bits/sec, 0 packets/sec
     196 packets input, 15554 bytes, 1282 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     1416 packets output, 923195 bytes, 3 total output drops
     Output 0 broadcast packets, 15 multicast packets
```

PWHE Load Balancing using FAT Label

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
PWHE Load Balancing using FAT Label	Release 7.4.1	This feature allows you to generate a flow-aware transport (FAT) label for the traffic going out of the PWHE-L3 interface on the PE device. P routers use the FAT label to load balance the traffic based on the flow but not on the VC label. This feature provides a better traffic distribution across ECMP paths.

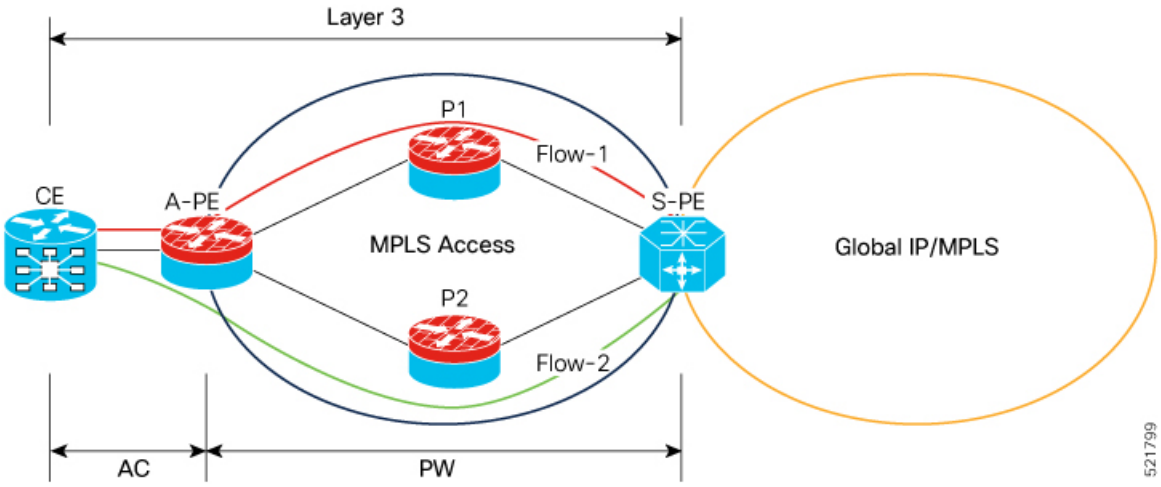
FAT labels provide the capability to identify individual flows within one PW traffic, and provide routers the ability to use these flows to load balance the traffic. A flow is identified by the source and destination IP address of the traffic and is defined as an indivisible packet having the same source and destination pair which is sent from a source PE to a destination PE.

An extra label, which is called the flow label is added for each unique incoming flow at the A-PE. A flow label is created based on indivisible packet flows entering an imposition PE and is inserted as the lowermost label in the packet. The flow label contains the end of the label stack (EOS) bit set and is inserted after the VC label. Core routers use the flow label for load balancing to provide better traffic distribution across ECMP paths.

When a flow label exists in the MPLS stack, the traffic is hashed based on the flow label instead of the VC label. This helps improve the efficiency of the load balancing method compared to the VC label hashing.

Access Provider Edge (A-PE) can use the flow label for load balancing the traffic which provides a better traffic distribution across ECMP paths.

Figure 11: Topology



In this topology, the imposition router, A-PE, adds a FAT label to the traffic. The P router uses the FAT label to load balance the traffic among the ECMP paths.

A-PE receives two indivisible packet flows from the CE. A-PE adds a unique FAT label to two different flows. A-PE sends the traffic with Flow-1 label through P1 to Service Provider Edge (S-PE) and the traffic with Flow-2 label through P2 to S-PE.

Configure PWHE Load Balancing using FAT Label

Perfrom these tasks to configure PWHE load balancing using FAT label.

- Configure PWHE Ethernet interface and attach the generic interface list with a PWHE Ethernet interface
- Configure cross-connect between S-PE and A-PE

Configuration Example

Perform these tasks at S-PE.

```
/* Configure PWHE Ethernet interface and attach the generic interface list with a PWHE
Ethernet interface */
Router# configure
Router(config)# generic-interface-list gil1
Router(config-gen-if-list)# interface TenGigE0/0/0/0
Router(config-gen-if-list)# interface TenGigE0/0/0/1
Router(config-gen-if-list)# interface TenGigE0/0/0/31
Router(config-gen-if-list)# exit
Router(config)# interface PW-Ether1
Router(config-if) mtu 1514
Router(config-if)# ipv4 address 10.10.10.2 255.255.255.0
Router(config-if)# ipv6 address 10:10:10::2/64
Router(config-if)# attach generic-interface-list gil1

/* Configure cross-connect between S-PE and A-PE */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class PWE
Router(config-l2vpn-pwc)# encapsulation mpls
```

```
Router(config-l2vpn-pwc-mpls)# control-word
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet
Router(config-l2vpn-pwc-mpls)# load-balancing flow-label both
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn-pwc-mpls)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCPW1
Router(config-l2vpn-xc)# p2p 1
Router(config-l2vpn-xc-p2p)# interface PW-Ether1
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 209.165.200.225 pw-id 1 > loopback of A-PE
Router(config-l2vpn-xc-p2p-pw)# pw-class PWE
Router(config-l2vpn-xc-p2p-pw)# commit
```

Perform these tasks at A-PE.

```
/* Configure Ethernet L2 transport interface */
Router# configure
Router(config)# interface TenGigE0/0/0/5/7 l2transport

/* Configure PWHE cross-connect */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class PWE
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet
Router(config-l2vpn-pwc-mpls)# load-balancing flow-label both
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn-pwc-mpls)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCPW1
Router(config-l2vpn-xc)# p2p 1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/5/7
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 172.16.0.1 pw-id 1 > loopback of S-PE
Router(config-l2vpn-xc-p2p-pw)# pw-class PWE
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the running configuration of PWHE load balancing using FAT label.

```
/* S-PE Configuration */

generic-interface-list gil1
 interface TenGigE0/0/0/0
 interface TenGigE0/0/0/1
 interface TenGigE0/0/0/31
!
interface PW-Ether1
 mtu 1514
 ipv4 address 10.10.10.2 255.255.255.0
 ipv6 address 10:10:10::2/64
 attach generic-interface-list gil1
!
l2vpn
 pw-class PWE
  encapsulation mpls
   control-word
   transport-mode ethernet
```

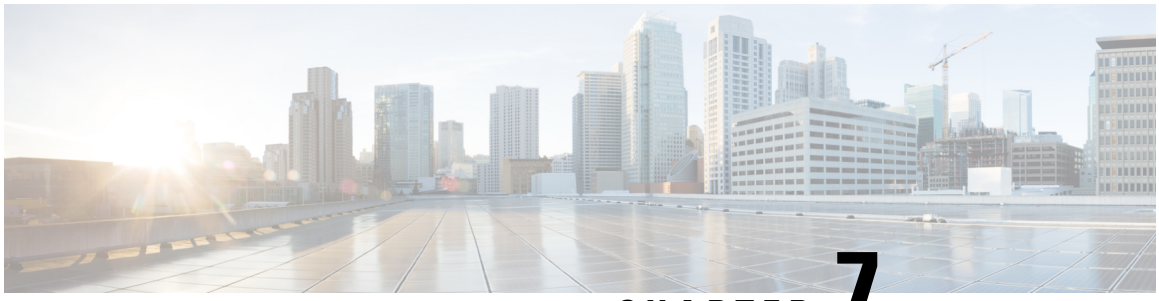
```
   load-balancing
   flow-label both
   !
  !
!
xconnect group XCPW1
  p2p 1
   interface PW-Ether1
   neighbor ipv4 209.165.200.225 pw-id 1 << loopback of A-PE
    pw-class PWE

/* A-PE Configuration */

interface TenGigE0/0/0/5/7
 l2transport
!
l2vpn
 pw-class PWE
  encapsulation mpls
   control-word
   transport-mode ethernet
   load-balancing
    flow-label both
   !
  !
!
xconnect group XCPW1
  p2p 1
   interface TenGigE0/0/0/5/7
   neighbor ipv4 172.16.0.1 pw-id 1 <<loopback of S-PE
    pw-class PWE
   !
  !
!
!
```

Verification

Verify the imposition and disposition labels.

```
Router:S-PE# show mpls lsd forwarding detail | i PW
Tue Jun  8 10:22:52.605 UTC
24010, (PW-HE, intf=PE1), 2 Paths,
   1/2: PW-HE Imp, intf_list_id=1,   flags=0x100 (Flow-Lbl)  ext_flags=0x0 ()
   2/2: PW-HE Imp, vc_type=5, cw=TRUE, nh=209.165.200.225, lbl=24019,
24013, (PW, pw=209.165.200.225:1), 1 Paths,
   1/1: PW-HE Disp, vc_type=5, cw=TRUE, ingress_intf=PE1


/* Verify imposition flag */
(flow label, control word)
Router:S-PE# show mpls forwarding labels 24010 hardware ingress detail location 0/0/CPU0 |
 i fl_flag
Tue Jun  8 10:23:55.036 UTC
  label/array: 0x5dd3       cw_enable  : 1            fl_flag    : 1
  label/array: 0x5dd3       cw_enable  : 1            fl_flag    : 1
  label/array: 0x5dd3       cw_enable  : 1            fl_flag    : 1
  label/array: 0x5dd3       cw_enable  : 1            fl_flag    : 1

/* Verify disposition flag */
(flow label, control word)
Router:S-PE# show mpls forwarding labels 24013 hardware ingress detail loc 0/0/CPU0 | i
fl_enable
```

```
Tue Jun  8 10:23:55.036 UTC
  fl_enable  : 1            uidb       : 3694         vc_type    : 1
  fl_enable  : 1            uidb       : 3694         vc_type    : 1
  fl_enable  : 1            uidb       : 3694         vc_type    : 1
  fl_enable  : 1            uidb       : 3694         vc_type    : 1


Router:S-PE# show mpls forwarding labels 24013 hardware ingress detail loc 0/0/CPU0 | i
cw_enable
Tue Jun  8 10:23:55.036 UTC
  entrytype  : PWHEDCAP     leaf       : 0            cw_enable  : 1
  entrytype  : PWHEDCAP     leaf       : 0            cw_enable  : 1
  entrytype  : PWHEDCAP     leaf       : 0            cw_enable  : 1
  entrytype  : PWHEDCAP     leaf       : 0            cw_enable  : 1
```



CHAPTER 7

Implementing of Layer 2 Access Lists

An Ethernet services access control list (ACL) consists 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), or VLAN ID.

This module describes tasks required to implement Ethernet services access lists on your Cisco NCS 6000 Series Network Convergence System Services Router.

Note For a complete description of the Ethernet services access list commands listed in this module, refer to the *Access List Commands on Cisco NCS 6000 Series Routers* module in the *Cisco NCS 6000 Series Network Convergence System Services Router IP Addresses and Services Command Reference* publication.

Feature History for Implementing Ethernet Services Access Lists on Cisco NCS 6000 Series Routers

Release	Modification
Release 6.6.1	This feature was introduced on Cisco NCS 6000 Series Routers.

Prerequisites for Implementing Layer 2 Access Lists

This prerequisite applies to implement access 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.

Information About Implementing Layer 2 Access Lists

Ethernet Services Access Lists Feature Highlights

Ethernet services access 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 and to resequence, add, or remove such statements from a named access list.
- Provides packet filtering on interfaces to forward packets.
- Ethernet services ACLs can be applied on interfaces, VLAN subinterfaces, bundle-Ethernet interfaces, EFPs, and EFPs over bundle-Ethernet interfaces. Atomic replacement of Ethernet services ACLs is supported on these physical interfaces.

Purpose of Ethernet Services Access Lists

Using ACL-based forwarding (ABF), Ethernet services access 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 an Ethernet Services Access List Works

An Ethernet services access 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 or leaving the router, but not traffic originating at the router.

Ethernet Services Access List Process and Rules

Use this process and rules when configuring an Ethernet services access 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. Incoming packets are processed before being routed to an outbound interface. 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.
- Outbound access lists process packets before they leave the router. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, permit means send it to the output buffer; 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.

Helpful Hints for Creating Ethernet Services Access Lists

Consider these when creating an Ethernet services access 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.

Source and Destination Addresses

Source MAC address and destination MAC address are two of the most typical fields on which to base an access list. Specify source MAC addresses to control packets from certain networking devices or hosts. Specify destination MAC addresses to control packets being sent to certain networking devices or hosts.

Ethernet Services Access List Entry Sequence Numbering

The ability to apply sequence numbers to Ethernet services access-list entries simplifies access list changes. The access list entry sequence numbering feature allows you to add sequence numbers to access-list entries and resequence them. When you add a new entry, you choose the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Sequence Numbering Behavior

These details the sequence numbering behavior:

- If entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum sequence number is 2147483646. If the generated sequence number exceeds this maximum number, this message is displayed:

```
Exceeded maximum sequence number.
```

- If you provide an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- ACL entries can be added without affecting traffic flow and hardware performance.
- Distributed support is provided so that the sequence numbers of entries in the route-switch processor (RSP) and interface card are synchronized at all times.

How to Implement Layer 2 Access Lists

Restrictions for Implementing Layer 2 Access Lists

These restrictions apply for implementing Ethernet services access lists:

- Ethernet services access lists are not supported over management interfaces.
- NetIO (software slow path) is not supported for Ethernet services access lists.

Configuring Ethernet Services Access Lists

This task configures an Ethernet services access list.

SUMMARY STEPS

1. **configure**
2. **ethernet-service access-list** *name*
3. [*sequence-number*] **{ permit | deny }** { *src-mac-address src-mac-mask* | **any** | **host** } [{ *ethertype-number* } | **vlan** *min-vlan-ID* [*max-vlan-ID*]] [**cos** *cos-value*] [**dei**] [**inner-vlan** *min-vlan-ID* [*max-vlan-ID*]] **inner-cos** *cos-value*] [**inner-dei**]
4. Repeat Step 3 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
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 **ethernet-service access-list** *name*

Example:

```
RP/0/RP0/CPU0:router(config)# ethernet-service access-list L2ACL2
```

Enters Ethernet services access list configuration mode and configures access list L2ACL2.

Step 3 [*sequence-number*] **{ permit | deny }** { *src-mac-address src-mac-mask* | **any** | **host** } [{ *ethertype-number* } | **vlan** *min-vlan-ID* [*max-vlan-ID*]] [**cos** *cos-value*] [**dei**] [**inner-vlan** *min-vlan-ID* [*max-vlan-ID*]] **inner-cos** *cos-value*] [**inner-dei**]

Example:

```
RP/0/RP0/CPU0:router(config-es-al)# 0 permit 1.2.3 3.2.1
or
RP/0/RP0/CPU0:router(config-es-al)# 30 deny any dei
```

Specifies one or more conditions allowed or denied, which determines whether the packet is passed or dropped.

Step 4 Repeat Step 3 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.

Allows you to revise an access list.

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.

What to Do Next

After creating an Ethernet services access list, you must apply it to an interface. See the Applying Ethernet Services Access Lists section for information about how to apply an access list.

Applying Ethernet Services Access Lists

After you create an access list, you must reference the access list to make it work. Access lists can be applied on either outbound or inbound interfaces. This section describes guidelines on how to accomplish this task for both terminal lines and network interfaces.

For inbound access lists, after receiving a packet, Cisco IOS XR software checks the source MAC address of the packet against the access list. If the access list permits the address, the software continues to process the packet. If the access list rejects the address, the software discards the packet.

For outbound access lists, after receiving and routing a packet to a controlled interface, the software checks the source MAC address of the packet against the access list. If the access list permits the address, the software sends the packet. If the access list rejects the address, the software discards the packet.

Note An empty access-list (containing no access control elements) cannot be applied on an interface.

Controlling Access to an Interface

This task applies an access list to an interface to restrict access to that interface. Access lists can be applied on either outbound or inbound interfaces.

SUMMARY STEPS

1. **configure**
2. **interface** *type instance*
3. **ethernet-services access-group** *access-list-name* { **ingress** | **egress** }
4. 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 **interface** *type instance*

Example:

```
RP/0/RP0/CPU0:router(config)#  interface gigabitethernet 0/2/0/
```

Configures an interface and enters interface configuration mode.

- The *type* argument specifies an interface type. For more information on interface types, use the question mark (?) online help function.
- The *instance* argument specifies either a physical interface instance or a virtual instance.
  - The naming notation for a physical interface instance is *rack/slot/module/port*. The slash (/) between values is required as part of the notation.
  - The number range for a virtual interface instance varies depending on the interface type.

Step 3 **ethernet-services access-group** *access-list-name* { **ingress** | **egress** }

Example:

```
RP/0/RP0/CPU0:router(config-if)# ethernet-services access-group p-in-filter ingress

RP/0/RP0/CPU0:router(config-if)# ethernet-services access-group p-out-filter egress
```

Controls access to an interface.

- Use the *access-list-name* argument to specify a particular Ethernet services access list.
- Use the *ingress* keyword to filter on inbound packets or the *egress* keyword to filter on outbound packets.

This example applies filters on packets inbound and outbound from GigabitEthernet interface 0/2/0/2.

Step 4 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.

Copying Ethernet Services Access Lists

This task copies an Ethernet services access list.

SUMMARY STEPS

1. **copy access-list ethernet-service** *source-acl destination-acl*
2. **show access-lists ethernet-services** [*access-list-name* | **maximum** | **standby** | **summary**]

DETAILED STEPS

Step 1 **copy access-list ethernet-service** *source-acl destination-acl*

Example:

```
RP/0/RP0/CPU0:router# copy access-list ethernet-service list-1 list-2
```

Creates a copy of an existing Ethernet services access list.

- Use the *source-acl* argument to specify the name of the access list to be copied.
- Use the *destination-acl* argument to specify where to copy the contents of the source access list.
  - The *destination-acl* argument must be a unique name; if the destination-acl argument name exists for an access list, the access list is not copied.

Step 2 **show access-lists ethernet-services** [*access-list-name* | **maximum** | **standby** | **summary**]

Example:

```
RP/0/RP0/CPU0:router# show access-lists ethernet-services list-2
```

(Optional) Displays the contents of a named Ethernet services access list. For example, you can verify the output to see that the destination access list list-2 contains all the information from the source access list list-1.

Resequencing Access-List Entries

This task shows how to reassign sequence numbers to entries in a named access list. Resequencing an access list is optional.

SUMMARY STEPS

1. **resequence access-list ethernet-service** *access-list-name* [*starting-sequence-number* [*increment*]]
2. Use the **commit** or **end** command.
3. **show access-lists ethernet-services** [*access-list-name* | **maximum** | **standby** | **summary**]

DETAILED STEPS

Step 1 **resequence access-list ethernet-service** *access-list-name* [*starting-sequence-number* [*increment*]]

Example:

```
RP/0/RP0/CPU0:router# resequence access-list ethernet-service L2ACL2 20 10
```

(Optional) Resequences the specified Ethernet services access list using the desired starting sequence number and the increment of sequence numbers.

- This example resequences an Ethernet services access list named L2ACL2. The starting sequence number is 20 and the increment is 10. If you do not select an increment, the default increment 10 is used.

Note If during the resequencing process it is determined that the ending number will exceed the maximum sequence number allowed, the configuration will not take effect and will be rejected. The sequence numbers will not be changed.

Step 2 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 3 **show access-lists ethernet-services** [*access-list-name* | **maximum** | **standby** | **summary**]

Example:

```
RP/0/RP0/CPU0:router# show access-lists ethernet-services L2ACL2
```

(Optional) Displays the contents of a named Ethernet services access list.

- Review the output to see that the access list includes the updated information.

Configuration Examples for Implementing Layer 2 Access Lists

Resequencing Entries in an Access List: Example

This example shows access-list resequencing. The starting value in the resequenced access list is 1, and the increment value is 2. The subsequent entries are ordered based on the increment values that users provide, and the range is from 1 to 2147483646.

When an entry with no sequence number is entered, by default, it has a sequence number of 10 more than the last entry in the access list.

```
ethernet service access-list acl_1
10 permit 1.2.3 4.5.6
20 deny 2.3.4 5.4.3
30 permit 3.1.2 5.3.4 cos 5

resequence access-list ethernet service acl_1 10 20

show access-list ethernet-service acl1_1

ipv4 access-list acl_1
  10 permit 1.2.3 4.5.6
  30 deny 2.3.4 5.4.3
  50 permit 3.1.2 5.3.4 cos 5
```

Adding Entries with Sequence Numbers: Example

In this example, a new entry is added to Ethernet services access list acl_5.

```
ethernet-service access-list acl_5
2 permit 1.2.3 5.4.3
5 permit 2.3.4. 6.5.4 cos 3
10 permit any dei
20 permit 6.5.4 1.3.5 VLAN vlan3

configure
 ethernet-service access-list acl_5
 15 permit 1.5.7 7.5.1
 end

ethernet-service access-list acl_5
2 permit 1.2.3 5.4.3
5 permit 2.3.4. 6.5.4 cos 3
10 permit any dei
15 permit 1.5.7 7.5.1
20 permit 6.5.4 1.3.5 VLAN vlan3
```