L2VPN Configuration Guide for Cisco 8000 Series Routers, Cisco IOS XR Releases

Cisco 8000 Series Routers
Updated September 11, 2026

Topics included

1 What's changed in this book....8
Whats changed in this book....9

2 YANG Data Models for L2VPN Features....10
Using YANG Data Models....11

3 Introduction to Layer 2 Virtual Private Networks....12
Layer 2 virtual private networks....13
Requirements for L2VPN network deployment....13
L2VPN interface types and key attributes....13
Benefits of Layer 2 virtual private networks....14

4 Virtual LANs in Layer 2 VPNs....16
VLANs in Layer 2 VPNs....17
VLAN services in Layer 2 VPNs....18
Benefits of VLANs in Layer 2 networks....19
Attachment circuit models in Layer 2 VPNs....19
Dot1Q Q-in-Q tunneling (0x8100/0x8100) requirements for VLAN subinterface encapsulation....19
Configure Dot1Q Q-in-Q (0x8100/0x8100) tunneling for VLAN subinterface encapsulation....20
Configure Dot1Q Q-in-Q (0x8100/0x8100) tag rewrite on a VLAN subinterface....21
Double-tagged 802.1ad encapsulation options for Layer 2 and Layer 3 physical and bundle subinterfaces....22
L2 VLAN subinterface encapsulation and rewrite....23
Configure exact matching for single-tagged encapsulations....24
Configure legacy QinQ encapsulation 0x9100/0x8100....25
Configure priority-tagged traffic on a VLAN subinterface....27
Supported encapsulation modes for Layer 2 interfaces....27
Encapsulation default catch-all mode....28
VLAN subinterfaces....30
VLAN subinterface characteristics....31
VLAN list and VLAN range....31
Best practice for high-density subinterface deployment....32
Configure VLAN subinterface....32
Ethernet flow points....34
Ethernet flow point features....35
Ethernet flow point operational capabilities....35
Frame identification methods for an EFP....36
Apply features....36

Features applied after EFP matching....36
Encapsulation modifications for EFP....37
Valid ingress rewrite actions....37
Configure VLAN header rewrite on Layer 2 subinterfaces....38
Data-forwarding behaviors....40
Ethernet flow point visibility....40
Configure Ethernet flow point interfaces....40

5 Transparent Layer 2 Protocol Tunneling....44
Transparent Layer 2 protocol tunneling....45
Requirements for transparent Layer 2 protocol tunneling....46
Supported protocols for transparent Layer 2 protocol tunneling....46
Protocol handling for transparent Layer 2 protocol tunneling....47
Verify transparent Layer 2 protocol tunneling....48

6 Link Bundles for Layer 2 VPNs....50
Link bundles for Layer 2 VPNs....51
Advantages of link bundles....51
Configure an interface link bundle....51
Configure a VLAN bundle....53
References for configuring link bundles....55
Characteristics of link bundles....55
Ethernet bundle configuration modes in Cisco IOS-XR....56
LACP compatibility requirements for link aggregates....57

7 Layer 2 Bridging Services....58
Layer 2 bridge services....59
Layer 2 bridge components....59
Bridge domains....59
Bridge ports....60
Bridge port flush and bridge flush....61
MAC address tables in bridge domains....63
Replication member lists....64
Configure a bridge domain....64
VLAN bridge modes....65
MAC address-related parameters in bridge domains....69
Flooding disable....77
Configure flooding at the bridge level....78
Virtual private LAN bridging services....79
VPLS service attributes....80
How VPLS works....80
Pseudowires in VPLS bridge domains....81
Configure a pseudowire under a bridge domain....81

Check ingress pseudowire statistics....82
Virtual forwarding instances in VPLS....83
VPLS pseudowire transport types....84
Supported pseudowire types and VLAN tag transport behavior....84
Pseudowire control-word....85
Flow-aware transport pseudowires....87
VPLS discovery and signaling models....90
How VPLS discovery and signaling work....91
BGP-based VPLS autodiscovery....91
How BGP autodiscovery with BGP signaling work....91
How BGP autodiscovery with LDP signaling work....93
CFM on VPLS....94
CFM components and protocols for VPLS....95
Restrictions for CFM on VPLS....96
Offload types and supported CCM timers for CFM on VPLS interfaces....96
How CFM on VPLS works....97
Configure CFM for VPLS services....98
Split-horizon groups....100
Supported split-horizon groups....102
Configure split-horizon groups....102
Traffic storm control....104
Storm control on a VPLS bridge....105
Supported traffic types....105
How traffic storm control works....105
Feature behavior for traffic storm control....106
Supported traffic types....106
Restrictions for traffic storm control....107
Configure traffic storm control....107
GTP load balancing....108
Key attributes of GTP....109
How GTP load balancing works....110
Guidelines for GTP load balancing....111

8 Pseudowire over MPLS and Point-to-Point Service Modes....114

Point-to-point Layer 2 services....115
Local switching....115
Attachment circuits....115
Pseudowires....115
Pseudowire over MPLS....115
Limitations for Pseudowire over MPLS....117
How pseudowire over MPLS works....117
How static cross-connect circuits work....118
Requirement: Static cross-connect circuit readiness....119
Static cross-connect circuit topology....119

Configure static point-to-point cross-connects....119
Configure dynamic point-to-point cross-connects....121
Pseudowire modes over MPLS....122
Ethernet port mode....122
VLAN mode....125
VLAN passthrough mode....128
Pseudowire redundancy....129
Inter-AS modes....133
Local switching between attachment circuits....137
Configure local switching between attachment circuits....139

9 Pseudowire Headend and Advanced Pseudowire Features....142
Virtual circuit connection verification features....143
Virtual circuit connection verification packet handling....144
Pseudowire headend....144
Requirement: Pseudowire headend hardware support....145
Pseudowire headend interface behavior....146
Benefits of pseudowire headend....146
How pseudowire headend works....146
Traffic flow types on PWHE interfaces....148
How PWHE decapsulation works....148
How PWHE encapsulation works....149
Generic interface lists....149
Restrictions for pseudowire headend....150
Pseudowire headend configuration guidelines....150
Configure pseudowire headend....151
GIL prune behavior for PWHE interfaces....154
Multisegment pseudowires....156
Main components and functions of multisegment pseudowires....156
Overcoming connectivity barriers with multisegment pseudowire....157
Benefits of multisegment pseudowire....157
How multisegment pseudowires work....158
Configure multisegment pseudowires....159

10 Traffic Engineering, Load Balancing, and Protection for L2 Services....164
Preferred tunnel path....165
Preferred tunnel path benefits....167
Preferred tunnel path restrictions....167
Configure preferred tunnel path....167
Configure MPLS-TE tunnel for VPLS....169
Configure VPLS over preferred TE tunnel....170
How MPLS PW traffic load balancing works on P routers....173
Load balance MPLS PW traffic using control word and flow label....174
L2VPN traffic load balancing on PE routers....180

Supported VLAN tag formats for load balancing....183
Load balancing scenarios for VPWS and VPLS unicast traffic....183
BUM traffic in VPLS service load balancing....184
Impact of disabling MPLS non-IP hash mode....185
Hash fields and load balancing for traffic received on L2 interfaces....185
Hash criteria for traffic received on L3 interfaces....186
G.8032 Ethernet ring protection switching....188
How G.8032 Ethernet ring protection switching works....189
G.8032 Ethernet ring protection switching restrictions....196
Configure G.8032 Ethernet ring protection switching....196
G.8032 Ethernet ring protection switching example....203
VPLS preferred path over SR-TE policy....207
SR-TE policy pseudowire path options....208
VPLS and SR-TE preferred path attributes....209
How VPLS services use SR-TE policies for preferred path selection....209
Configure VPLS preferred path over SR-TE policy....210

11 Integrated Routing and Bridging....220
Integrated routing and bridging....221
Bridge-group virtual interfaces....221
Best practice for IRB configuration....221
How packet forwarding works in IRB....222
Configure IRB....223
EVPN IRB....225

12 Multiple Spanning Tree Protocol....248
Multiple spanning tree protocol....249
MSTP instance mapping and load balancing....250
MSTP supported features....250
BPDU Guard....250
Flush containment....251
Bringup delay for MSTP interfaces....252
Restrictions for MSTP....252
Configure MSTP base services....253
MSTP interface parameters....254
MSTP enablement....254
Configurable MSTP global parameters....255
MSTP interface parameters....255
Configure MSTP region parameters....255
Configure MSTP instance parameters....257
Configure MSTP interface cost parameters....257
Configure MSTP interface protection parameters....258
Verify MSTP....259
MSTP configuration examples....260

Per-VLAN Rapid Spanning Tree....262
Characteristics of PVRST operation....263
Information about Multiple Spanning Tree Protocol....264
Spanning Tree Protocol loop prevention and redundancy....264
Attributes of MSTP regions....265
MSTP port fast....267
MSTP Root Guard....267
MSTP topology change guard....268

13 Ethernet Service Activation Testing with Y.1564....270
Y.1564 Ethernet service activation test....271
Ethernet service activation tests for SLA assurance....271
Key performance indicators for Y.1564....271
SLA validation objectives of Y.1564 tests....272
Supported encapsulation modes for Y.1564 service activation and deactivation tests....272
Restrictions for Y.1564 service activation test....273
Usage guidelines for SAT and SADT testing....273
Bandwidth parameters for Y.1564 service activation test....274
Service activation test targets for Y.1564 support....274
Start or stop a service activation test on an interface....274

1 What's changed in this book

Topics:

- Whats changed in this book

This cumulative guide provides a single, continuously updated version that includes all the latest IOS XR features and release updates. It simplifies your experience by letting you bookmark one link and access the complete guide, instead of navigating through multiple release-specific versions.

Whats changed in this book

This cumulative guide provides a single, continuously updated version that includes all the latest IOS XR features and release updates. It simplifies your experience by letting you bookmark one link and access the complete guide, instead of navigating through multiple release-specific versions.

Specific changes or updates tied to individual releases are clearly called out within the relevant sections. For a list of features introduced in a specific release, refer to the Release Notes.

The table lists the release numbers for which this document has been updated since its initial publication.

Table 1: Changes to this document

Date	Summary
September 2026	Republished for Release 26.3.1.
February 2026	Republished for Release 26.1.1.
December 2025	Republished for Release 25.4.1.
October 2025	First published for Release 25.3.1.

2 YANG Data Models for L2VPN Features

Topics:

- Using YANG Data Models

Lists the supported YANG data models and their corresponding L2VPN features on this platform.

This chapter provides information about the YANG data models for L2VPN features.

Using YANG Data Models

Provides a summary of the locations, tools, and resources for finding and exploring Cisco IOS XR YANG data models.

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the *Available-Content.md* file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.

3 Introduction to Layer 2 Virtual Private Networks

Topics:

- Layer 2 virtual private networks

Introduces Layer 2 virtual private networks, covering service requirements, supported interfaces, and key benefits to provide a foundational understanding of L2VPN technology, its deployment considerations, and operational advantages.

Layer 2 virtual private networks

Outlines the core concepts of Layer 2 virtual private networks, detailing L2VPN service requirements, supported interface types, and the primary benefits of deploying L2VPN solutions in modern network environments.

A Layer 2 virtual private network (L2VPN) is a private Layer 2 service that

- emulates a physical sub-network in an IP or MPLS network
- creates private connections between two points, and
- helps maintain customer privacy while using service provider resources to establish the network.

Requirements for L2VPN network deployment

Lists key requirements and considerations for building an L2VPN network.

To successfully deploy an L2VPN network, these requirements and capabilities are essential:

- Layer 2 connectivity coordination: The service provider establishes Layer 2 connectivity, while the customer uses these data link resources to build their own network.
- Customer network privacy: The service provider does not require knowledge of the customer's internal network topology, helping maintain customer privacy.
- Provider edge router capabilities: Provider edge (PE) routers must support encapsulation of Layer 2 protocol data units (PDUs) into Layer 3 packets, interconnection of any-to-any Layer 2 transports, MPLS tunneling mechanisms, and processing databases for circuit and connection information.
- Hardware support: Line cards and routers equipped with Q100, Q200, and P100-based Silicon One ASICs provide support for L2VPN services.

These requirements ensure successful and secure L2VPN deployment that meets both service provider and customer needs.

L2VPN interface types and key attributes

Lists the types and attributes of interfaces used in an L2VPN network.

L2VPN interfaces enable service providers to deliver Layer 2 connectivity between geographically separated customer sites. The main interface types used in L2VPN networks, along with their attributes, are:

- Attachment circuit (AC): Connects the customer site to the service provider's edge router, carrying Layer 2 traffic.
- Provider edge (PE) interface: Terminates the AC at the service provider's edge router and handles forwarding to the core.
- Core tunnel interface: Tunnels traffic from the PE router across the service provider core to reach another PE router.

Key facts about L2VPN interfaces:

- Each customer site connects to the nearest service provider edge router through an AC.
- Traffic from one customer site can be carried through the AC into the service provider core, traversing the core tunnel interface to reach another edge router.
- The receiving edge router uses another AC to deliver traffic to the remote customer site, maintaining Layer 2 connectivity across wide geographic distances.

Benefits of Layer 2 virtual private networks

Provides the main features and advantages of Layer 2 virtual private networks (VPNs) for service providers, including infrastructure consolidation and cost efficiency.

- Allow service providers to consolidate both Layer 2 and Layer 3 services onto a single infrastructure, simplifying network management and operations.
- Enable cost-effective delivery through a converged IP or MPLS network, reducing expenses associated with maintaining separate systems for different service types.

4 Virtual LANs in Layer 2 VPNs

Topics:

- VLANs in Layer 2 VPNs
- L2 VLAN subinterface encapsulation and rewrite
- VLAN subinterfaces
- Ethernet flow points

Outlines Virtual LAN operations in Layer 2 VPN environments, covering VLAN services, subinterface encapsulation, Ethernet Flow Points, configuration procedures, encapsulation types, and deployment guidelines for scalable and flexible networking.

VLANs in Layer 2 VPNs

Introduces VLAN services in Layer 2 VPNs, highlights the benefits of VLANs, explains attachment circuit types, describes Dot1Q Q-in-Q tunneling, and provides procedures for encapsulation, tag rewrite, and double-tagged 802.1ad options.

A VLAN in Layer 2 VPNs is a network segmentation method that

- uses VLAN-tagged Ethernet subinterfaces as attachment circuits to connect geographically separated customer sites
- inserts VLAN membership information into Ethernet frames according to IEEE 802.1Q, and
- improves segmentation, bandwidth use, and security between internal network segments.

A VLAN is a logical group of devices that communicate as if they are on the same LAN even when they are distributed across multiple LAN segments.

To create a Layer 2 VLAN attachment circuit, create a VLAN subinterface and include the **l2transport** keyword so the subinterface operates as a Layer 2 interface.

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) The support for Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation is now extended to all systems in the Cisco 8000 Series Routers.

Feature Name	Release Information	Feature Description
Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation	Release 24.1.1	We have optimized VLAN implementation by enabling service providers to: • expand VLAN space to segregate their networks for customers with multiple VLANs and overlapping VLAN IDs. • enhance service mapping for efficiently differentiating data packets and applying QoS policies based on users and services. Such optimization is possible because this release supports Dot1Q Q-in-Q (0x8100/0x8100) encapsulation for VLAN subinterfaces. This involves configuring these subinterfaces to add an outer 802.1Q tag to packets that are already carrying an 802.1Q VLAN tag. The feature introduces these changes: **CLI:** New L2VPN commands • **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* • **rewrite ingress tag** **YANG Data Model:** • New XPaths for `openconfig-interfaces.yang` (see GitHub, YANG Data Models Navigator) This feature is supported on Cisco 8000 series routers that are based on the Q200 silicon chip application-specific integrated circuit (ASIC).

VLAN services in Layer 2 VPNs

Provides details on the service-provider context, VLAN behavior, supported interface types, and configuration models for VLAN services in Layer 2 VPNs.

Service-provider context and VLAN behavior

Supported platforms for VLAN services

VLAN attachment circuit configuration model

The L2VPN feature enables service providers (SPs) to provide Layer 2 services to geographically disparate customer sites.

A virtual local area network (VLAN) is a group of devices on one or more LANs that are configured so that they can communicate as if they were attached to the same wire, even when they are located on different LAN segments.

Cisco IOS XR software supports VLAN subinterface configuration on:

- 400 Gigabit Ethernet interfaces
- 100 Gigabit Ethernet interfaces

VLAN attachment circuits (ACs) use a configuration model similar to basic VLAN configuration. To set up a VLAN AC:

- create a VLAN subinterface
- enter subinterface configuration mode
- configure the VLAN on the subinterface

To create an attachment circuit, the configuration includes the **l2transport** keyword in the interface command string to identify the interface as a Layer 2 interface.

Benefits of VLANs in Layer 2 networks

Lists the ways VLANs improve segmentation, efficiency, and security in Layer 2 networks.

VLANs provide several key benefits in Layer 2 networks:

- Simplify user and host management
- Support bandwidth allocation
- Optimize network resources
- Reduce unnecessary broadcast and multicast traffic by dividing large networks into smaller parts
- Increase security between internal network segments

Attachment circuit models in Layer 2 VPNs

Identifies the attachment circuit models used for VLAN services in Layer 2 VPNs.

These attachment circuit models are used for VLAN services in Layer 2 VPNs:

- Basic Dot1Q attachment circuit: matches traffic with a specific single VLAN tag.
- Q-in-Q attachment circuit: matches traffic with a specific outer VLAN tag and a specific inner VLAN tag.

Dot1Q Q-in-Q tunneling (0x8100/0x8100) requirements for VLAN subinterface encapsulation

Provides the requirements and key facts about Dot1Q Q-in-Q tunneling, double VLAN tags, and their roles in VLAN subinterface encapsulation.

Key facts about Dot1Q Q-in-Q tunneling

Requirements for VLAN subinterface encapsulation

Dot1Q Q-in-Q tunneling (also called stacked VLAN tagging or double VLAN) enables a scalable VLAN implementation by adding an extra 802.1Q tag to packets already tagged. This technique allows multiple VLAN domains within a larger network, helping service providers support multitenancy with overlapping VLAN IDs.

- Purpose: Expands VLAN capacity and facilitates service mapping with two levels of VLAN tags.
- Outer VLAN tag: Represents the service provider VLAN, added by the provider as frames enter the network. Can use EtherType 0x8100, 0x9100, or 0x88A8, depending on platform or configuration.
- Inner VLAN tag: Represents the customer VLAN, added by the customer network. Must use EtherType 0x8100.
- VLAN identifier assignment: All Ethernet subinterfaces default to 802.1Q VLAN encapsulation and must have a specific VLAN identifier explicitly defined before they can pass traffic.
- Multitenancy: Outer and inner VLAN tags enable the router to distinguish packets based on users and services, allowing multiple customers with overlapping VLAN IDs to share the same carrier network.
- Each subinterface must have a valid tagging protocol encapsulation and be assigned a VLAN identifier for traffic to pass.
- Double tagging allows for service mapping and distinction between provider and customer VLAN domains.

Q-in-Q tunneling creates a scalable, flexible VLAN environment supporting multiple customers over a single carrier network while maintaining clear distinction between provider and customer VLANs.

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

Configure Dot1Q Q-in-Q (0x8100/0x8100) tunneling for VLAN subinterface encapsulation

Configure a Layer 2 VLAN subinterface to use Q-in-Q tunneling with both an outer and inner 0x8100 tag.

Enable Q-in-Q encapsulation, allowing a VLAN subinterface to carry double VLAN tags (0x8100/0x8100) for service provider tunneling.

Q-in-Q tunneling allows a Layer 2 VLAN subinterface to encapsulate customer VLAN traffic with an additional provider VLAN tag. This configuration is required before a subinterface can pass traffic, with explicit VLAN identifiers provided.

Before you begin

Decide the outer and inner VLAN IDs for the service and identify the target subinterface.

Procedure

1. Enter interface configuration mode for the Layer 2 subinterface.

   Example

   ```
   Router# configure
   Router(config)# interface TenGigE 0/0/0/1.102 l2transport
   ```

2. Configure the Q-in-Q encapsulation.

Example

```
Router(config-subif)# encapsulation dot1q 200 second-dot1q 201
```

3. Commit the configuration.

Example

```
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# exit
```

4. Use the **show interfaces TenGigE 0/0/0/1.102** command to verify that the subinterface is operating in Q-in-Q mode.

Example

```
Router# show interfaces TenGigE 0/0/0/1.102
```

Successful verification shows the subinterface up and the line `Encapsulation 802.1Q-802.1Q Virtual LAN`.

```
HundredGigE0/0/0/1.102 is up, line protocol is up
Hardware is VLAN sub-interface(s)
Encapsulation 802.1Q-802.1Q Virtual LAN
```

Configure Dot1Q Q-in-Q (0x8100/0x8100) tag rewrite on a VLAN subinterface

Configure rewrite actions for double-tagged Layer 2 frames on a Q-in-Q VLAN subinterface.

This is an optional procedure. Use these example configurations to add or modify double Dot1Q Q-in-Q VLAN tags (with an outer tag of 0x8100) on Layer 2 Ethernet frames.

Enable the addition or modification of double Dot1Q (Q-in-Q) tags for ingress Layer 2 Ethernet frames, supporting flexible VLAN handling for provider and customer edge separation.

This procedure is common in service provider environments that use Q-in-Q tunneling, where double-tagged (VLAN stacked) frames need custom rewrite operations. Typical actions include removing, adding, or translating VLAN tags on incoming frames.

Before you begin

Make sure the Q-in-Q encapsulation is configured on the subinterface before you configure rewrite actions.

Procedure

1. Enter Layer 2 subinterface configuration mode.

Example

```
Router# configure
Router(config)# interface TenGigE 0/0/0/1.102 l2transport
```

2. Configure the double-tag encapsulation.

Example

```
Router(config-subif)# encapsulation dot1q 200 second-dot1q 201
```

3. Apply the required rewrite action.

Example

```
Router(config-subif)# rewrite ingress tag pop 2 symmetric
```

This example demonstrates push and translate operations for Q-in-Q services.

```
/* Configure Dot1Q Q-in-Q Tag Rewrite: Pop 2 */
interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 200 dot1q 201
  rewrite ingress tag pop 2 symmetric
 !
!

/* Configure Dot1Q Q-in-Q Tag Rewrite: Push */
interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 200 dot1q 201
  rewrite ingress tag push dot1q 200 second-dot1q 201 symmetric
 !
!
```

4. Commit the configuration.

Example

```
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# exit
```

Double-tagged 802.1ad encapsulation options for Layer 2 and Layer 3 physical and bundle subinterfaces

Provides the encapsulation requirements for double-tagged 802.1ad services on physical, bundle, and subinterface configurations.

Key facts

Subinterfaces are logical interfaces created on a hardware interface. These software-defined interfaces allow segregation of traffic into separate logical channels and help optimize physical-interface bandwidth.

The encapsulation options and requirements for double-tagged 802.1ad services are as follows:

- All Ethernet subinterfaces use 802.1Q VLAN encapsulation by default.
- A VLAN identifier must be explicitly assigned to each subinterface to allow traffic forwarding.
- Double-tagged 802.1ad services require a valid tagging protocol encapsulation (802.1ad or 802.1Q) configured on the physical, bundle, or subinterface.
- Double-tagged services are often used to provide service segregation for different customers or applications.
- Incorrect VLAN or encapsulation configurations may prevent traffic from passing through the subinterface.

L2 VLAN subinterface encapsulation and rewrite

Details encapsulation and rewrite options for L2 VLAN subinterfaces, including configuring exact matching for single-tagged and legacy QinQ encapsulation, handling priority-tagged traffic, and reviewing supported encapsulation types.

L2 VLAN subinterface encapsulation and rewrite is a Layer 2 VLAN capability that

- matches single-tagged Ethernet frames with exact VLAN criteria
- supports legacy QinQ encapsulation with 0x9100 and 0x8100 ethertypes as well as priority-tagged traffic, and
- expands subinterface encapsulation and tag-rewrite behavior for more flexible VLAN handling.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
L2 VLAN Subinterface Encapsulation and Rewrite	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
L2 VLAN Subinterface Encapsulation and Rewrite	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8404-SYS-D router.
L2 VLAN Subinterface Encapsulation and Rewrite	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D
L2 VLAN Subinterface Encapsulation and Rewrite	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
L2 VLAN Subinterface Encapsulation and Rewrite	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now use the VLAN Subinterface Encapsulation and Rewrite operations to: • Configure exact matching for all single-tagged encapsulations. • Support legacy Q-in-Q encapsulation 0x9100/0x8100. • Enable priority tagged traffic to map to the specified interface. The feature introduces these changes: **CLI:** • **dot1q tunneling ethertype 0x9100** • **hw-module profile encap-exact** • **encapsulation dot1ad priority-tagged** • **encapsulation dot1q priority-tagged** • **rewrite ingress tag** **YANG Data Model:** • `Cisco-IOS-XR-um-8000-hw-module-profile-cfg` (see GitHub, YANG Data Models Navigator)

Configure exact matching for single-tagged encapsulations

Configure exact matching so double-tagged traffic does not match single-tagged encapsulations on Layer 2 subinterfaces.

Set up exact matching for all single-tagged encapsulations on the specified interface, interface type, or location to prevent double-tagged traffic from incorrectly matching single-tagged subinterfaces.

The **hw-module profile encap-exact** command applies only to single-tagged encapsulations and prevents double-tagged traffic from matching single-tagged subinterfaces. This configuration ensures that double-tagged traffic will not match single-tagged subinterfaces on Layer 2 interfaces. To activate this profile, you must reload the affected line card or router.

Before you begin

Decide whether to apply the profile to all interfaces, a device or line card, all bundle interfaces, or a specific interface.

Procedure

1. Enter configuration mode and apply the exact-matching profile.

Example

```
Router# configure
Router(config)# hw-module profile encap-exact location all

Router# configure
Router(config)# hw-module profile encap-exact location <node-id>

Router# configure
Router(config)# hw-module profile encap-exact location all-virtual

Router# configure
Router(config)# hw-module profile encap-exact interface <intf>
```

2. Reload the affected chassis or line cards so the profile takes effect.

   Running configuration:

```
configure
hw-module profile encap-exact location all
Wed Nov 20 16:30:52.007 EST
In order to activate/deactivate this profile, you must manually reload the
chassis/all line cards
```

3. Use the **show hw-module profile encap-exact** command to verify the profile state before and after reload.

   Example

```
/* Before Reload */
Router# show hw-module profile encap-exact
Encap Exact                   Configured      No        Reload

/* After Reload */
Router# show hw-module profile encap-exact
Encap Exact                   Configured      Yes       None
```

Configure legacy QinQ encapsulation 0x9100/0x8100

Configure legacy QinQ encapsulation so the outer EtherType uses 0x9100 instead of 0x8100 for Dot1Q second-dot1q services in Layer 2 VPN deployments.

Set up legacy QinQ encapsulation with outer EtherType 0x9100 for Dot1Q tunneling, enabling compatibility with older systems and specific service requirements.

This task configures legacy QinQ encapsulation on Cisco IOS XR interfaces. The **dot1q tunneling ethertype 0x9100** command ensures that Dot1Q second-dot1q services use 0x9100 as the outer EtherType instead of the default 0x8100.

Before you begin

Identify the main interface and the outer and inner VLAN IDs for the service.

Procedure

1. Configure legacy QinQ encapsulation on the interface and subinterface.

Example

```
Router# configure
Router(config)# interface FH0/0/0/3
Router(config-subif)# dot1q tunneling ethertype 0x9100
Router(config-subif)# interface FH0/0/0/3.1 l2transport
Router(config-subif)# encapsulation dot1q 500 second-dot1q 600
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# exit
```

Running configuration:

```
configure
interface FH0/0/0/3
    dot1q tunneling ethertype 0x9100
interface FH0/0/0/3.1 l2transport
    encapsulation dot1q 500 second-dot1q 600
```

2. Verify the running configuration and active encapsulation.

Example

```
Router# show run int FH0/0/0/3.1
interface FourHundredGigE0/0/0/3.1 l2transport
encapsulation dot1q 500 second-dot1q 600

Router# show run int FH0/0/0/3
interface FourHundredGigE0/0/0/3
dot1q tunneling ethertype 0x9100

Router# show int FH0/0/0/3.1
FourHundredGigE0/0/0/3.1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 0075.f409.6818
  Layer 2 Transport Mode
  MTU 1522 bytes, BW 400000000 Kbit (Max: 400000000 Kbit)
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation 802.1Q Virtual LAN,
    Outer Match: Dot1Q VLAN 500
    Inner Match: Dot1Q VLAN 600
    Ethertype Any, MAC Match src any, dest any
  loopback not set,
  Last link flapped 00:00:34
  Last input never, output never
  Last clearing of "show interface" counters never
     0 packets input, 0 bytes
     0 input drops, 0 queue drops, 0 input errors
     0 packets output, 0 bytes
     0 output drops, 0 queue drops, 0 output errors

Router# show int FH0/0/0/3
FourHundredGigE0/0/0/3 is up, line protocol is up
  Interface state transitions: 1
  Hardware is FourHundredGigE, address is 0075.f409.6818 (bia 0075.f409.6818)

  Internet address is Unknown
  MTU 1514 bytes, BW 400000000 Kbit (Max: 400000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 400000Mb/s, link type is force-up
```

```
  output flow control is off, input flow control is off
  Carrier delay (up) is 200 msec
  loopback not set,
  Last link flapped 00:00:53
  Last input never, output never
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
```

Configure priority-tagged traffic on a VLAN subinterface

Configure a VLAN subinterface so priority-tagged traffic with VLAN ID 0 maps to the specified interface.

Map priority-tagged traffic to a Layer 2 VLAN subinterface to enable proper handling of packets with VLAN ID 0.

Priority-tagged traffic uses VLAN ID 0. This configuration ensures that such traffic is correctly mapped to the intended Layer 2 interface for handling and transport.

Procedure

1. Configure priority-tagged encapsulation on the Layer 2 subinterface.

   Example

```
Router# configure
Router(config)# interface TenGigE0/0/0/0.1 l2transport
Router(config-subif)# encapsulation dot1q priority-tagged
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# exit
```

2. Use the **show int TenGigE0/0/0/0.1** command to verify that the priority-tagged encapsulation is active.

   Example

```
Router# show int TenGigE0/0/0/0.1
TenGigE0/0/0/0.1 is up, line protocol is up
Layer 2 Transport Mode
Encapsulation 802.1Q priority tagged,
  Outer Match: Dot1Q VLAN Priority-tagged
```

Supported encapsulation modes for Layer 2 interfaces

This reference lists the encapsulation modes supported for Layer 2 interfaces, subinterfaces, and bundle subinterfaces.

Table 4: 802.1ad encapsulation support for Layer 2 interfaces and subinterfaces

Interface type	Encapsulation class	Encapsulation	Support status
Layer 2 interface, Layer 2 subinterface, and Layer 2 bundle subinterface	Single-tag encapsulation	`dot1ad`	Supported from Cisco IOS XR Release 24.4.1 onward.
Layer 2 interface, Layer 2 subinterface, and Layer 2 bundle subinterface	Single-tag encapsulation	`dot1q`	Supported.
Layer 2 interface, Layer 2 subinterface, and Layer 2 bundle subinterface	Single-tag encapsulation	`default`	Supported.
Layer 2 interface, Layer 2 subinterface, and Layer 2 bundle subinterface	Single-tag encapsulation	`untagged`	Supported.
Layer 2 interface, Layer 2 subinterface, and Layer 2 bundle subinterface	Single-tag encapsulation	`dot1q priority-tagged`	Supported from Cisco IOS XR Release 24.4.1 onward.
Layer 2 interface, Layer 2 subinterface, and Layer 2 bundle subinterface	Single-tag encapsulation	`dot1ad priority-tagged`	Supported from Cisco IOS XR Release 24.4.1 onward.
Layer 2 interface, Layer 2 subinterface, and Layer 2 bundle subinterface	Double-tag encapsulation	`dot1ad <> dot1q <>`	Supported.
Layer 2 interface, Layer 2 subinterface, and Layer 2 bundle subinterface	Double-tag encapsulation	`dot1q <> dot1q <>`	Supported on Q200-based line cards from Cisco IOS XR Release 24.1.1 onward and supported for all Cisco 8000 Series Routers from Cisco IOS XR Release 24.4.1 onward.

Encapsulation default catch-all mode

Explains how catch-all matching maps unmatched and untagged traffic to a Layer 2 service and identifies the supported behavior, configuration, and rewrite options.

Encapsulation default catch-all mode is a Layer 2 (L2) attachment circuit (AC) traffic-matching mode that maps unmatched VLAN-tagged traffic and untagged traffic to an encapsulation default subinterface.

Without the catch-all hardware profile enabled, an encapsulation default L2 AC matches only VLAN-tagged traffic that does not match another subinterface encapsulation. With the catch-all hardware profile enabled, the L2 AC also matches untagged traffic.

Table 5: Feature History Table

Feature name	Release information	Feature description
Encapsulation default catch-all mode	Release 26.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100, P100], 8210 [ASIC: P100]); Centralized Systems (8400); Modular Systems (8800 [LC ASIC: K100, P100]) This feature enables an encapsulation default Layer 2 subinterface to match untagged traffic in addition to unmatched VLAN-tagged traffic. This catch-all capability allows the matched traffic to be carried over VPWS or VPLS services through a single subinterface.

Operational capabilities and limitations of encapsulation default catch-all mode

Lists traffic-matching behavior, supported VLAN rewrite operations, protocol handling, quality-of-service behavior, dependencies, and restrictions for encapsulation default catch-all mode on Cisco 8000 routers.

Use this information to determine whether catch-all mode supports the required Layer 2 service behavior.

The hardware profile state determines which traffic types map to the encapsulation default subinterface.

Table 6: Traffic mapping by hardware profile state

Traffic type	Catch-all profile enabled	Catch-all profile disabled
VLAN-tagged traffic that does not match another subinterface	Mapped	Mapped
Untagged traffic	Mapped	Not mapped

Catch-all L2 ACs support only these symmetric ingress rewrite operations:

- **rewrite ingress tag push dot1q** *vlan-id* **symmetric**
- **rewrite ingress tag push dot1q** *outer-vlan-id* **second-dot1q** *inner-vlan-id* **symmetric**
- **rewrite ingress tag push dot1ad** *vlan-id* **symmetric**
- **rewrite ingress tag push dot1ad** *outer-vlan-id* **dot1q** *inner-vlan-id* **symmetric**

Related feature and protocol support

Encapsulation default catch-all L2 attachment circuits support:

- Connectivity fault management (CFM) on main interfaces and subinterfaces
- Slow-path handling of untagged Link Aggregation Control Protocol (LACP), Link Layer Discovery Protocol (LLDP), and Cisco Discovery Protocol (CDP) packets
- QoS functions, including classification, remarking, policing, and queueing

Restrictions of catch-all mode

- Catch-all mode is supported only on L2 ACs. It is not supported on Layer 3 ACs.

- Priority code point and drop eligible indicator remarking does not change tagged traffic unless a VLAN push operation adds a tag.

Configure encapsulation default catch-all mode

Enables the encapsulation default catch-all hardware profile, reloads the affected chassis, and verifies that the profile is applied.

Enable the hardware profile that supports catch-all traffic matching on encapsulation default interfaces.

Use catch-all mode when one L2 attachment circuit must accept untagged traffic and VLAN-tagged traffic that does not match another subinterface.

Before you begin

Configure a physical or bundle main interface that supports an L2 subinterface.

Procedure

1. Enter global configuration mode and enable the encapsulation default catch-all hardware profile.

   Example

   ```
   Router# configure
   Router(config)# hw-module profile interface encap-default catch-all
   Router(config)# commit
   ```

   The router displays a message indicating that you must reload the chassis to apply the updated profile:

   ```
   Hw-module profile config changed for "interface encap-default catch-all", do
    chassis reload to get the most recent config updated
   ```

2. Reload the affected chassis for the profile to take effect.
3. Verify that the encapsulation default catch-all profile is configured and applied.

   Example

   ```
   Router#show hw-module profile interface
   ------------------------------------------------------------------------
   Knob                                    Status          Applied   Action

   ------------------------------------------------------------------------
   Encap-Default Catch-All                 Configured      Yes       None
   ```

   Confirm that the **Encap-Default Catch-All** row shows **Configured** in the Status column and **Yes** in the Applied column.

The encapsulation default catch-all hardware profile is enabled and active on the affected chassis.

VLAN subinterfaces

Explains VLAN subinterface fundamentals, outlines the use of VLAN lists and ranges, presents guidelines for high-density subinterface deployment, and demonstrates configuration procedures for VLAN subinterfaces.

A VLAN subinterface is a Layer 2 subinterface that

- classifies ingress traffic by encapsulation and VLAN identifiers
- supports both basic dot1q and double-tagged dot1ad and dot1q attachment circuits, and
- maps matching frames to the correct Layer 2 service instance.

VLAN subinterface characteristics

Provides key information on VLAN subinterfaces, including their naming format, encapsulation requirements, and maximum transmission unit inheritance.

Key VLAN subinterface attributes

VLAN subinterfaces are logical interfaces created on a hardware interface. These software-defined interfaces enable separation of traffic into distinct logical channels on a single physical port.

- Naming format: Subinterfaces are named by appending an extension to the base interface name (for example, `HundredGigE 0/1/0/0.23` for subinterface 23 on physical interface `HundredGigE 0/1/0/0`).
- Encapsulation requirement: Each subinterface must have a valid tagging protocol encapsulation and VLAN identifier assigned before it can pass traffic. Ethernet subinterfaces default to 802.1Q VLAN encapsulation, but require an explicit VLAN ID.
- Maximum transmission unit (MTU): The MTU is inherited from the physical interface, with an additional 4 bytes for the 802.1Q VLAN tag.
- Attachment circuit mode: The basic dot1q Attachment Circuit mode of VLAN subinterface configuration is supported through the command `encapsulation dot1q vlan_id`.
- Layer 2 encapsulation: Layer 2 subinterfaces can be configured with **encapsulation default** command.

```
configure
 interface HundredGigE 0/0/0/10.1
  l2transport
   encapsulation default
```

VLAN list and VLAN range

Lists VLAN list and VLAN range notation and the encapsulation modes that support them.

VLAN notation

- VLANs separated by commas form a VLAN list.
- VLANs separated by dashes form a VLAN range.

For more information about VLAN list and VLAN range, see *L2 Interface VLAN Encapsulation using VLAN Range and List*.

Starting from Cisco IOS XR Software Release 24.4.1 onwards, VLAN list and VLAN range are supported for these encapsulation types:

- **encapsulation dot1ad**
- **encapsulation dot1ad dot1q**
- **encapsulation dot1q**
- **encapsulation dot1q second-dot1q**

Best practice for high-density subinterface deployment

Outlines best practices for deploying high-density Layer 2 VLAN subinterfaces to maximize hardware utilization and prevent Out of Resource conditions.

To avoid uneven hardware utilization and Out of Resource (OOR) conditions in dense Layer 2 VLAN subinterface designs, follow these best practices:

- Configure Layer 2 VLAN subinterfaces across multiple physical interfaces to improve resource utilization and ensure service mappings are not concentrated on a single hardware slice.
- For high-density deployments, distribute services across different interfaces to prevent uneven allocation of system resources. Use bundle interfaces to distribute services across multiple member links, improving load balancing and resiliency.
- Prefer interfaces outside the fabric slice for very high-density deployments to optimize resource allocation.
- Be aware that the system automatically assigns service mapping resources using internal hashing mechanisms, which can result in uneven resource distribution.
- Monitor for Out of Resource (OOR) conditions; these may occur if resources are unevenly allocated, even when overall capacity remains available, and monitoring tools may report maximum utilization for affected resources.

Configure VLAN subinterface

Create a VLAN subinterface, enable Layer 2 transport mode, and define the encapsulation used to match ingress traffic.

Use this task to configure either a basic dot1q attachment circuit or a double-tagged dot1ad and dot1q attachment circuit.

The source includes both single-tag and double-tag examples and shows how to verify each mode.

Before you begin

Identify the physical interface, subinterface number, and VLAN IDs that the service requires.

Procedure

1. Create the Layer 2 subinterface.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/10.1 l2transport
   ```

2. Configure encapsulation for the required service type.

   Example

   ```
   Router(config-if)# encapsulation dot1q 10
   ```

   For a double-tag service, the source uses:

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/10.1 l2transport
   Router(config-if)# encapsulation dot1ad 200 dot1q 201
   ```

3. Bring the subinterface up.

Example

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

4. Use the **show interfaces hundredGigE 0/0/0/29.300** command to verify that the VLAN subinterface is active in a double-tag scenario.

   Example

```
Router# show interfaces hundredGigE 0/0/0/29.300
HundredGigE0/0/0/29.300 is up, line protocol is up
Interface state transitions: 1
Hardware is VLAN sub-interface(s), address is 0029.c201.3f0c
Internet address is Unknown
MTU 1518 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation 802.1Q Virtual LAN, VLAN Id 300, loopback not set,
Last link flapped 00:00:19
Last input never, output never
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
0 packets input, 0 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 0 broadcast packets, 0 multicast packets
0 packets output, 0 bytes, 0 total output drops
Output 0 broadcast packets, 0 multicast packets
```

5. Use the **show interface HundredGigE 0/0/0/29.200** command to verify that the VLAN subinterface is active in a double-tag scenario.

   Example

```
Router# show interface HundredGigE 0/0/0/29.200
HundredGigE0/0/0/29.200 is up, line protocol is up
Interface state transitions: 1
Hardware is VLAN sub-interface(s), address is 0029.c201.3f0c
Internet address is 40.40.50.1/24
MTU 1522 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation 802.1ad-802.1Q Virtual LAN,
Last link flapped 00:01:25
ARP type ARPA, ARP timeout 04:00:00
Last input never, output never
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
0 packets input, 0 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 0 broadcast packets, 0 multicast packets
0 packets output, 0 bytes, 0 total output drops
Output 0 broadcast packets, 0 multicast packets
```

Successful verification shows the interface up and the encapsulation line for each scenario.

Ethernet flow points

Describes Ethernet Flow Point features, including classification, processing, and benefits, identifies EFP frames, outlines feature application and encapsulation modifications, provides VLAN header rewrite and configuration procedures, and explains EFP visibility and data-forwarding behavior.

An Ethernet Flow Point is a Layer 2 logical subinterface that

- classifies ingress traffic with filters that match zero, one, or two VLAN tags
- applies VLAN rewrite, QoS, and forwarding behavior to matching traffic, and
- operates under a physical interface or a bundle interface.

The source uses these terms interchangeably: VLAN AC, L2 interface, and EFP.

Feature history
Table 7: Feature History Table

Feature Name	Release Information	Feature Description
Ethernet Flow Point	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Ethernet Flow Point	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is now supported on the Cisco 8404-SYS-D routers.
Ethernet Flow Point	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The Ethernet Flow Point functionality is now extended to the Cisco 8712-MOD-M routers.
Ethernet Flow Point	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) * The Ethernet Flow Point functionality is now extended to these fixed systems and line cards: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
Ethernet Flow Point	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) An Ethernet Flow Point (EFP) enhances traffic management and network efficiency by classifying and processing Layer 2 traffic under a physical or bundle interface based on specific filters, such as VLAN tags. This logical sub-interface allows for precise traffic classification and manipulation, including changing VLAN IDs, adding or removing VLAN tags, and modifying ethertypes upon ingress. *This functionality is now extended to routers with the 88-LC1-36EH line cards.

Ethernet flow point features

Lists the key features and handling actions supported by Ethernet flow points for classifying and processing ingress traffic.

- Ethernet flow points (EFPs) are Layer 2 logical sub-interfaces for physical or bundle interfaces.
- EFPs classify ingress traffic using configurable filters, called entries, based on VLAN tags (0, 1, or 2 tags).
- Matching criteria:
  - Single VLAN tag
  - QinQ double tagging
- When a packet matches an EFP filter, the EFP applies frame handling actions:
  - changes VLAN IDs
  - adds or removes VLAN tags
  - changes Ethertypes
- After classification, additional actions like frame manipulation and Quality of Service (QoS) can be applied.

> **Note**
>
> These terms are used interchangeably throughout this document:
> - VLAN AC
> - L2 interface
> - EFP

Ethernet flow point operational capabilities

Provides a summary of the key operational capabilities of Ethernet Flow Points in Cisco IOS XR Layer 2 services.

- Identify all frames that belong to a particular flow on a given interface.

- Perform VLAN header rewrites to accommodate different service requirements.
- Add features to the identified frames as required by the service.
- Optionally define forwarding actions for identified frames within the data path.

Frame identification methods for an EFP

Provides information on how EFPs identify frames using VLAN tagging in the Ethernet header and outlines the limitations for classifying traffic beyond that boundary.

Key facts about EFP frame identification

Ethernet Flow Points (EFPs) identify frames belonging to a particular flow on a specific port by examining fields within the Ethernet frame header, independent of higher-level payload encapsulation. EFPs flexibly map frames into flows based on VLAN tags in the outer Ethernet header.

- An EFP can match frames by one or two outer VLAN tags.
- Frame matching is independent of encapsulation beyond the Ethernet header.
- Frames cannot be matched to an EFP using information outside the outermost Ethernet frame header and its tags.

Table 8: VLAN tag identification

Encapsulation type	EFP identifier rule
Single outer-most tag	The tag must use EtherType 0x8100 or 0x88A8.
Two outer-most tags	The outer-most tag can use 0x8100, 0x9100, or 0x88A8 depending on platform or configuration, and the second outer-most tag must use 0x8100.
Unsupported match inputs	The EFP does not classify by IPv4, IPv6, or MPLS tag header data, or by C-DMAC, C-SMAC, or C-VLAN information outside the outermost Ethernet frame header and tags.

Apply features

After frames are matched to an EFP, the router can apply frame manipulation and service features such as QoS and VLAN header rewrite.

Applied features are Layer 2 service behaviors that

- apply QoS and other configured policies after a frame matches an EFP
- modify VLAN header encapsulation on ingress and egress, and
- use symmetric rewrite behavior for the supported actions.

For more information about QoS polices that are supported on L2 interfaces, see *Modular QoS Configuration Guide for Cisco 8000 Series Routers*.

Features applied after EFP matching

This topic explains the types of features that can be applied after traffic matches an Ethernet Flow Point.

After the frames are matched to a particular EFP, any appropriate features can be applied. In this context, features means any frame manipulations specified by the configuration. For example, QoS. You can apply QoS policy on Layer 2 interfaces.

The Layer 2 header encapsulation modification is the Layer 2 interface VLAN tag rewrite that is applied on an EFP as part of the Ethernet infrastructure.

Encapsulation modifications for EFP

Lists the available VLAN header encapsulation modifications that Ethernet Flow Points (EFPs) can apply to packet ingress and egress, including tag manipulation and EtherType selection for VLAN IDs depending on tunneling configuration.

EFP supports these Layer 2 header encapsulation modifications on both ingress and egress:

- Push 1 VLAN tag: push an EtherType 0x8100 or 0x88A8 VLAN tag at ingress, and pop the top VLAN tag at egress.
- Push 2 VLAN tags: push an outer EtherType 0x8100 or 0x9100, depending on tunneling configuration, or 0x88A8 VLAN tag and an inner EtherType 0x8100 VLAN tag at ingress, and pop the top two VLAN tags at egress.
- Pop 1 VLAN tag: pop the top VLAN tag at ingress, and push an EtherType 0x8100 VLAN tag or EtherType 0x88A8 VLAN tag at egress.
- Pop 2 VLAN tags: pop the top two VLAN tags at ingress, and push an inner EtherType 0x8100 VLAN tag and an outer EtherType 0x8100 or 0x9100, depending on tunneling configuration, or 0x88A8 VLAN tag at egress.
- Rewrite one or two VLAN tags, including rewriting the outer tag, rewriting the outer two tags, translating one outer tag and pushing an additional outer VLAN tag, and popping the outer tag while rewriting the second outer VLAN tag.

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

For each of the VLAN ID manipulations, you can specify EtherType 0x88A8, 0x8100, or 0x9100 depending on the tunneling configuration.

Valid ingress rewrite actions

Supports a defined set of ingress rewrite operations and encapsulation types for Layer 2 subinterfaces and EFPs.

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

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

> **Note**
>
> Prior to Cisco IOS XR Software Release 7.8.1, the EtherType cannot be changed at 1-to-1 and 2-to-2 VLAN tag translation operations.

The following encapsulation types are supported:

- encapsulation dot1q <x>
- encapsulation dot1q priority-tagged (Supported from Cisco IOS XR Software Release 24.4.1 onwards)
- encapsulation dot1ad <x> (Supported from Cisco IOS XR Software Release 24.4.1 onwards)
- encapsulation dot1ad priority-tagged (Supported from Cisco IOS XR Software Release 24.4.1 onwards)
- encapsulation dot1ad <x> dot1q <y>

- encapsulation dot1q <x> second-dot1q <y> (Supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.)
- dot1q tunneling ethertype 0x9100 (Supported from Cisco IOS XR Software Release 24.4.1 onwards)
- hw-module profile encap-exact (Supported from Cisco IOS XR Software Release 24.4.1 onwards)

The following lists the supported L2 sub-interface rewrite actions:

- rewrite ingress tag push dot1q <x> symmetric
- rewrite ingress tag push dot1ad <x> symmetric
- rewrite ingress tag push dot1ad <x> dot1q <y> symmetric
- rewrite ingress tag push dot1q <x> second-dot1q <y> symmetric (Supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.)
- rewrite ingress tag pop 1 symmetric
- rewrite ingress tag pop 2 symmetric
- rewrite ingress tag translate 1-to-1 dot1q <x> symmetric
- rewrite ingress tag translate 1-to-1 dot1ad <x> symmetric
- rewrite ingress tag translate 1-to-2 dot1ad <x> dot1q <y> symmetric
- rewrite ingress tag translate 1-to-2 dot1q <x> second-dot1q <y> symmetric (Supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.)
- rewrite ingress tag translate 2-to-1 dot1q <x> symmetric
- rewrite ingress tag translate 2-to-1 dot1ad <x> symmetric
- rewrite ingress tag translate 2-to-2 dot1ad <x> dot1q <y> symmetric
- rewrite ingress tag translate 2-to-2 dot1q <x> second-dot1q <y> symmetric (Supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.)

Configure VLAN header rewrite on Layer 2 subinterfaces

Configure VLAN header rewrite by creating Layer 2 subinterfaces, defining encapsulation, and applying ingress rewrite actions to match a specified service design.

Enable VLAN header rewrite for single-tagged or double-tagged Layer 2 subinterfaces.

This task allows you to adjust the VLAN headers on ingress frames, ensuring that Layer 2 services meet network design requirements through precise encapsulation and rewrite actions. Typical scenarios include mapping single- or double-tagged frames to specific service instances.

Before you begin

Create the subinterface and decide which ingress rewrite action is required.

Procedure

1. Create the Layer 2 subinterface and define encapsulation.

Example

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/0.1 l2transport
Router(config-subif)# encapsulation dot1q 10
```

2. Apply the required rewrite behavior.

Example

```
Router(config-subif)# rewrite ingress tag push dot1q 20 symmetric
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# rewrite ingress tag translate 1-to-1 dot1q 20 symmetric
```

For double-tagged services:

```
interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1ad 2 dot1q 1
  rewrite ingress tag pop 2 symmetric
 !
!

interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1ad 2 dot1q 1
  rewrite ingress tag translate 1-to-2 dot1ad 2 dot1q 1 symmetric
 !
!
```

3. Commit the configuration and review the running configuration.

Example

```
Router(config-subif)# commit
```

The running configuration for the double-tagged examples:

```
/* Configuration without rewrite */
interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1ad 2 dot1q 1
 !
!

/* Configuration with rewrite */

/* POP 2 */
interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1ad 2 dot1q 1
  rewrite ingress tag pop 2 symmetric
 !
!

/* TRANSLATE 1-2 */
interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1ad 2 dot1q 1
  rewrite ingress tag translate 1-to-2 dot1ad 2 dot1q 1 symmetric
 !
!
```

Data-forwarding behaviors

Explains how data-forwarding behaviours enable Ethernet Frame Processing (EFP) to identify and forward frames based on configured Layer 2 service models.

A data-forwarding behaviour is a Layer 2 service forwarding method that

- uses the EFP to identify the frames that belong to a specific Ethernet flow
- maps those frames to a bridge domain for switching by destination MAC address, and
- supports multipoint Ethernet-to-Ethernet bridging services.

Forwarding cases for EFP

Lists the types of forwarding supported for EFPs.

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

- Layer 2 switched service (bridging): The EFP is mapped to a bridge domain, where frames are switched based on their destination MAC address.
- Multipoint Ethernet-to-Ethernet bridging: Frames are forwarded across multiple Ethernet segments in a multipoint configuration.

Ethernet flow point visibility

Explains how EFP visibility lets you configure multiple VLANs in the same bridge domain and add more EFPs within a bridge group.

Ethernet flow point visibility is an operational capability that

- allows multiple VLAN-based service instances to coexist within the same bridge domain
- uses one or two VLAN tags to identify each EFP service instance, and
- lets you add multiple EFPs within a bridge group regardless of the number of available ports.

EFP visibility fundamentals

Provides key facts about EFP visibility, enabling support for multiple VLAN service instances within the same bridge domain.

EFP visibility enables you to configure multiple VLANs in the same bridge domain by providing these features:

- Multiple VLANs in a bridge domain: Allows you to configure more than one VLAN within a single bridge-domain for flexible service delivery.
- Ethernet Flow Point (EFP) service instances: Logical interfaces that connect a bridge domain to a physical port or EtherChannel group. EFPs are identified using one or two VLAN tags.
- Flexible EFP addition: Supports adding multiple EFPs to one bridge group, regardless of the number of available ports, allowing for expanded networking capabilities.

Configure Ethernet flow point interfaces

Create EFP interfaces, define VLAN matching and rewrite behavior, and attach the EFPs to bridge domains.

Use this task to configure VLAN interfaces under bridge domains by using multiple EFPs.

This example shows how to configure VLAN interfaces under a bridge domain with multiple EFPs. To configure an EFP, the source explicitly uses three interface configuration commands:

- **l2transport** to identify the interface as an EFP
- **encapsulation** to define VLAN matching criteria
- **rewrite** to define VLAN tag rewrite criteria

Before you begin

Know which VLAN IDs map to each EFP and which EFPs belong to each bridge domain.

Procedure

1. Create the EFP interfaces and configure VLAN matching.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE0/0/0/4.1 l2transport
   Router(config-subif)# encapsulation dot1q 1
   Router(config-subif)# rewrite ingress tag pop 1 symmetric
   Router(config-subif)# exit
   ```

   Repeat the source sequence for the other EFPs:

   ```
   Router(config)# interface HundredGigE0/0/0/4.2 l2transport
   Router(config-subif)# encapsulation dot1q 2
   Router(config-subif)# rewrite ingress tag pop 1 symmetric
   Router(config-subif)# exit
   Router(config)# interface HundredGigE0/0/0/5.1 l2transport
   Router(config-subif)# encapsulation dot1q 3
   Router(config-subif)# rewrite ingress tag pop 1 symmetric
   Router(config-subif)# exit
   Router(config)# interface HundredGigE0/0/0/5.2 l2transport
   Router(config-subif)# encapsulation dot1q 4
   Router(config-subif)# rewrite ingress tag pop 1 symmetric
   Router(config-subif)# exit
   ```

2. Create the first bridge domain and add the required EFP interfaces.

   Example

   ```
   Router(config)# l2vpn
   Router(config-l2vpn)# bridge group bg1
   Router(config-l2vpn-bg)# bridge-domain bd1
   Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/4.1
   Router(config-l2vpn-bg-bd-ac)# exit
   Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/5.1
   Router(config-l2vpn-bg-bd-ac)# exit
   ```

3. Create the second bridge domain and add the remaining EFP interfaces.

Example

```
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# bridge group bg2
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/4.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/5.2
```

4. Commit the configuration.

Example

```
Router(config-l2vpn-bg-bd-ac)# commit
```

5. Use the **show interfaces HundredGigE0/0/0/4.2** to verify the interface and bridge-domain state.

Example

```
Router# show interfaces HundredGigE0/0/0/4.2
HundredGigE0/0/0/4.2 is up, line protocol is up
  Interface state transitions: 101
  Hardware is VLAN sub-interface(s), address is c4b2.39da.1620
  Layer 2 Transport Mode
  MTU 1518 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation 802.1Q Virtual LAN,
    Outer Match: Dot1Q VLAN 2
    Ethertype Any, MAC Match src any, dest any
  loopback not set,
  Last link flapped 2d10h
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters 3d18h
     21364536641 packets input, 2734660346522 bytes
     0 input drops, 0 queue drops, 0 input errors
     8420820982 packets output, 1077864630044 bytes
     0 output drops, 0 queue drops, 0 output errors

Router# show l2vpn bridge-domain summary
Number of groups: 2, VLAN switches: 0
Number of bridge-domains: 510, Up: 510, Shutdown: 0, Partially-
programmed: 0
Default: 510, pbb-edge: 0, pbb-core: 0
Number of ACs: 1530 Up: 1275, Down: 255, Partially-programmed: 0
Number of PWs: 0 Up: 0, Down: 0, Standby: 0, Partially-programmed: 0
Number of P2MP PWs: 0, Up: 0, Down: 0, other-state: 0
Number of VNIs: 0, Up: 0, Down: 0, Unresolved: 0
```

5 Transparent Layer 2 Protocol Tunneling

Topics:

- Transparent Layer 2 protocol tunneling

Introduces Transparent Layer 2 protocol tunneling, outlining requirements, supported protocols, protocol handling mechanisms, and verification steps to enable efficient Layer 2 protocol transport across network boundaries.

Transparent Layer 2 protocol tunneling

Describes Transparent Layer 2 protocol tunneling, detailing requirements for deployment, supported protocols, protocol handling processes, and provides instructions to verify tunneling functionality in network environments.

Transparent Layer 2 protocol tunneling is a Layer 2 service that

- tunnels Layer 2 protocol data units (PDUs) across the core network without being interpreted and processed by intermediary network devices
- forwards any packet on the L2 network without any change, and
- is enabled by default.

Feature History Table
Table 9: Feature History Table

Feature Name	Release Information	Feature Description
Transparent Layer 2 Protocol Tunneling	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Transparent Layer 2 Protocol Tunneling	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Transparent Layer 2 Protocol Tunneling	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The Layer 2 protocol tunneling functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E • 88-LC1-36EH • 8712-MOD-M

Feature Name	Release Information	Feature Description
Transparent Layer 2 Protocol Tunneling	Release 7.3.2	This feature allows Layer 2 protocol data units (PDUs) to be kept intact and delivered across the service-provider network to the other side of the customer network. Such delivery is transparent because the VLAN and Layer 2 protocol configurations are maintained throughout. With this feature, service providers can send traffic from multiple customers across a core network without impacting the traffic of other customers. This feature is enabled by default.

Requirements for transparent Layer 2 protocol tunneling

Outlines the requirements for correct interface and protocol configuration for transparent Layer 2 protocol tunneling.

Follow these requirements to ensure transparent Layer 2 protocol tunneling operates correctly:

- Configure supported protocols only on main and bundle interfaces.
- To punt specific protocol packets over bundle members or subinterfaces, enable the protocol on the main interface as well.
- For CFM and PVRST protocols, enable these protocols on a subinterface.

Supported protocols for transparent Layer 2 protocol tunneling

Lists the Layer 2 control and operations protocols supported by transparent Layer 2 protocol tunneling. This reference enables you to verify which protocols can be tunneled transparently and provides guidance for planning network deployments.

Transparent Layer 2 protocol tunneling supports these protocols:

- Link Layer Discovery Protocol (LLDP) and Cisco Discovery Protocol (CDP)
- Multiple Spanning Tree Protocol (MSTP)
- Per-VLAN Rapid Spanning Tree (PVRST)
- Connectivity Fault Management (CFM)
- Link Aggregation Control Protocol (LACP)
- Operation, Administration, Management (OAM)
- Synchronized Ethernet (SyncE)
- MAC security
- Priority Flow Control (PAUSE)

All packets on PW VPLS or VPWS are always tunnelled and no packet is sent to the CPU for processing (punted).

Protocol handling for transparent Layer 2 protocol tunneling

Lists the router behaviors for handling tagged and untagged packets with various protocols enabled or disabled on Layer 2 interfaces.

This table provides details on how the router treats Layer 2 protocol traffic when specific protocols are enabled or disabled on an interface.

L2 Protocol	Untagged Packet	Tagged Packet
Cisco Protocols	If Cisco protocols are enabled on the physical port, the traffic is sent to the CPU for processing. If Cisco protocols are disabled, the traffic is tunneled.	Traffic is always tunneled.
LLDP	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	Traffic is always tunneled.
PVRST/PVRST+	If this protocol is enabled on the main port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	If this protocol is enabled on the subinterface, the traffic is sent to CPU for processing. If it is disabled, the traffic is tunneled.
MSTP	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	Traffic is always tunneled.
CFM	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	If this protocol is enabled on the Xconnect, the traffic is sent to CPU for processing. If it is disabled, the traffic is tunneled.
LACP/SyncE/LOAM	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	Traffic is always tunneled.
PFC	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	Traffic is always tunneled.

Verify transparent Layer 2 protocol tunneling

Use show commands and interface counters to check which protocols are enabled and whether Layer 2 traffic is flooded or forwarded.

Determine which protocols are enabled for each interface and whether Layer 2 packets are flooded or forwarded by the network processing unit (NPU).

Transparent protocol tunneling is always enabled by default and cannot be disabled. This task helps you check protocol state and traffic handling for Layer 2 interfaces.

Procedure

1. Use the **show ofa objects ethport base location 0/1/CPU0** command to display the protocols that are enabled per interface.

   Example

```
Router# show ofa objects ethport base location 0/1/CPU0
ethport element 0 (hdl:0x308f38e360):
  base
  |-- dpd_slf -   pending(cr/up/dl):0/0/0, sibling:0x3093b811c8, child:2,
num_parents:3, parent-trans_id:1523, visits:0
   color_mask:0, last_bwalk_id:0 num_bwalks_started:0
  |-- keylen - 4
  |-- trans_id - 489153
  |-- create_trans_id - 1523
  |-- obj_handle - 0x308f38e360
  |-- flag - 10
  |-- reason - 0
  |-- table_operation - 6
  |-- total_obj_size - 632
  |-- idempotent - 0
  |-- inflight - 0
  |-- table_prop - jid=169 mtime=(GMT)2021.Jan.09 13:05:46.670570
  |--  (cont'd)  - replayed=0times
  `-+ npu_results
    |-- npu0 - 0:Success
    |-- npu1 - 0:Success
    |-- npu2 - 0:Success
    `-- npu3 - 0:Success
  ofa_npu_mask_t npu_mask =>
...
  ofa_bool_t remote_chain_in_use => TRUE
  ofa_bool_t local_chain_in_use => TRUE
  uint8_t copc_profile => 0
  ofa_bool_t lldp_enable => FALSE
  ofa_bool_t slow_proto_enable => FALSE
  ofa_bool_t cdp_enable => (not set)
  ofa_bool_t pvrst_enable => FALSE
  ofa_bool_t mstp_enable => FALSE
  ofa_bool_t macsec_enable => FALSE
  ofa_bool_t mka_enable => FALSE
  ofa_bool_t pfc_enable => FALSE
  ofa_bool_t cfm_enable => FALSE
  dpa_npu_mask_t npu_bmap => (not set)
```

2. Use the **show ofa objects l2if base location 0/1/CPU0** command to display the enabled protocol state for the Layer 2 interface.

Example

```
Router# show ofa objects l2if base location 0/1/CPU0
l2if element 0 (hdl:0x3094ba70a8):
  base
  |-- dpd_slf -   pending(cr/up/dl):0/0/0, sibling:0x308f8087c8, child:1,
num_parents:1, visits:0
   color_mask:0, last_bwalk_id:0 num_bwalks_started:0
  |-- flag - 10
      |-- flag.is_id_alloced - 0x1
  |-- keylen - 4
  |-- trans_id - 18311
  |-- create_trans_id - 18299
  |-- obj_handle - 0x3094ba70a8
  |-- obj_rc - 0x0
  |-- reason - 0
  |-- table_operation - 6
  |-- total_obj_size - 776
  |-- idempotent - 1
  |-- inflight - 0
  |-- table_prop - jid=137 mtime=(GMT)2021.Jun.21 14:53:56.644917
  |--  (cont'd)  - replayed=0times
  `-- obj_rc - 0:Success
  ofa_npu_mask_t npu_mask => 0 (not set)
  uint32_t member_count => 1
 @dpa_intf_t intf => 0x0f00000a
...
ofa_l2vpn_fwd_state_type fwd_state => (not set)
  ofa_bool_t cfm_enable => FALSE
  ofa_bool_t pvrst_enable => TRUE
  dpa_npu_mask_t npu_bmap => 1
```

3. Look at the interface counters to verify whether the L2 packet is flooded or forwarded by NPU.

   In case of flood, like multicast MAC, you will notice an increment in the output counters of the interface. When the traffic is forwarded with unicast MAC, you will notice an increment in the output counters only on the egress interface.

   The following output displays the interface counters:

   Example

```
Router# show interface hundredGigE 0/0/2/0 accounting

HundredGigE0/0/2/0
  Protocol              Pkts In         Chars In     Pkts Out        Chars
Out
  CDP                        0                0       163608         21923472
```

You can identify which protocols are enabled and determine whether the traffic is flooded or forwarded on the interface.

6 Link Bundles for Layer 2 VPNs

Topics:

- Link bundles for Layer 2 VPNs
- Configure an interface link bundle
- Configure a VLAN bundle
- References for configuring link bundles
- LACP compatibility requirements for link aggregates

Outlines the use of link bundles in Layer 2 VPNs, detailing their benefits, configuration procedures for interface and VLAN bundles, link aggregation methods, and providing reference information on bundle characteristics and formation.

Link bundles for Layer 2 VPNs

Introduces link bundles within Layer 2 VPNs, describing their core concepts and architecture and outlining key benefits to network efficiency and reliability.

A link bundle for a Layer 2 VPN is an aggregated Ethernet link that

- groups one or more ports together and treats them as a single link
- uses a single MAC address and a single configuration set such as ACLs or QoS, and
- aggregates the bundled ports into one logical link.

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

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

Advantages of link bundles

Lists the main advantages of using link bundles in a network.

Link bundles offer these key advantages:

- Redundancy: Bundling multiple physical links ensures that the failure of a single link does not cause loss of connectivity.
- Increased bandwidth: Traffic is distributed across all bundle members, aggregating the capacity of individual ports for greater overall throughput.

Configure an interface link bundle

Provides step-by-step instructions for configuring an interface link bundle, guiding users through setup, required parameters, and validation procedures for proper operation.

Configure a link bundle between two routers and verify that the interfaces forming the bundle are active.

Cisco IOS XR software supports the EtherChannel method for bundling Ethernet interfaces. EtherChannel is a Cisco proprietary technology that allows you to configure multiple links in a bundle but does not check link compatibility automatically. IEEE 802.3ad encapsulation uses Link Aggregation Control Protocol (LACP) to ensure all member links in an Ethernet bundle are compatible.

Incompatible or failed links are automatically removed from the bundle.

Before you begin

- Create a bundle instance.
- Map physical interface(s) to the bundle.
- For an Ethernet bundle to be active, you must perform the same configuration on both connection endpoints of the bundle.

Procedure

1. Create the Ethernet link bundle.

Example

```
Router# configure
Router(config)# interface Bundle-Ether 3
Router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
Router(config-if)# bundle maximum-active links 32 hot-standby
Router(config-if)# bundle minimum-active links 1
Router(config-if)# bundle minimum-active bandwidth 30000000
Router(config-if)# exit
```

2. Map the physical interfaces to the bundle.

   Mixed link bundle mode is supported only when active-standby operation is configured.

   Example

```
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# bundle id 3 mode active
Router(config-if)# no shutdown
Router(config)# exit

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

Router(config)# interface HundredGigE 0/0/0/2
Router(config-if)# bundle id 3 mode active
Router(config-if)# no shutdown
Router(config-if)# exit
```

3. Check the running configuration.

   Example

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

interface HundredGigE 0/0/0/1
 bundle-id 3 mode active
!
interface HundredGigE 0/0/0/2
 bundle-id 3 mode active
!
```

4. Use the **show bundle bundle-ether 3** to verify that interfaces forming the bundle are active and the status of the bundle is Up.

Example

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

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

  Port                  Device           State        Port ID         B/W,
kbps
  --------------------  ---------------  -----------  --------------
----------
  Hu0/0/0/0           Local           Active      0x8000, 0x0000    10000000

      Link is Active
  Hu0/0/0/1           Local           Active      0x8000, 0x0000    10000000

      Link is Active
  Hu0/0/0/2           Local           Active      0x8000, 0x0000    10000000

      Link is Active
```

The interface link bundle is configured, and you can verify that the bundle status is Up and the member links are active.

Configure a VLAN bundle

Explains how to configure a VLAN bundle, detailing necessary configuration steps and guidelines to ensure successful deployment within Layer 2 VPN environments.

Set up a VLAN bundle and ensure that the VLAN is operational.

Creating a VLAN bundle enables efficient use of multiple links and organizes traffic by VLAN. The procedure is similar to creating VLAN sub-interfaces on a physical Ethernet interface.

Before you begin

To configure VLAN bundles, complete these configurations:

- Create a bundle instance.
- Create a VLAN interface (bundle sub-interface).
- Map the physical interface(s) to the bundle.
- Ensure that the same configuration is performed on both endpoints of the VLAN bundle.

Procedure

1. Create the VLAN bundle.

   Example

```
Router# configure
Router(config)# interface Bundle-Ether 2
Router(config-if)# ipv4 address 50.0.0.1/24
Router(config-if)# bundle maximum-active links 32 hot-standby
Router(config-if)# bundle minimum-active bandwidth 30000000
Router(config-if)# bundle minimum-active links 1
Router(config-if)# commit
Router(config-if)# exit
```

2. Create the VLAN sub-interface and add it to the bundle.

   Example

```
Router(config)# interface Bundle-Ether 2.201
Router(config-subif)# ipv4 address 12.22.1.1 255.255.255.0
Router(config-subif)# encapsulation dot1q 201
Router(config-subif)# commit
Router(config-if)# exit
```

3. Map the physical interface to the bundle.

   Example

```
Router(config)# interface HundredGigE 0/0/0/14
Router(config-if)# bundle id 2 mode on
Router(config-if)# no shutdown
Router(config-if)# commit
Router(config-if)# exit
```

   Repeat the same mapping for member interfaces 0/0/0/15, 0/0/0/16, and 0/0/0/17 in this example.

4. Check the running configuration.

   Example

```
Router# show running-configuration
configure
interface Bundle-Ether2
 ipv4 address 50.0.0.1 255.255.255.0
 mac-address 1212.1212.1212
 bundle maximum-active links 32 hot-standby
 bundle minimum-active links 1
 bundle minimum-active bandwidth 30000000
!
interface Bundle-Ether2.201
 ipv4 address 12.22.1.1 255.255.255.0
 encapsulation dot1q 201
!
interface HundredGigE0/0/0/14
 bundle id 2 mode on
!
```

```
interface HundredGigE0/0/0/15
 bundle id 2 mode on
!
interface HundredGigE0/0/0/16
 bundle id 2 mode on
!
interface HundredGigE0/0/0/17
 bundle id 2 mode on
!
```

5. Use the **show interfaces bundle-ether 2.201** command to verify that the VLAN status is UP.

   Example

```
Router# show interfaces bundle-ether 2.201

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

The VLAN bundle is configured, and you can verify that the VLAN status is up.

References for configuring link bundles

Details essential reference information for configuring link bundles, including characteristics of link bundles and bundle formation methods to support deployment decisions.

Use these reference topics to review factual information about configuring link bundles.

Characteristics of link bundles

Lists the operational and configuration characteristics supported by Ethernet link bundles.

Ethernet link bundles provide a set of features and behaviors that enhance network flexibility and manageability. Key characteristics include:

- Any type of Ethernet interfaces can be bundled, with or without the use of LACP (Link Aggregation Control Protocol).
- Physical layer and link layer configuration are performed on individual member links of a bundle.
- Configuration of network layer protocols and higher layer applications is performed on the bundle itself.
- A bundle can be administratively enabled or disabled.

- Each individual link within a bundle can be administratively enabled or disabled.
- Ethernet link bundles are created in the same way as Etherokinet channels, where the user enters the same configuration on both end systems.
- The MAC address that is set on the bundle becomes the MAC address of the links within that bundle.
- When LACP configured, each link within a bundle can be configured to allow different keepalive periods on different members.
- Load balancing is done by flow instead of by packet. Data is distributed to a link in proportion to the bandwidth of the link in relation to its bundle.
- QoS is supported and is applied proportionally on each bundle member.
- Link layer protocols, such as CDP, work independently on each link within a bundle.
- Upper layer protocols, such as routing updates and hello messages, are sent over any member link of an interface bundle.
- Bundled interfaces are point to point.
- A link must be in the UP state before it can be in distributing state in a bundle.
- Access Control List (ACL) configuration on link bundles is identical to ACL configuration on regular interfaces.
- Multicast traffic is load balanced over the members of a bundle. For a given flow, internal processes select the member link and all traffic for that flow is sent over that member.

Ethernet bundle configuration modes in Cisco IOS-XR

Lists and compares IEEE 802.3ad (LACP) and EtherChannel methods supported by Cisco IOS-XR software for aggregating Ethernet interfaces.

IEEE 802.3ad (LACP)

Key details

EtherChannel

Cisco IOS-XR software supports multiple bundle configuration modes to aggregate Ethernet interfaces for improved bandwidth and resiliency. The two primary methods are IEEE 802.3ad (LACP) and Cisco EtherChannel.

- Standard technology for Ethernet aggregation.
- Employs Link Aggregation Control Protocol (LACP) to ensure compatibility among member links.
- Automatically removes incompatible or failed links from the bundle.

Each bundle member exchanges these information between router systems:

- A globally unique local system identifier (usually a MAC address).
- An operational key identifying the bundle.
- A port ID for the link.
- The current aggregation status.
- Member links sharing a common Link Aggregation Group (LAG) ID can be aggregated. Each link has a unique LAG ID.
- System and bundle identifiers must be unique per router.

- Peer system information determines link compatibility.
- Bundle MAC addresses are assigned from reserved sets and persist for the life of the bundle, used by all member links for traffic. Unicast or multicast addresses configured on the bundle are propagated to all member links.
- Cisco proprietary aggregation technology.
- Allows manual link configuration into bundles.
- Does not perform compatibility checks between bundled links.

Table 10: Summary table

Mode	Compatibility Checks	Automatic Link Removal	Protocol Used	Standard/Proprietary
IEEE 802.3ad	Yes	Yes	LACP	Standard
EtherChannel	No	No	None	Proprietary

LACP compatibility requirements for link aggregates

Describes link aggregation using LACP, explaining how to enhance bandwidth and redundancy for Layer 2 VPNs through aggregation protocols and best practices.

Key compatibility requirements enforced by LACP:

- All aggregated links terminate on the same two systems.
- Both systems recognize the links as belonging to the same bundle.
- All links are configured with appropriate settings on both peers.

LACP operates by transmitting frames that contain information about the local port state and the local view of the partner system's state. These frames are analyzed to ensure both partners are in agreement regarding bundle membership and link status.

LACP ensures only compatible and operational links are included in a bundle, maintaining network reliability and preventing misconfigurations.

7 Layer 2 Bridging Services

Topics:

- Layer 2 bridge services
- Flooding disable
- Virtual private LAN bridging services
- Pseudowires in VPLS bridge domains
- VPLS discovery and signaling models
- CFM on VPLS
- Split-horizon groups
- Traffic storm control
- GTP load balancing

Outlines Layer 2 bridging solutions, including bridge domains, VLAN bridging, MAC address management, VPLS configuration, pseudowire operations, storm control, and GTP load balancing for robust Ethernet and VPN service delivery.

Layer 2 bridge services

Introduces Layer 2 bridging concepts, detailing bridge domains, ports, flushing operations, MAC address tables, replication lists, VLAN bridging workflows, and comprehensive configuration and management procedures for Ethernet bridging.

A Layer 2 bridging service is a logical bridge-based Layer 2 service that

- switches data frames within a Layer 2 broadcast domain by using destination MAC addresses
- floods multicast, broadcast, and unknown unicast frames within that domain, and
- learns source MAC addresses on incoming frames.

A logical bridge contains bridge domains, bridge ports, MAC address tables, and replication member lists.

Layer 2 bridge components

Lists the logical components that comprise Layer 2 bridge services.

Layer 2 bridge services use a small set of logical components that provide connectivity and segmentation in network environments such as data centers, campuses, and global networks. The main components of a logical bridge are:

- Bridge domain: The fundamental segment that defines the broadcast domain for Layer 2 traffic.
- Bridge port: The interface or logical port that connects devices or network segments to the bridge domain.
- MAC address table: Maintains a mapping of device MAC addresses to bridge ports, enabling correct traffic forwarding within the bridge.
- Replication member list: Identifies all the bridge ports or endpoints where Layer 2 frames should be replicated for broadcast or multicast traffic.

These components work together to deliver efficient Layer 2 bridging services in various network configurations.

Bridge domains

Provides information about bridge domains, including how they function as Layer 2 broadcast domains, their methods for switching and learning MAC addresses, and notable feature enhancements on Cisco platforms.

A bridge domain refers to a Layer 2 broadcast domain consisting of a set of physical or virtual ports. Data frames are switched within a bridge domain based on the destination MAC address. Multicast, broadcast, and unknown destination unicast frames are flooded within the bridge domain. In addition, the source MAC address learning is performed on all incoming frames on a bridge domain.

A learned MAC address has an age attribute. The MAC address is remembered for a specified aging time and is removed if it has not been seen in received traffic during the aging period.

Switches assign a locally significant ID to each bridge domain, called the bridge domain ID. Many legacy switches use VLAN as the bridge domain ID, known as the bridging VLAN.

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
Bridge Domain	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.

Feature Name	Release Information	Feature Description
Bridge Domain	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The Bridge Domain functionality is now extended to the Cisco 8712-MOD-M routers.
Bridge Domain	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The Bridge Domain functionality is now extended to these fixed systems and line cards: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Bridge Domain	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) A bridge domain provides a flexible and efficient Layer 2 broadcast domain by grouping physical or virtual ports to facilitate data frame switching based on MAC addresses. This setup enables effective handling of multicast, broadcast, and unknown unicast frames by flooding them within the bridge domain. *This functionality is now extended to routers with the 88-LC1-36EH line cards.

Bridge ports

Lists the key attributes and functions of logical bridge ports in a bridge domain.

A logical bridge port identifies a unique network segment in a bridge domain. Logical bridge ports enable L2 traffic to traverse a bridge domain and operate independently of traffic encapsulation type, such as VLAN or MPLS.

Key facts about bridge ports:

- Each logical bridge port is associated with a specific network segment within the bridge domain.
- Bridge ports facilitate L2 traffic transit in the bridge domain.
- The function of a bridge port is independent of L2 traffic encapsulation (for example, VLAN, MPLS).
- Logical bridge ports perform native bridging operations:
  - Forwarding of L2 frames.
  - Destination MAC address lookup.
  - Source MAC address learning.
  - MAC address aging.

Bridge port flush and bridge flush

Explains how the bridge port flush and bridge flush feature deletes learned MAC addresses after a bridge-port failure.

A bridge port flush and bridge flush feature is a MAC cleanup feature that

- deletes learned MAC addresses at the bridge port and bridge domain levels after a bridge port goes down
- prevents traffic from other ports from unicasting to the affected port, and
- expedites flooding-based MAC relearning.

Table 12: Feature history table

Feature Name	Release Information	Feature Description
Bridge Port Flush and Bridge Flush	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Bridge Port Flush and Bridge Flush	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The Bridge Port Flush functionality is now extended to the Cisco 8712-MOD-M routers.
Bridge Port Flush and Bridge Flush	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The Bridge Port Flush functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Bridge Port Flush and Bridge Flush	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) * The Bridge Port Flush functionality is now extended to routers with the 88-LC1-36EH line cards.
Bridge Port Flush and Bridge Flush	Release 7.3.2	During a port failure, this feature allows the router to delete the learned MAC addresses at the bridge port and bridge domain levels. The deletion of MAC addresses is important because it prevents traffic from other ports to unicast to the affected port, leading to traffic drop. Also, the clean-up ensures flooding of data packets to expedite the process of relearning MAC addresses. The Bridge Port Flush feature enables the router to delete the MAC addresses automatically, whereas, to delete the learned MAC addresses at the bridge domain level, use the **clear l2vpn bridge-domain mac-address-table** command.

Bridge port flush behavior

Bridge port flush and bridge flush remove learned MAC addresses after a bridge-port event and can also use unicast-disable during convergence.

A VPLS bridge sends out a MAC address withdrawal message on every PW when a bridge port (AC or PW) goes down. Upon receiving the MAC address withdrawal message, a VPLS bridge deletes all the MAC addresses learned on a PW. When MAC flush occurs, the MAC addresses are deleted one at a time. The time required to delete all the MAC addresses depends on the number of MAC addresses learned on that bridge port.

You can transition the bridge to a unicast-disable mode for a brief period during the MAC flush at the bridge-domain level.

You can use the following commands to remove MAC addresses from the hardware MAC table:

- **clear l2vpn bridge-domain mac-address-table**: Removes learned MAC addresses from the L2VPN MAC address tables at the bridge domain level.
- **clear l2vpn forwarding mac-address-table**: Removes entries from the L2VPN forwarding MAC address tables.

By removing MAC addresses from the hardware MAC table, you eliminate the need to wait for MAC addresses to age out naturally. This allows you to troubleshoot or recover quickly from MAC learning and forwarding issues. After you clear the MAC addresses, Cisco IOS XR software treats unicast traffic destined for those addresses as unknown unicast, which results in unicast flooding.

Always use the **clear l2vpn** commands with extreme caution to avoid unintended network issues.

Limitations of bridge port flush and bridge flush

Outlines the recommended limitations for using MAC-clear and bridge-flush commands in order to maintain network stability and MAC synchronization.

Follow these principles when clearing MAC addresses at the bridge-port or bridge-domain level:

- Use the **clear l2vpn bridge-domain mac-address-table** and **clear l2vpn forwarding mac-address-table** commands only for troubleshooting, as they can disrupt ARP and ND learning on BVI interfaces in both Fixed Systems (8200, 8700, 8010) and Modular Systems (8800).
- When you use the **clear l2vpn forwarding mac-address-table location x/y/z** command on Modular Systems (8800), Cisco IOS XR software removes the MAC table only on the specified line card. This can cause the MAC tables on different line cards to become out of sync, leading to inconsistent forwarding behavior across the modular system.
- Always use the **clear l2vpn** commands with extreme caution to avoid unintended network issues.

Disable unicast traffic during bridge flush

Configure unicast-disable during bridge flush and verify that the hardware profile shows the setting as applied.

Disable unicast forwarding during bridge flush to ensure all traffic floods to the bridge, which accelerates convergence as MAC entries are being deleted.

By default, unicast traffic remains enabled during a MAC flush event at the bridge-domain level. However, disabling unicast traffic during bridge flush using the **hw-module profile l2fib bridge-flush-convergence** command floods all traffic to the bridge, expediting convergence as table lookups are avoided. Unicast traffic is disabled for a duration from 1 to 30 seconds, depending on the number of MAC addresses learned on the bridge domain, and is not user configurable. Once the MAC flush completes, unicast forwarding is automatically reenabled.

Before you begin

- Ensure you have access to the device's global configuration mode.
- Confirm you have administrative privileges

- Review current hardware profile configuration.

Follow these steps to disable unicast traffic during bridge flush and verify the applied profile.

Procedure

1. Enable bridge-flush convergence in global configuration mode.

   Example

   ```
   Router# configure
   Router(config)# hw-module profile l2fib bridge-flush-convergence
   Router(config)# commit
   ```

2. Check the running configuration.

   Example

   ```
   configure
    hw-module profile l2fib bridge-flush-convergence
   !
   ```

3. Use the **show hw-module profile l2fib** command to verify that the hardware profile shows BD-Flush-Convergence as configured and applied.

   Example

   ```
   Router# show hw-module profile l2fib
   ------------------------------------------------------------
   Knob                          Status          Applied   Action
   ------------------------------------------------------------
   PW-Stats                      Unconfigured    N/A       None
   BD-Flush-Convergence          Configured      Yes       None
   ------------------------------------------------------------
   ```

Unicast traffic is disabled during bridge flush, and the hardware profile reflects the applied configuration.

MAC address tables in bridge domains

Each bridge domain has a unique MAC address table that records source MAC addresses and forwarding information.

A MAC address table records forwarding or filtering information for a bridge domain. Each bridge domain contains a unique MAC address table comprised of MAC address entries. When an Ethernet frame is received on a bridge port, the source MAC address and bridge port are recorded in the MAC address table. This information is then used for traffic forwarding in the reverse direction.

Table 13: MAC address table

MAC Address	Ports
1001.1001.2002	Port 2
1001.1001.2003	Port 5

MAC Address	Ports
1001.1001.2004	Drop

Replication member lists

Provides information about replication member lists and their role in traffic flooding.

A replication member list is a list of virtual bridge ports that allow traffic flooding within a bridge domain. Each bridge domain has one replication member list.

Key facts about replication member lists:

- Identify the virtual bridge ports that allow traffic flooding.
- Each bridge domain maintains its own replication member list.
- The lists enable efficient data distribution across network segments.

Configure a bridge domain

Configure a bridge domain, add member interfaces, set the flooding parameter, and optionally disable the bridge domain.

Configure a bridge domain and its operational settings in the network.

Use this task to create a bridge domain, associate interfaces, adjust parameters such as flooding, and disable the domain if needed.

Before you begin

Make sure you have access to the device CLI and the necessary privileges to configure bridge domains.

Follow these steps to configure the bridge domain and its associated settings:

Procedure

1. Create the bridge domain.

   Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# commit
```

2. Associate an interface with the bridge domain.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# commit
```

3. Configure the flooding parameter for the bridge domain.

Flooding is enabled by default.

Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# flooding disable
Router (config-l2vpn-bg-bd)# commit
```

4. Disable the bridge domain when you need to shut it down.

   When a bridge domain is disabled, all ACs that are associated with the bridge domain are disabled. You are still able to attach or detach members to the bridge domain.

   Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# shutdown
Router (config-l2vpn-bg-bd)# commit
```

5. Review the resulting running configurations.

   Example

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface HundredGigE0/0/0/0
    flooding disable
    shutdown
   !
  !
```

The bridge domain and its member interface are configured with the requested operational settings.

VLAN bridge modes

Explains the simplest Layer 2 bridging technique using VLAN-tagged attachment circuits to extend Layer 2 flood domains and classify Ethernet traffic.

A VLAN bridging mode is a Layer 2 bridging mode that

- receives Ethernet II and IEEE 802.3 traffic on VLAN-tagged attachment circuits
- classifies ingress traffic into Layer 2 bridge domains, and
- can apply VLAN tag rewrite on ingress and egress.

In modern networks, a majority of the Ethernet frames are in Ethernet II frame format. Legacy Layer 2 protocol traffic, such as spanning tree protocol and CDP are in IEEE 802.3 frame format.

How VLAN bridging works

VLAN bridging classifies tagged ingress traffic into bridge domains and forwards it to local or remote customer edge devices.

VLAN bridging preserves host mobility by extending the Layer 2 flood domain instead of relying on IP segmentation.

Summary

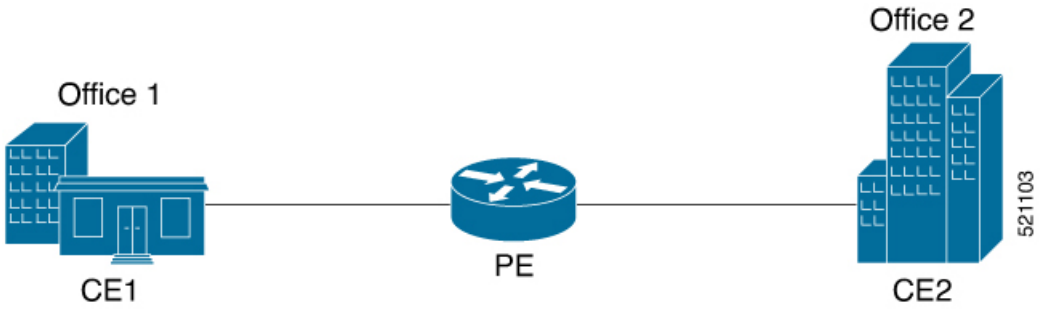
The key components involved in VLAN bridging are:

- Customer edge devices: Send and receive VLAN-tagged Layer 2 traffic.
- The edge router: Classifies ingress traffic into bridge domains and applies optional VLAN tag rewrite.
- The remote router: Bridges the traffic to local office buildings after optional VLAN tag rewrite.

VLAN bridging in a campus network extends the Layer 2 flood domain across floors and buildings, enabling MAC hosts to move without dropping TCP or IP sessions.

Workflow

Figure 1: VLAN bridging



VLAN bridging involves these stages:

1. The customer edge devices send Ethernet II or IEEE 802.3 traffic to the edge router on single-tagged or double-tagged VLAN attachment circuits.
2. The edge router classifies the ingress traffic into the correct Layer 2 bridge domains and performs optional VLAN tag rewrite.
3. The edge router forwards the traffic either to a different local customer edge device or to a remote router, and the remote router bridges the traffic to local office buildings after optional VLAN tag rewrite.

Result

Hosts can move across the bridged campus network while remaining in the same Layer 2 flood domain.

Configure VLAN bridging

Configure VLAN attachment circuits, map them to bridge domains, and verify the VLAN bridging state.

Set up VLAN bridging by creating attachment circuits, associating them to bridge domains, and confirming successful bridging.

This task explains how to configure VLAN bridging using two bridge domains and four VLAN attachment circuits, mapping each pair of circuits to a bridge domain.

Before you begin

- Prepare the required VLAN IDs and ensure you have access to the router CLI.

- The example configuration described here uses four attachment circuits and divides them into two groups, each mapped to a separate bridge domain.

Follow these steps to configure VLAN bridging:

Procedure

1. Configure the first pair of VLAN attachment circuits.

   Example

```
Router# configure
Router(config)# interface HundredGigE0/0/0/4.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# exit
Router(config)# interface HundredGigE0/0/0/4.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# exit
```

2. Configure the second pair of VLAN attachment circuits.

   Example

```
Router(config)# interface HundredGigE0/0/0/5.1 l2transport
Router(config-subif)# encapsulation dot1q 3
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# exit
Router(config)# interface HundredGigE0/0/0/5.2 l2transport
Router(config-subif)# encapsulation dot1q 4
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# exit
```

3. Create the first bridge domain and associate the first two attachment circuits.

   Example

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/4.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/5.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
```

4. Create the second bridge domain and associate the second two attachment circuits.

   Example

```
Router(config-l2vpn)# bridge group bg2
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/4.2
Router(config-l2vpn-bg-bd-ac)# exit
```

```
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/5.2
Router(config-l2vpn-bg-bd-ac)# commit
```

5. Review the running configuration.

Example

```
interface HundredGigE0/0/0/4.1 l2transport
encapsulation dot1q 1
rewrite ingress tag pop 1 symmetric
!
interface HundredGigE0/0/0/4.2 l2transport
encapsulation dot1q 2
rewrite ingress tag pop 1 symmetric
!
interface HundredGigE0/0/0/5.1 l2transport
encapsulation dot1q 3
rewrite ingress tag pop 1 symmetric
!
interface HundredGigE0/0/0/5.2 l2transport
encapsulation dot1q 4
rewrite ingress tag pop 1 symmetric
!
bridge group bg1
 bridge-domain bd1
  interface HundredGigE0/0/0/4.1
  !
  interface HundredGigE0/0/0/5.1
  !
 !
bridge group bg2
 bridge-domain bd2
  interface HundredGigE0/0/0/4.2
  !
  interface HundredGigE0/0/0/5.2
  !
 !
```

6. Use the **show interfaces hundredGigE 0/0/0/4.2**, **show l2vpn bridge-domain summary**, **show l2vpn forwarding bridge-domain location 0/RP0/CPU0**, and **show l2vpn forwarding bridge-domain bg1:bd1 location 0/RP0/CPU0** command to verify the VLAN bridging state.

Example

```
Router# show interfaces hundredGigE 0/0/0/4.2
Tue Sep 22 11:32:06.993 PDT
HundredGigE0/0/0/4.2 is up, line protocol is up
Interface state transitions: 101
Hardware is VLAN sub-interface(s), address is c4b2.39da.1620
Layer 2 Transport Mode
MTU 1518 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability Unknown, txload Unknown, rxload Unknown
Encapsulation 802.1Q Virtual LAN, Outer Match: Dot1Q VLAN 2
Ethertype Any, MAC Match src any, dest any
loopback not set,
Last link flapped 2d10h
Last input 00:00:00, output 00:00:00
Last clearing of "show interface" counters 3d18h
21364536641 packets input, 2734660346522 bytes
```

```
0 input drops, 0 queue drops, 0 input errors
8420820982 packets output, 1077864630044 bytes
0 output drops, 0 queue drops, 0 output errors
Router# show l2vpn bridge-domain summary
Tue Sep 22 11:31:29.819 PDT
Number of groups: 2, VLAN switches: 0
Number of bridge-domains: 510, Up: 510, Shutdown: 0, Partially- programmed:
0
Default: 510, pbb-edge: 0, pbb-core: 0
Number of ACs: 1530 Up: 1275, Down: 255, Partially-programmed: 0
Number of PWs: 0 Up: 0, Down: 0, Standby: 0, Partially-programmed: 0
Number of P2MP PWs: 0, Up: 0, Down: 0, other-state: 0
Number of VNIs: 0, Up: 0, Down: 0, Unresolved: 0
Router# show l2vpn forwarding bridge-domain location 0/RP0/CPU0
Tue Sep 22 11:36:01.888 PDT
Bridge MAC Bridge-Domain Name ID Ports HW addr SW addr Flooding Learning State
------------------------------- ------ ----- ------- ------- -------- --------
 ---------
bg1:bd1 511 2 405 405 Enabled Enabled UP
bg1:bd2 510 2 405 405 Enabled Enabled UP
------------------------------------------------------------------------------
Router# show l2vpn forwarding bridge-domain bg1:bd1 location 0/RP0/CPU0
Tue Sep 22 11:36:37.141 PDT
Bridge MAC Bridge-Domain Name ID Ports HW addr SW addr Flooding Learning State
------------------------------- ------ ----- ------- ------- -------- --------
 ---------
bg1:bd1 511 2 405 405 Enabled Enabled UP
------------------------------------------------------------------------------
```

The VLAN attachment circuits and bridge domains are configured, and the verification output shows the bridge domains in the UP state.

MAC address-related parameters in bridge domains

Provides a list of MAC address parameters used in bridge domains to control Layer 2 forwarding behavior.

The MAC address table contains a list of known MAC addresses and their forwarding information. The MAC address table is managed and stored on the route processor card.

The following MAC address parameters govern bridge-domain forwarding behavior:

MAC address flooding

- Frames sent to broadcast, unknown unicast, and multicast addresses are flooded to all attachment circuits within the bridge domain.
- The provider edge (PE) router replicates packets across all circuits for such addresses.

MAC address-based forwarding

- A PE associates each destination MAC address with an attachment circuit, either statically or through dynamic learning.
- Static configuration or dynamic learning enables forwarding decisions.

MAC address source-based learning

- When a frame arrives on a bridge port and the source MAC address is unknown, the PE associates the address with the attachment circuit.

- The hardware notifies the control plane about the new MAC address and its port. The control plane stores this information and programs it into MAC tables across all forwarding ASICs.
- If a MAC address is statically configured on an attachment circuit, it does not age or move through dynamic learning.
- Do not send packets with a source MAC address matching another statically configured MAC address on a different attachment circuit.

MAC address aging

- MAC addresses are valid for the duration of the configured aging time in the MAC table.
- Static entries are added by the network manager or bridge; dynamic entries by the bridge learning process. Dynamic entries are removed after the aging time.
- Adjust the aging time based on host behavior: decrease to quickly adapt to moving hosts, increase to reduce flooding for intermittently transmitting hosts.
- Aging time ranges from 300 to 30,000 seconds. The highest value among configured bridge domains is used.
- Set aging time at the bridge domain level, not per interface.
- There is no command to display the highest configured aging time.

The following table illustrates a sample aging scenario.

Bridge Domain	Aging Time
bd1	300s
bd2	600s
bd3	800s

MAC address aging time for all three bridge domains is 800 seconds.

MAC address limit

- Bridge domains support a MAC address limit to alert when the number of addresses exceeds the configured threshold.
- When the limit is exceeded, notifications can include:
  - Syslog (default)
  - Simple Network Management Protocol trap
  - Syslog and SNMP trap
  - None
- To enable both syslog and SNMP notifications, use the **mac limit notification** command both in bridge-domain configuration mode.
- The limit applies only to local MAC addresses.
- If no threshold is configured, the default is 131072.

Dynamic MAC withdrawals between peer PE routers

Explains how dynamic MAC withdrawals prevent packet drops between peer PE routers when an attachment circuit goes down.

A dynamic MAC withdrawal feature is a MAC address withdrawal feature that

- uses an LDP-based MAC address withdrawal message
- allows peer PE routers to remove learned dynamic MAC addresses after an attachment-circuit state change, and
- prevents packet drops and speeds convergence when the attachment circuit comes back up.

Table 14: Feature history table

Feature Name	Release Information	Feature Description
Withdraw Dynamic MAC Addresses Between Peer PE Routers	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-32Y8L2H2FH routers.
Withdraw Dynamic MAC Addresses Between Peer PE Routers	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Withdraw Dynamic MAC Addresses Between Peer PE Routers	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The MAC address withdrawal functionality is now extended to the Cisco 8712-MOD-M routers.
Withdraw Dynamic MAC Addresses Between Peer PE Routers	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only* *The MAC address withdrawal functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Withdraw Dynamic MAC Addresses Between Peer PE Routers	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) We now prevent packet drops between peer routers when the attachment circuit (AC) of a PE router goes down, by withdrawing all dynamic MAC addresses from that PE router. When the AC goes down, the PE routers remove or unlearn the MAC addresses learned from the peer routers, that do not need to be relearned. This enables faster convergence when the AC comes up. *This feature is supported on routers with the 88-LC1-36EH line cards.

Optimized MAC address withdrawal behavior on PE routers

Provides details on how optimized MAC address withdrawal on provider edge (PE) routers prevents packet drops, improves convergence, and streamlines MAC address management.

Key behavior

Benefits

Default behavior and configuration

Optimized MAC address withdrawal prevents packet drops between peer routers when the attachment circuit (AC) on a provider edge (PE) router goes down.

This feature uses a Label Distribution Protocol (LDP)-based MAC address withdrawal message. The message includes a MAC address list type-length-value (TLV).

- The feature optimizes MAC address withdrawal during an AC failure.
- The PE retains MAC addresses that it learns from customer edge (CE) devices on the access side.
- The PE clears only the MAC addresses that it learns from peer PEs.
- Because the PE does not need to relearn the cleared MAC addresses from the access side, the network achieves faster convergence when the AC comes back up.
- Prevents packet drops between peer routers.
- Reduces unnecessary MAC relearning.
- Improves convergence time after AC recovery.
- MAC address withdrawal is enabled by defaut.
- To disable MAC address withdrawal, use the **mac withdraw disable** command.

How MAC address withdrawal works

Dynamic MAC withdrawal uses LDP MAC-withdraw messages to clear peer-learned MAC addresses and speed convergence after attachment-circuit state changes.

MAC address withdrawal clears only the peer-learned MAC addresses that must be relearned, improving network convergence and minimizing disruption during attachment circuit state changes.

Summary

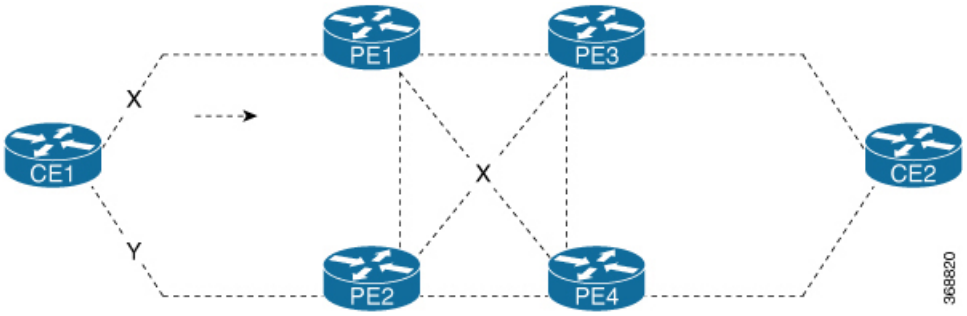
The key components involved in MAC address withdrawal are:

- Dual-homed customer edge device: Connects to PE1 and PE2 over active and redundant attachment circuits.
- Peer PE routers: Learn forwarding entries based on the traffic profile.
- LDP MAC-withdraw messages: Tell the peer PE routers which MAC entries to clear.

MAC address withdrawal prevents packet drops between peer routers when an attachment circuit goes down by withdrawing dynamic MAC addresses. The feature is enabled by default, retaining MAC addresses learned from customer edge devices on the access side.

Workflow

Figure 2: MAC address withdrawal



MAC address withdrawal involves these stages:

1. The PE routers learn MAC address forwarding entries from the traffic flowing across the active topology, and traffic becomes known unicast.
2. When link X, which is the attachment circuit of PE1, goes down, PE1 sends an LDP MAC-withdraw TLV message of "FLUSH ALL MAC FROM ME" to the neighbor PEs, and PE2, PE3, and PE4 clear the MAC addresses learned only from PE1.
3. When link Y, which is the attachment circuit of PE2, comes up, PE2 sends an LDP MAC-withdraw TLV message of "FLUSH ALL MAC BUT ME" to the neighbor PEs, and the peers clear the MAC addresses learned from the other PEs while retaining the entries learned from PE2.

Result

The peer PE routers preserve access-side learning where possible and converge faster when the attachment circuit state changes.

Restrictions on withdrawing dynamic MAC addresses between peer PE routers

Outlines the restrictions and unsupported scenarios for MAC address withdrawal between peer PE routers.

You must not use MAC address withdrawal in the following topologies or signaling combinations:

- Access pseudowires
- Hierarchical Virtual Private LAN Service (VPLS) networks
- Networks configured with BGP signaling and discovery

Additionally, do not expect MAC withdraw relaying (the option to forward received MAC withdraw messages), because it is not supported.

Configure MAC address withdrawal for a bridge domain

Configure MAC withdrawal for a bridge domain, optionally disable this feature, and verify the bridge-domain detail output.

Enable MAC address withdrawal so that dynamically learned MAC addresses are withdrawn when the attachment circuit is down.

By default, MAC address withdrawal is enabled for each bridge domain. You may disable it with the **mac withdraw disable** command, if required.

Before you begin

- Confirm you have access to the router CLI and bridge domain configuration.
- Ensure the relevant bridge group and bridge-domain already exist.

Follow these steps to configure MAC address withdrawal and verify the bridge-domain status.

Procedure

1. Configure MAC withdrawal on PE1 for the bridge domain and attachment circuit.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# withdraw state-down
Router(config-l2vpn-bg-bd-mac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# commit
```

2. Review the running configuration for the enabled state-down behavior.

   Example

```
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw state-down
   !
   interface HundredGigE0/0/0/0
   !
```

3. Disable MAC address withdrawal if you do not want the default behavior when the attachment circuit comes up.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# withdraw disable
Router(config-l2vpn-bg-bd-mac)# commit
```

4. Use the **show l2vpn bridge-domain detail** command to verify whether MAC withdrawal is enabled or disabled.

   Example

```
Router# show l2vpn bridge-domain detail
MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port down

Router# show l2vpn bridge-domain detail
MAC learning: enabled
  MAC withdraw: disabled
    MAC withdraw sent on: bridge port up
```

MAC address withdrawal is configured or disabled according to the bridge-domain requirements, and the bridge-domain detail output shows the current state.

Disable MAC address source-based learning for a bridge domain

Configure bridge domains to disable MAC address source-based learning. This prevents dynamically learned MAC addresses from aging or moving across attachment circuits.

Prevent dynamic MAC address learning in a bridge domain, allowing only static learning behavior.

By default, MAC address source-based learning is enabled. Disabling it ensures statically configured MAC addresses do not age or migrate between attachment circuits, which may be required in certain bridge-domain designs.

Before you begin

- Ensure you have access to the router CLI.
- Identify the bridge group and bridge domain you wish to modify.

Follow these steps to disable MAC address source-based learning for a bridge domain:

Procedure

1. Disable source-based learning under the bridge-domain MAC configuration.

   Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# mac
Router (config-l2vpn-bg-bd-mac)# learning disable
Router (config-l2vpn-bg-bd-mac)# commit
```

2. Review the running configuration.

   A statically configured MAC address cannot age or move to another attachment circuit through dynamic learning.

   Example

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    mac
    learning disable
    !
  !
```

MAC address source-based learning is disabled for the bridge domain.

Configure the MAC address limit

Configure the MAC address limit, notification mode, and threshold for a bridge domain.

Configure the MAC address limit parameters for a bridge domain to control address usage and receive notifications.

You can set the MAC address limit for the bridge domain as needed.

Before you begin

Follow these steps to configure the MAC address limit and notification threshold.

Procedure

1. Configure the MAC address maximum and notification mode.

   Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# mac
Router (config-l2vpn-bg-bd-mac)# limit
Router (config-l2vpn-bg-bd-mac-limit)# maximum 131072
Router (config-l2vpn-bg-bd-mac-limit)# notification both
Router (config-l2vpn-bg-bd-mac-limit)# exit
```

2. Configure the limit threshold and commit the change.

   Example

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

3. Review the running configuration.

   Example

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    mac
     limit
     maximum 131072
     notification both
    !
    mac limit threshold 80
   !
  !
```

The bridge domain uses the configured MAC address limit, notification mode, and threshold.

Set the MAC address aging timer

Set the MAC aging timer for the bridge domain.

Configure the MAC address aging timer to determine how long MAC addresses remain in the bridge domain's table before expiring.

MAC address aging time can be configured from 300 seconds to 30,000 seconds. If multiple bridge domains are configured with different aging times, the system uses the highest value across all bridge domains.

Before you begin

Follow these steps to configure MAC address aging in the bridge-domain MAC configuration mode.

Procedure

1. Configure the MAC address aging timer.

   Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# mac
Router (config-l2vpn-bg-bd-mac)# aging
Router (config-l2vpn-bg-bd-mac-aging)# time 300
Router (config-l2vpn-bg-bd-mac-aging)# commit
```

2. Review the running configuration.

   Example

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    mac
     aging
      time 300
     !
    !
```

The bridge domain uses the configured MAC address aging timer, ensuring MAC addresses are removed after the specified period.

Flooding disable

Explains the flooding disable feature, providing guidance on configuration steps to control unwanted broadcast and unknown traffic within bridge domains.

A flooding disable feature is a bridge-level forwarding control that

- prevents forwarding of broadcast, unknown-unicast, and multicast traffic on a bridge domain
- can disable flooding of BUM traffic at the bridge level, and
- can also disable only unknown-unicast traffic at the bridge level.

Configure flooding at the bridge level

Configure flooding disable for BUM traffic or unknown-unicast traffic to improve network efficiency at the bridge domain level.

Prevent unwanted traffic flooding within a bridge domain by selectively disabling flooding for BUM (broadcast, unknown-unicast, and multicast) traffic or unknown-unicast traffic.

You can improve network performance and control how traffic is handled by disabling flooding of BUM traffic or unknown-unicast traffic at the bridge domain level. Disabling unknown-unicast flooding only applies when BUM flooding disable is not configured.

Before you begin

Follow these steps to configure either flooding disable behavior.

Procedure

1. Disable flooding of BUM traffic at the bridge level.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# flooding disable
Router(config-l2vpn-bg-bd)# commit
```

2. Disable flooding of unknown-unicast traffic at the bridge level when BUM flooding disable is not configured.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# flooding unknown-unicast disable
Router(config-l2vpn-bg-bd)# commit
```

3. Review the running configuration.

   Example

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding disable
   !

configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding unknown-unicast disable
```

```
    !
  !
```

The bridge domain uses the specified flooding-disable setting, effectively controlling network traffic as desired.

Virtual private LAN bridging services

Details VPLS architecture, outlining service characteristics, operational principles, and process flows for implementing multipoint Layer 2 VPN services over packet-switched networks.

A virtual private LAN bridging service is a multipoint Layer 2 VPN service that

- connects two or more provider edge devices by using bridge domains
- floods unknown, broadcast, and multicast traffic across the bridged domain, and
- learns source MAC addresses to forward known unicast traffic to the correct customer edge device.

Table 15: Feature history table

Feature Name	Release Information	Feature Description
Virtual Private LAN Bridging Services	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Virtual Private LAN Bridging Services	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Virtual Private LAN Bridging Services	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The Virtual Private LAN Bridging Services functionality is now extended to the Cisco 8712-MOD-M routers.
Virtual Private LAN Bridging Services	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only* *The Virtual Private LAN Bridging Services functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
Virtual Private LAN Bridging Services	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The Virtual Private LAN Bridging Services functionality is now extended to routers with the 88-LC1-36EH line cards.
Virtual Private LAN Bridging Services	Release 7.3.2	This feature employs PE routers connected by a mesh of tunnels, enabling you to connect multiple customer devices in a single bridged domain. Such a setup allows service providers to seamlessly offer a variety of services that they can provision rapidly.

VPLS service attributes

Provides the technical attributes and functional features of a VPLS service.

Key benefits

A Virtual Private LAN Service (VPLS) enables a service provider to deliver private multipoint Layer 2 VPN services to multiple customers over an MPLS network. These attributes characterize a VPLS service:

- Customer-specific bridge domains: Each customer receives a dedicated bridge domain, ensuring complete separation and privacy. Packets from one bridge domain are never delivered to another.
- Attachment circuits: Access to each bridge domain on provider edge (PE) routers is managed through attachment circuits, which can include physical or virtual ports.
- MAC-based forwarding: The service provider network switches packets within the customer's bridged domain by using destination MAC addresses. Unknown, broadcast, and multicast MAC addresses are flooded to all customer edge devices, while known MAC addresses are forwarded directly.
- Multipoint connectivity: VPLS connects multiple PE devices through bridging, enabling private multipoint LAN-like connectivity for customers.
- Dynamic learning: Provider edge devices dynamically learn source MAC addresses as packets are flooded within the customer-specific bridge domain.
- Traffic privacy: All traffic is isolated per customer through dedicated bridge domains and MAC-based forwarding, ensuring LAN privacy.
- Enables private, scalable Layer 2 multipoint services.
- Provides traffic isolation and privacy for each customer.
- Supports flexible attachment circuits (physical and virtual ports).

How VPLS works

Describes how VPLS connects PE routers to provide a multipoint LAN service using bridge domains, attachment circuits, and pseudowires.

VPLS enables customers to extend their LAN across geographically dispersed locations by emulating an Ethernet switch across the MPLS network, eliminating the need for multiple point-to-point circuits.

Summary

The key components involved in VPLS are:

- The PE routers: Host the bridge domains and VPLS forwarding MAC tables.
- The bridge domains: Provide the building blocks for multipoint bridging on each PE router.
- The attachment circuits: Connect the customer edge devices to the bridge domains.
- The pseudowires and VFI: Interconnect the PE routers into a single multipoint service.

VPLS is a technology that enables a service provider to deliver multipoint Ethernet service over an MPLS network. By interconnecting provider edge (PE) routers through bridge domains, attachment circuits, and a mesh of pseudowires, customer devices communicate as if they are on the same LAN.

Workflow

Figure 3: VPLS architecture



VPLS involves these stages:

1. The service provider creates a bridge domain on each PE router and connects the customer edge devices to that bridge domain through attachment circuits.
2. The PE routers interconnect through a full mesh of pseudowires, and the VFI links that pseudowire mesh to the bridge domain to form a virtual switching instance.
3. The PE routers switch traffic across the service by using destination MAC addresses, flood unknown, broadcast, and multicast traffic where required, and forward learned unicast traffic to the correct customer edge device.

Result

All customer edge devices in the VPLS instance appear to be on the same LAN without requiring a full mesh of point-to-point circuits at the customer edge.

Pseudowires in VPLS bridge domains

Describes pseudowire functionality within VPLS bridge domains, covering configuration, statistics monitoring, forwarding instances, transport types, control-word operations, and flow-aware transport for optimized load balancing.

A pseudowire in a VPLS bridge domain is a point-to-point connection that

- connects pairs of PE routers
- emulates Ethernet service over an MPLS core, and
- encapsulates the service in a common MPLS format.

Multiple pseudowires within a bridge domain are supported. However, multiple pseudowires to the same destination within the same bridge domain are not supported.

Configure a pseudowire under a bridge domain

Configure a bridge-domain pseudowire and review the resulting running configuration. This task ensures connectivity over Layer 2 VPN using a pseudowire within a bridge domain.

Configure a pseudowire under a bridge domain to establish Layer 2 connectivity between routers.

This task is commonly used in MPLS or carrier Ethernet scenarios where a pseudowire needs to be associated with a specific bridge domain.

Before you begin

Follow these steps to configure the pseudowire exactly as shown in the source example.

Procedure

1. Configure the pseudowire under the bridge domain.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# neighbor 10.0.0.2 pw-id 1
Router(config-l2vpn-bg-bd-pw)# commit
```

2. Review the running configuration.

   Example

```
l2vpn
 bridge group BG1
  bridge-domain BD1
   neighbor 10.0.0.2 pw-id 1
  !
!
```

The bridge domain includes the configured pseudowire neighbor and pw-id.

Check ingress pseudowire statistics

Enable pseudowire statistics, reload the slots, and inspect the aggregate and unicast counters.

Configure pseudowire statistics for ingress pseudowires, reload the slots, and verify aggregate and unicast counters.

Use the **hw-module profile l2fib pw-stats** command to enable pseudowire statistics. After you enable pseudowire statistics, you must reload all slots for the configuration to take effect.

Pseudowire statistics are not supported on line cards based on Q100 Silicon.

Before you begin

Follow these steps to enable and inspect pseudowire statistics.

Procedure

1. Enable pseudowire statistics and reload the slots so the configuration takes effect.

   Only aggregate and unicast statistics are supported for ingress PW disposition. Statistics resources are limited, and enabling statistics on pseudowires impacts forwarding performance.

   Example

```
Router# configure
Router(config)# hw-module profile l2fib pw-stats
```

```
Router(config)# commit
Router# reload location all
```

2. Use the **show hw-module profile l2fib** command to check the hardware profile before and after the reload.

   Example

```
Before the reload:
Router# show hw-module profile l2fib
-----------------------------------------------------------------
Knob                          Status          Applied   Action
-----------------------------------------------------------------
PW-Stats                      Configured      No        Reload
BD-Flush-Convergence          Unconfigured    N/A       None

After the reload:
Router# show hw-module profile l2fib
-----------------------------------------------------------------
Knob                          Status          Applied   Action
-----------------------------------------------------------------
PW-Stats                      Configured      Yes       None
BD-Flush-Convergence          Unconfigured    N/A       None
```

3. Inspect the aggregate and unicast statistics for ingress pseudowires.

   Example

```
Router# show l2vpn forwarding bridge-domain detail location 0/RP0/CPU0 | inc
 "state:Nbor|XC ID|received"
    XC ID: 0x1
      packets: received 2081 (multicast 0, broadcast 2081, unknown unicast
0, unicast 0), sent 998
      bytes: received 266368 (multicast 0, broadcast 266368, unknown unicast
 0, unicast 0), sent 127744
    XC ID: 0x2
     packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast
 0), sent 3079
      bytes: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast
 0), sent 394112
    XC ID: 0xa0000001
      packets: received 998 (unicast 0), sent 1996
      bytes: received 145708 (unicast 0), sent 307384
```

Pseudowire statistics are enabled and the aggregate and unicast counters are visible in the forwarding output.

Virtual forwarding instances in VPLS

Provides an overview of virtual forwarding instances (VFIs) in VPLS, including their purpose, features, and sample configurations for implementation.

Key features of VFIs in VPLS

Virtual forwarding instances (VFIs) are virtual bridge ports used in VPLS networks to interconnect the pseudowire mesh for each VPLS instance. VFIs perform native bridging functions required for packet forwarding, including learning and aging of MAC addresses, and directing data based on destination MAC addresses.

- Provide virtual bridge ports within the VPLS instance.

- Perform native bridging functions such as forwarding based on destination MAC addresses, source MAC address learning, and aging.
- Enable PE routers to make packet-forwarding decisions by referencing the VFI associated with a particular VPLS instance.
- Support connection of multiple attachment circuits belonging to a given VPLS.

Here is the sample configuration for a VFI under the bridge domain.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# vfi V1
Router(config-l2vpn-bg-bd-vfi)# neighbor 10.0.0.2 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)# commit
```

The corresponding running configuration is:

```
l2vpn
 bridge group BG1
  bridge-domain BD1
   vfi V1
    neighbor 10.0.0.2 pw-id 1
  !
!
```

VPLS pseudowire transport types

Lists the transport types available for VPLS pseudowires, providing a quick reference for supported options in VPLS deployments.

VPLS pseudowires support several transport types, enabling flexible and scalable network designs for multipoint Layer 2 VPN services. Each transport type provides different underlying mechanisms for establishing connectivity across provider networks.

The supported transport types for VPLS pseudowires are:

- LDP: Utilizes LDP for signaling and establishing pseudowire tunnels.
- Segment routing: Leverages segment routing for simplified path management and traffic engineering.
- LDP over TE: Combines LDP signaling with TE tunnels for enhanced path control.
- BGP-LU for inter-AS C topology: Uses BGP with Label Unicast to support inter-AS connections in C topology scenarios.

Supported pseudowire types and VLAN tag transport behavior

Provides details on supported pseudowire types for MPLS and describes how VLAN tags are transported in each mode.

Ethernet port mode (Type 5)

VLAN mode (Type 4)

VLAN tag transport table

These pseudowire types are supported for MPLS:

- Both ends of the pseudowire connect to Ethernet ports and transport a complete ethernet trunk.

- Frames received on a main interface or subinterface are transported by the ingress PE.
- This mode eliminates the need for dummy VLAN tags and reduces overhead.
- Frame tagging is no longer necessary.
- The ingress PE does not remove incoming VLAN tags; they are transported over the pseudowire.
- Type 4 inserts an extra dummy tag with VLAN 0 onto the frame, which is removed at the remote end.
- Allows service providers to segregate traffic per customer using VLANs.
- Each VLAN on a customer-to-provider link can be configured as a separate Layer 2 VPN connection.

Pseudowire Mode	VC Type	VLAN Tag Handling	Dummy Tag Required	Use Cases
Ethernet Port Mode	5	Original tags are removed; the entire trunk is transported as-is.	No	Full trunk transport between Ethernet ports (Port-to-Port).
VLAN Mode	4	VLAN tags are preserved; a dummy tag (VLAN 0) may be added for frame prioritization.	Yes (if only 1 tag); not needed with >1 tag	Segregation of traffic per VLAN; Layer 2 VPN per specific VLAN.

Key considerations:

- VLAN-based VC type 4 pseudowires transport VLAN tags by pushing a dummy tag at the ingress when only a single tag is present.
- If multiple VLAN tags are pushed, a dummy tag is unnecessary—they are transported directly.
- The dummy tag (if added) is removed on the remote router before egress.
- Mixing Type 4 and Type 5 pseudowires in the same Virtual Forwarding Instance (VFI) is not supported: all pseudowires in a VFI must be of the same type.

Pseudowire control-word

Explains the role and structure of pseudowire control-words within MPLS networks.

A pseudowire control-word is an optional 4-byte protocol field that

- sits between the MPLS label stack and the Layer 2 payload in a pseudowire packet
- carries generic and Layer 2 payload-specific information, and
- enables sequencing, assists identification for load balancing, and supports Ethernet pseudowire packet handling.

The control-word can pad small packets, carry Layer 2 control bits, preserve sequence, support load balancing, and indicate payload fragmentation.

Functions of the pseudowire control-word

Provides details on the key functions of the pseudowire control-word, including padding, control bits, sequencing, load balancing, and fragmentation signaling.

The pseudowire control-word serves several key functions in AToM packet transport:

- Padding small packets: If an AToM packet is below the minimum length, the frame is padded to meet Ethernet requirements. The control-word includes a length field that helps the egress PE router determine whether padding was added and remove it before forwarding.
- Carrying control bits: The control-word transports control bits from the Layer 2 header of the protocol being carried.
- Sequencing transported frames: The control-word preserves the sequence of frames. Each packet sent over the pseudowire receives an incrementing sequence number, starting with 1 and continuing to 65535. Out-of-sequence packets are dropped; packets are never re-ordered.
- Load balancing: The control-word enables correct identification of Ethernet PW packets and prevents misordering caused by equal-cost multipath (ECMP) path selection. The control-word, inserted after the MPLS label, separates payload from MPLS and carries Layer 2 control bits and supports sequencing.
- Fragmentation signaling:

  The control-word indicates payload fragmentation status using two bits:

  - 00: Unfragmented
  - 01: First fragment
  - 10: Last fragment
  - 11: Intermediate fragment

Configure a pseudowire control-word

Configure a pseudowire class that enables the control word and apply the class to the pseudowire.

Enable the control-word for a bridge-domain pseudowire by using a pseudowire class.

The control-word keyword is inserted immediately after the MPLS label to separate the payload from the MPLS label over a pseudowire.

Before you begin

Follow these steps to configure the pseudowire control-word.

Procedure

1. Create a pseudowire class that enables the control-word.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class path1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
```

2. Apply the pseudowire class to the bridge-domain pseudowire and commit.

Example

```
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# neighbor 10.0.0.2 pw-id 1
Router(config-l2vpn-bg-bd-pw)# pw-class path1
Router(config-l2vpn-bg-bd-pw)# commit
```

3. Review the running configuration.

Example

```
l2vpn
 pw-class path1
  encapsulation mpls
   control-word
 bridge group BG1
  bridge-domain BD1
   neighbor 10.0.0.2 pw-id 1
    pw-class path1
  !
!
```

The pseudowire uses a pseudowire class with the control word enabled, ensuring correct payload separation after the MPLS label.

Flow-aware transport pseudowires

Explains how flow-aware transport pseudowires enable effective load balancing by identifying individual flows, inserting flow labels, and allowing routers to distribute traffic across equal-cost paths.

A flow-aware transport pseudowire is a pseudowire transport behavior that

- identifies individual flows within a pseudowire
- inserts a flow label as the lowermost label in the packet, and
- allows routers to use that flow label to load balance traffic across equal-cost or bundled core paths.

Table 16: Feature history table

Feature Name	Release Information	Feature Description
Load Balancing using Flow Aware Transport Pseudowire	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Load Balancing using Flow Aware Transport Pseudowire	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The load balancing with FAT PW functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
Load Balancing using Flow Aware Transport Pseudowire	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The load balancing with FAT PW functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Load Balancing using Flow Aware Transport Pseudowire	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) This feature enhances network performance by distributing traffic flows evenly across multiple pseudowires, preventing congestion and optimizing bandwidth usage. This method identifies and labels individual traffic flows, allowing for precise load distribution in MPLS networks. *This functionality is now extended to routers with the 88-LC1-36EH line cards.

Flow-aware transport pseudowire attributes and ECMP load balancing

Provides the attributes and benefits of flow-aware transport pseudowires (FAT PW) that improve load balancing for traffic carried over pseudowires in ECMP networks.

Flow-aware transport pseudowires (FAT PW) enhance load balancing for traffic carried over pseudowires. Routers typically load balance traffic using the lowermost label in the label stack. In standard pseudowires, this label remains the same for all flows, which can cause uneven traffic distribution and asymmetric load balancing.

Key characteristics of flow-aware transport pseudowires:

- FAT PW identifies individual packet flows within a pseudowire.
- FAT PW creates a flow label based on each packet flow entering the pseudowire.
- FAT PW inserts the flow label as the lowermost label in the packet.
- Routers use the flow label to make load-balancing decisions.

FAT PW enables routers to identify individual flows within a pseudowire and leverage that flow information for more effective load balancing. This capability is especially valuable in core networks using equal-cost multipaths (ECMP).

How flow-aware transport pseudowires work

Flow-aware transport pseudowires add a flow label so routers can identify flows and distribute them across ECMP or bundle paths.

Routers typically load balance pseudowire traffic using the lowermost label in the label stack. Since this label is uniform across all flows on a pseudowire, it can lead to uneven and asymmetric load balancing. Flow-aware transport pseudowires mitigate these issues by adding flow labels that differentiate packets, improving traffic distribution.

Summary

The key components involved in the process are:

- Ingress PE: Identifies individual packet flows and generates a unique flow label for each flow entering the pseudowire.
- Flow label: Provides entropy based on source and destination MAC and IP addresses, enabling differentiation among flows.
- Core routers: Use the flow label to load balance traffic across ECMP or bundled paths.
- Egress PE: Discards the flow label before forwarding the traffic.

Flow-aware transport pseudowires enable more efficient traffic distribution by adding entropy to packet flows. This ensures that routers can better identify individual flows and distribute them evenly across ECMP and bundled links.

Workflow

Figure 4: FAT PW with two flows distributing over ECMPs and bundle links



Flow-aware transport pseudowires involve these stages:

1. Flow identification and label generation: The ingress PE examines packets entering the pseudowire, identifies individual flows, and derives a flow label based on their source and destination MAC and IP addresses.
2. Label insertion: The ingress PE inserts the flow label after the VC label and before the control word (when present), setting the end-of-stack bit on the flow label.
3. Traffic load balancing and label removal: Core routers use the flow label and other packet information to distribute traffic across ECMP paths and link bundles. The egress PE discards the flow label before forwarding the traffic onward.

Result

Pseudowire traffic gains additional entropy, allowing for more even and efficient distribution across the core network. Note that MPLS OAM ping traffic cannot use flow aware transport pseudowires since MPLS OAM does not support the flow label.

Configure a flow-aware transport pseudowire

Configure a pseudowire class with a flow label and apply it to a bridge-domain pseudowire to enable improved ECMP load balancing.

Enable load balancing across ECMP and bundle links by applying a flow label to a transport pseudowire.

Routers typically perform load balancing based on the lowermost label in the label stack, which remains the same for all flows over a given pseudowire. Adding a flow label enables distribution of traffic across multiple ECMP or bundled paths in the core.

You cannot send MPLS OAM ping traffic over a FAT pseudowire, because there is no flow-label support for MPLS OAM.

Before you begin

Follow these steps to enable load balancing with a flow label under the pseudowire class.

Procedure

1. Create a pseudowire class that enables the flow label.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class FLOW
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# flow-label both
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
```

2. Apply the pseudowire class to the bridge-domain pseudowire and commit.

   Example

```
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# neighbor 10.0.0.2 pw-id 1
Router(config-l2vpn-bg-bd-pw)# pw-class FLOW
Router(config-l2vpn-bg-bd-pw)# commit
```

3. Review the running configuration.

   Example

```
l2vpn
 pw-class FLOW
  encapsulation mpls
   flow-label both
 bridge group BG1
  bridge-domain BD1
   neighbor 10.0.0.2 pw-id 1
    pw-class FLOW
  !
!
```

The pseudowire uses a flow label for improved load balancing across ECMP or bundle links.

VPLS discovery and signaling models

Explains mechanisms for VPLS discovery and signaling, including BGP and LDP autodiscovery, NLRI formats, and signaling processes for dynamic service setup across network domains.

A VPLS discovery and signaling model is a control-plane model that

- automatically discovers the PE routers that belong to a VPLS domain
- signals pseudowires between each pair of PE routers, and
- builds the full mesh of pseudowires used by the VPLS instance.

How VPLS discovery and signaling work

Describes how VPLS autodiscovery identifies PE routers within a VPLS domain and signaling establishes the full mesh of pseudowires between them.

VPLS is a Layer 2 multipoint service that emulates a LAN across a wide area network (WAN), allowing service providers to offer native Ethernet access to customers without manual neighbor provisioning.

Summary

The key components involved in VPLS discovery and signaling are:

- VPLS PE routers: Participate in the VPLS domain and emulate LAN connectivity for clients across a WAN.
- Autodiscovery: Identifies which PE routers belong to the same VPLS domain.
- Signaling: Establishes the pseudowires (virtual connections) between each pair of discovered PE routers.

VPLS discovery and signaling automate the creation of a full mesh of pseudowires among PE routers, enabling service providers to deliver native Ethernet access across a WAN.

Workflow

Figure 5: Figure 5. VPLS autodiscovery and signaling



VPLS discovery and signaling involve these stages:

1. Membership discovery: The VPLS PE routers advertise and learn which provider edge routers are members of the same VPLS domain.
2. Pseudowire establishment: After the PE routers are discovered, the control plane signals and sets up pseudowires between each pair of PE routers in the domain.
3. Operation: The VPLS domain operates with a full mesh of pseudowires across all participating PE routers.

Result

The VPLS service automatically builds the transport mesh required for the domain, avoiding manual neighbor provisioning and enabling seamless, scalable Ethernet connectivity across a WAN.

BGP-based VPLS autodiscovery

Provides an overview of how BGP-based autodiscovery distributes VPLS membership information and simplifies provisioning.

BGP-based VPLS autodiscovery enables network devices within a VPLS domain to automatically share membership information, eliminating manual neighbor provisioning. Each provider edge (PE) router can dynamically discover all other PE routers participating in the same VPLS domain. This mechanism ensures that when new routers join or leave the domain, the membership information is updated and distributed efficiently.

After the discovery process, each PE router receives the information needed to establish VPLS pseudowires and support multipoint connectivity. Even when BGP-based autodiscovery is used, operators retain the option to manually configure pseudowires for PE routers that do not participate in the automatic process.

The result is streamlined VPLS deployment, reduced operational effort, and improved scalability for multipoint VPN environments.

How BGP autodiscovery with BGP signaling work

Describes how BGP autodiscovery and BGP signaling collaborate to identify VPLS members and establish a full mesh of pseudowires using NLRI.

This process describes how BGP signaling and autodiscovery combine to establish the full mesh of pseudowires required for a VPLS instance automatically.

Summary

The key components involved in BGP autodiscovery with BGP signaling are:

- PE routers: Participate in the VPLS instance and exchange control-plane information.
- Autodiscovery mechanism: Identifies remote PE routers that are members of a given VPLS.
- Signaling mechanism: Advertises pseudowire labels expected by remote PE routers for the VPLS.
- BGP Network Layer Reachability Information (NLRI): Transports autodiscovery and signaling information together.

BGP autodiscovery and BGP signaling work together to automate the establishment of a VPLS pseudowire mesh. Autodiscovery identifies which remote PE routers are members of a VPLS instance, while signaling uses NLRI to distribute pseudowire labels and build the mesh without manual configuration.

Workflow

Figure 6: Discovery and signaling attributes



BGP autodiscovery with BGP signaling involves these stages:

1. PE routers exchange autodiscovery information so that each router learns which remote PE routers are members of the same VPLS.
2. Each PE router advertises the pseudowire label expected for the VPLS, enabling remote PE routers to learn the signaling information required.
3. BGP NLRI carries both autodiscovery and pseudowire signaling information simultaneously.
4. PE routers use the combined information to establish the full mesh of pseudowires for the VPLS, eliminating the need for manual configuration.

Result

The VPLS instance gains an automatically discovered and signaled pseudowire mesh, streamlining network configuration and scaling operations.

NLRI format for VPLS with BGP autodiscovery and signaling

Provides the NLRI format used for VPLS with BGP autodiscovery and signaling.

The NLRI format for a VPLS (Virtual Private LAN Service) with BGP autodiscovery and signaling defines how route information is structured and exchanged to enable VPLS operation. The format specifies fields and encoding used for identifying VPLS instances and communicating endpoint information.

Figure 7: NLRI format



The NLRI format typically includes these fields:

- Route type: Indicates the VPLS route type, such as autodiscovery or signaling.
- Length: Specifies the length in bytes of the NLRI payload.
- VPLS identifier: Uniquely identifies the VPLS instance.
- Endpoint attributes: Provides information about the endpoints or sites participating in the VPLS.

This format enables seamless setup and maintenance of a VPLS using BGP for autodiscovery and signaling by standardizing how relevant information is communicated between network devices.

How BGP autodiscovery with LDP signaling work

BGP autodiscovery with LDP signaling uses BGP to identify VPLS members and targeted LDP sessions to signal the pseudowire labels and attributes.

The process separates membership discovery and point-to-point signaling, allowing PE routers to build the VPLS transport mesh.

Summary

The key components involved in BGP autodiscovery with LDP signaling are:

- The PE routers: Advertise and receive VPLS membership information.
- BGP autodiscovery: Carries the VPLS identifier that tells each PE router which remote PE routers belong to the instance.
- Targeted LDP sessions: Exchange the pseudowire label values and attributes between the PE routers.
- FEC 129: Carries the VPLS ID, the Target Attachment Individual Identifier, and the Source Attachment Individual Identifier for pseudowire signaling.

This process describes how BGP identifies the PE routers in a VPLS instance and how LDP signals the pseudowires between those endpoints.

Workflow

Figure 8: Discovery and signaling attributes



BGP autodiscovery with LDP signaling involves these stages:

1. Each PE router advertises a VPLS identifier through BGP so the remote PE routers can identify which routers are members of the VPLS instance.
2. After membership is known, the PE routers use targeted LDP sessions to exchange label values and signaling attributes for the pseudowires.
3. FEC 129 carries the VPLS ID, the Target Attachment Individual Identifier, and the Source Attachment Individual Identifier so the remote PE routers can associate the pseudowire with the correct VPLS instance.
4. The LDP advertisement supplies the expected inner label for incoming traffic so each PE router can bind the pseudowire to the correct VPLS instance and use the proper label when sending traffic.

Result

PE routers identify VPLS membership through BGP and establish the pseudowires through LDP, enabling the creation of a VPLS transport mesh.

NLRI and extended communities for BGP autodiscovery with LDP signaling

Lists the Network Layer Reachability Information (NLRI) types and extended communities commonly used for BGP autodiscovery processes involving LDP signaling.

NLRI types used

Common extended communities

Example figure

The NLRI and extended communities are critical in BGP-based autodiscovery with LDP signaling, enabling efficient communication and label distribution across networks.

- Prefix NLRI: Indicates IP address prefixes relevant for route advertisement.
- VPN NLRI: Specifies VPN-related routes to distinguish traffic in MPLS VPN environments.
- Service-specific NLRI: Identifies particular services or network resources for BGP autodiscovery.
- Route Target: Defines VPN membership and import/export policies for routes.
- Site-of-Origin (SoO): Helps prevent routing loops and controls site-specific route distribution.
- Color: Used for traffic engineering, marking routes for path selection criteria.

For a visual representation of NLRI and extended communities, see:

Figure 9: Figure 9. NLRI and extended communities



CFM on VPLS

Outlines CFM implementation on VPLS, detailing components, supported offload types, timer values, operational restrictions, process flows, and configuration instructions for Ethernet connectivity fault management.

A CFM on VPLS feature is a service-level OAM feature that

- monitors a Layer 2 VPN running VPLS
- uses continuity-check and Y.1731 functions to detect and isolate faults, and
- reports end-to-end ethernet connectivity issues.

Table 17: Feature history table

Feature Name	Release Information	Feature Description
CFM on VPLS	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
CFM on VPLS	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Feature Name	Release Information	Feature Description
CFM on VPLS	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
CFM on VPLS	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, P100])(select variants only*) The CFM on VPLS feature is now extended to: • 8712-MOD-M • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E • 88-LC1-36EH
CFM on VPLS	Release 7.3.2	This feature helps you monitor and manage a Layer 2 VPN running VPLS. It does so by providing proactive network management, enabling fault detection and isolation, and reporting end-to-end ethernet connectivity issues.

CFM components and protocols for VPLS

Lists and describes the key CFM components and protocols used to monitor and manage Ethernet services in Virtual Private LAN Service (VPLS) environments.

The following are the key components of CFM:

- Maintenance Domain (MD): A management space for administering a network. A maintenance domain is owned and operated by a single entity and defined by the set of ports internal to the domain and at its boundary.
- Maintenance Association/Service (MA): Identifies a service that can be uniquely located within the maintenance domain. The maintenance association monitors the connectivity of a particular service instance in an MD.
- Maintenance Point: A demarcation point on an interface that participates in CFM within a maintenance domain. Two classes of maintenance points exist: Maintenance End Point (MEP) and Maintenance Intermediate Point.
- Maintenance End Point (MEP): Located at the edge of a domain, MEPs define the boundary for CFM messages. MEPs drop all lower-level frames and forward higher-level frames. MEPs are defined per maintenance domain level and service.

  The following MEPs are used:

  - Down MEP: Outward-facing MEPs that communicate through the wire, using the port MAC address. Often used for services spanning a single Layer 2 link.
  - Up MEP: Inward-facing MEPs that communicate through the bridge relay function. Up MEPs send and receive CFM frames at their level through the relay function, commonly used for services across multiple switches with end-to-end connections via xconnect Layer 2 AC interfaces.

CFM protocols used in VPLS include:

- Continuity Check protocol: Used for fault detection and notification, carrying the status of the port on which the MEP is configured.
- Loopback protocol: Used for fault verification.
- Linktrace protocol: Used for path discovery and fault isolation.

For more information about CFM, see the Configuring Ethernet OAM chapter in the *Interface and Hardware Component Configuration Guide for Cisco 8000 Series Routers*.

Restrictions for CFM on VPLS

Ensure that interface, MEP direction, and offload requirements are followed when configuring CFM on VPLS.

You must follow these restrictions when configuring CFM on VPLS:

- Do not configure Up MEPs and Down MEPs on the same interface.
- Multiple MEPs of the same direction are supported on the same interface.
- Maintenance Intermediate Point is not supported on any interface.
- MEPs on bundle member interfaces are limited to maintenance domain level 0.
- Software offload is supported only on bundle member Down MEPs.
- CFM does not work on Layer 2 subinterfaces with default encapsulation.

Offload types and supported CCM timers for CFM on VPLS interfaces

Provides the supported offload processing types and continuity check message (CCM) timer values for connectivity fault management (CFM) on VPLS interfaces.

Offload types and CCM timer
Table 18: Feature history table

Feature Name	Release Information	Feature Description
CFM Hardware Offload	Release 7.9.1	CFM Hardware offloading allows to implement connectivity and fault monitoring for physical and bundle interfaces, using continuity check messages (CCM). This feature helps to detect network failure with short CCM intervals, which enables the router to recover from the failure without dropping the packets.

Depending on where the continuity check messages are processed, offload is categorized into software offload, hardware offload, and non-offload. CCM intervals are the intervals in which CCMs are sent and received.

Depending on where the continuity check messages are processed, the offload method is categorized as software offload, hardware offload, or non-offload. CCM intervals define how frequently CCMs are sent and received.

Table 19: Supported CCM timers by offload type

Interface Type	Offload Type	Supported CCM Timers
Physical Interfaces and Subinterfaces	Non-offload	• 1 sec • 10 sec • 1 min • 10 min
Physical Interfaces and Subinterfaces	Hardware offload	• 3.3 ms • 10 ms
Bundle Members	Non-offload	1 sec; 10 sec; 1 min; 10 min • 1 sec • 10 sec • 1 min • 10 min
Bundle Members	Software offload	100 ms
Bundle Interfaces and Subinterfaces	Non-offload	• 1 sec • 10 sec • 1 min • 10 min
Bundle Interfaces and Subinterfaces	Hardware offload	• 3.3 ms • 10 ms

How CFM on VPLS works

Describes how CFM monitors connectivity across a VPLS mesh by exchanging continuity-check messages between provider edge (PE) routers.

This process uses a full-mesh VPLS topology to monitor connectivity between provider edge routers with CFM.

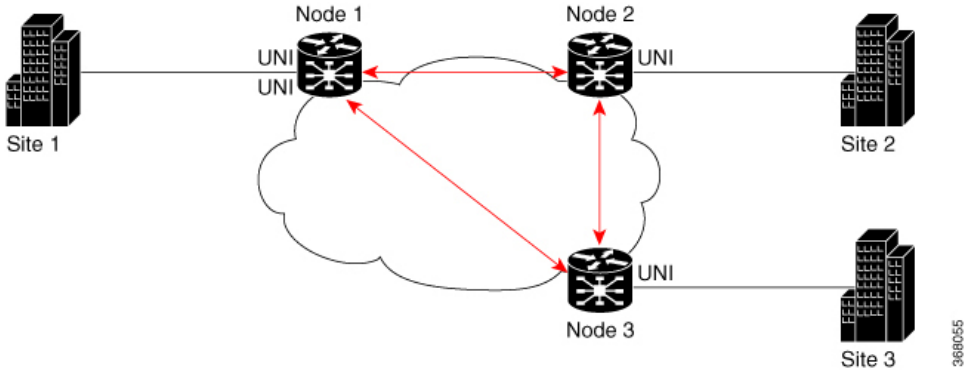
Summary

The key components involved in CFM on VPLS are:

- PE routers: Participate in the VPLS mesh and deliver the bridged service.
- CFM continuity checks: Monitor connectivity between the nodes.
- Maintenance end points (MEPs): Exchange CFM messages across the VPLS service.

CFM on VPLS verifies end-to-end connectivity across a VPLS mesh by sending continuity-check messages between routers participating in the service.

Workflow



CFM on VPLS involves these stages:

1. The PE routers form a VPLS mesh in which each node participates in the same bridged service.
2. The routers configure CFM continuity checks and MEP relationships for the VPLS service.
3. The MEPs exchange CFM messages so the routers can monitor connectivity and detect service faults across the mesh.

Result

The VPLS service provides monitored end-to-end connectivity between the PE routers, ensuring detection and management of service faults.

Configure CFM for VPLS services

Configure CFM continuity check, MEP cross-check, and interface-level CFM to ensure VPLS service health and verify peer and local MEP status.

Set up CFM for VPLS services, including continuity check, MEP cross-check, interface CFM, and verification of MEP states.

CFM is used to monitor and maintain Ethernet services. For details on CFM configuration and troubleshooting, see the *Configuring Ethernet OAM* chapter in the *Interface and Hardware Component Configuration Guide for Cisco 8000 Series Routers*.

Before you begin

Follow these steps to configure CFM for VPLS services:

Procedure

1. Enable CFM continuity check for the VPLS service.

   Example

```
Router# configure
Router# ethernet cfm
Router(config-cfm# domain vpls_bgp level 3 id null
Router(config-cfm-dmn)# service vpls_bgp_1 bridge group vpls bridge-domain
vpls-1 id number 1000
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

2. Configure MEP cross-check and logging.

Example

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-dmn-svc-xcheck)# mep crosscheck
Router(config-cfm-dmn-svc-xcheck)# exit
Router(config-cfm-dmn-svc)# log continuity-check errors
Router(config-cfm-dmn-svc)# log continuity-check mep changes
Router(config-cfm-dmn-svc)# commit
```

3. Enable CFM on the interface and configure the Up MEP.

Example

```
Router# configure
Router(config)# interface HundredGigE0/0/0/2/0.1000 l2transport
Router(config-subif)# encapsulation dot1q 1000
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain vpls_bgp service vpls_bgp_1 mep-id 1
Router(config-if-cfm-mep)# commit
```

4. Review the running configuration.

Example

```
ethernet cfm
 domain vpls_bgp level 3 id null
  service vpls_bgp_1 bridge group vpls bridge-domain vpls-1 id number 1000
   continuity-check interval 10s
   mep crosscheck
    mep-id 8191
   !
   log continuity-check errors
   log continuity-check mep changes
  !
!
!
interface HundredGigE0/0/0/2/0.1000 l2transport
 encapsulation dot1q 1000
 ethernet cfm
  mep domain vpls_bgp service vpls_bgp_1 mep-id 1
 !
```

5. Use **show ethernet cfm peer meps**, **show ethernet cfm peer meps detail**, and **show ethernet cfm local meps verbose** commands to verify peer and local MEP status.

Example

```
Router(PE1)# show ethernet cfm peer meps
Flags:
 > - Ok
 I - Wrong interval
 R - Remote Defect received
 V - Wrong level
 L - Loop (our MAC received)
 T - Timed out
 C - Config (our ID received)
 M - Missing (cross-check)
 X - Cross-connect (wrong MAID)
```

```
 U - Unexpected (cross-check)
 * - Multiple errors received
 S - Standby
Domain id_no (level 3) , Service id_no_vpws_1
Up MEP on TenGigE0/0/0/2/0.1 MEP-ID 1
==============================================================================
St ID MAC Address Port Up/Downtime CcmRcvd SeqErr RDI Error
-- ----- -------------- ------- ----------- --------- ------ ----- -----
> 8191 b0c5.3cff.c0c1 Up 00:01:26 9 0 4 0

Router(PE1)# show ethernet cfm peer meps detail
Domain id_no (level 3), Service id_no_vpws_1
Up MEP on TenGigE0/0/0/2/0.1 MEP-ID 1
==============================================================================
Peer MEP-ID 8191, MAC b0c5.3cff.c0c1
CFM state: Ok, for 00:01:44
Port state: Up
CCMs received: 11
Out-of-sequence: 0
Remote Defect received: 4
Wrong level: 0
Cross-connect (wrong MAID): 0
Wrong interval: 0
Loop (our MAC received): 0
Config (our ID received): 0
Last CCM received 00:00:04 ago:
Level: 3, Version: 0, Interval: 10s
Sequence number: 8495, MEP-ID: 8191
MAID: NULL, UINT: 1
Chassis ID: Local: Eyrie; Management address: 'Not specified'
Port status: Up, Interface status: Up

Router(PE1)# show ethernet cfm local meps verbose
Domain id_no (level 3), Service id_no_vpws_1
Up MEP on TenGigE0/0/0/2/0.1 MEP-ID 1
==============================================================================
Interface state: Up
MAC address: d46a.355c.b808
Peer MEPs: 1 up, 0 with errors, 0 timed out (archived)
Cross-check errors: 0 missing, 0 unexpected
CCM generation enabled: Yes, 10s (Remote Defect detected: No)
AIS generation enabled: No Sending AIS: No Receiving AIS: No
Sending CSF: No Receiving CSF: No
Packet Sent Received
------ ---------- ------------------------------------------------------------
CCM 8508 24 (out of seq: 0)
```

CFM continuity checking is successfully configured for the VPLS service. Verification commands confirm the expected state for peer and local MEPs.

Split-horizon groups

Introduces split-horizon group concepts, listing supported group types and providing configuration steps to prevent packet looping within VPLS bridge domains.

A split-horizon group is a bridge-domain grouping that

- prevents loops by applying forwarding and flooding restrictions between bridge ports based on group membership

- restricts traffic between members of the same group, and
- still allows traffic between different groups.

Table 20: Feature history table

Feature Name	Release Information	Feature Description
Split-Horizon Groups	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Split-Horizon Groups	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Split-Horizon Groups	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Split-Horizon Groups	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The split-horizon groups functionality is now extended to the Cisco 8712-MOD-M routers.
Split-Horizon Groups	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The split-horizon groups functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Split-Horizon Groups	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The split-horizon groups functionality is now extended to routers with the 88-LC1-36EH line cards.
Split-Horizon Groups	Release 7.3.2	This feature prevents packets from going into endless loops by aggregating attachment circuits and pseudowires into one of three groups called split-horizon groups. Split-horizon groups operate on the principle that members within a group will not send traffic to each other thereby preventing traffic loops.

Supported split-horizon groups

Lists the three split-horizon groups used by bridge domains, highlighting their forwarding and flooding behaviors.

Bridge domains use three split-horizon groups, each with distinct forwarding and flooding behavior. Traffic flooding applies to broadcast, multicast, and unknown unicast destination addresses. Unicast traffic consists of frames sent to bridge-ports where the destination MAC address is known.

Flooding traffic consists of:

- Unknown unicast destination MAC address frames
- Frames sent to Ethernet multicast addresses such as Spanning Tree BPDUs
- Ethernet broadcast frames with MAC address FF-FF-FF-FF-FF-FF

Bridge domain traffic is either unicast or flooding. Members within certain groups are forbidden to send traffic to each other, while members in different groups can send traffic to each other without restriction.

Table 21: Supported split-horizon groups

Split-Horizon Group	Behavior
0	Default AC group. There is no forwarding and flooding restrictions. Forwards and floods traffic within the group and between all groups. By default, all L2 ACs are added to this group and you cannot assign them manually through the CLI.
1	Default VFI core PW group. Forwarding or flooding of traffic is restricted between bridge ports in this group. Traffic to all other groups is allowed. All VFI pseudowires are added to this group and you cannot assign them manually through the CLI.
2	Optional AC group. Forwarding or flooding of traffic is restricted between bridge ports in this group. Traffic to all other groups is allowed. You can manually add ACs, and not VFI pseudowires, by using the CLI.

Configure split-horizon groups

Configure split-horizon groups on attachment circuits (ACs) and pseudowires, and verify the group assignments to ensure effective traffic isolation.

Assign split-horizon groups to ACs and pseudowires within a bridge domain to optimize traffic management and prevent unwanted loopbacks.

The bridge domain aggregates attachment circuits and pseudowires into one of three split-horizon groups, separating traffic and enhancing network stability.

Before you begin

Follow these steps to configure the split-horizon groups:

Procedure

1. Configure split-horizon groups on the attachment circuits and pseudowire.

Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg
Router(config-l2vpn-bg)# bridge-domain bd
Router(config-l2vpn-bg-bd-ac)# interface Ten0/7/0/22/0
Router(config-l2vpn-bg-bd-ac)# interface Ten0/7/0/22/1.1
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router(config-l2vpn-bg-bd-ac)# neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-bg-bd-pw)# split-horizon group
Router(config-l2vpn-bg-bd-pw)# vfi vf
Router(config-l2vpn-bg-bd-vfi)# neighbor 172.16.0.1 pw-id 10001
Router(config-l2vpn-bg-bd-vfi-pw)# commit
```

2. Review the running configuration.

Example

```
l2vpn
  bridge group bg
   bridge-domain bd
    int Ten0/7/0/22/0
    int Ten0/7/0/22/1.1
     split-horizon group
    neighbor 10.0.0.1 pw-id 1
     split-horizon group
    vfi vf
      neighbor 172.16.0.1 pw-id 10001
!
```

3. Use the **show l2vpn bridge-domain detail | i "AC:|Split Horizon|PW:|VFI** command to verify the split-horizon group assignments.

Example

```
Router# show l2vpn bridge-domain detail | i "AC:|Split Horizon|PW:|VFI"
MAC withdraw for Access PW: enabled
Split Horizon Group: none
AC: Ten0/7/0/22/0, state is up
Split Horizon Group: none
AC: Ten0/7/0/22/1, state is up
Split Horizon Group: enabled
PW: neighbor 10.0.0.1, pw-id 1, state is up ( established )
Split Horizon Group: enabled
VFI vf (up)
PW: neighbor 172.16.0.1, pw-id 10001, state is up ( established )
Split Horizon Group: none
```

The bridge-domain ACs and pseudowires are successfully assigned to the intended split-horizon groups, optimizing network traffic and preventing unwanted loopbacks.

Traffic storm control

Details traffic storm control features, covering operational behavior, supported traffic types, restrictions, configuration procedures, and process flows for mitigating excessive traffic in VPLS environments.

A traffic storm control feature is a bridge-port traffic suppression feature that

- monitors incoming traffic levels over a one-second hardware interval
- drops traffic when it reaches configured thresholds, and
- helps prevent excess traffic from disrupting the VPLS bridge.

Table 22: Feature history table

Feature Name	Release Information	Feature Description
Traffic Storm Control	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Traffic Storm Control	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *The traffic storm control functionality is now extended: • 8011-32Y8L2H2FH • 8011-12G12X4Y-D/A
Traffic Storm Control	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Traffic Storm Control	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The traffic storm control functionality is now extended to the Cisco 8712-MOD-M routers.
Traffic Storm Control	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The traffic storm control functionality is now extended: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
Traffic Storm Control	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The traffic storm control functionality is now extended to routers with the 88-LC1-36EH line cards.
Traffic Storm Control	Release 7.3.2	This feature monitors incoming traffic levels on a port in the VPLS bridge. It drops traffic when the number of packets reaches the configured threshold level, thus preventing packets from flooding the VPLS bridge and creating excessive traffic and degrading network performance.

Storm control on a VPLS bridge

Provides key facts and configuration details about storm control to protect a VPLS bridge from excessive Layer 2 traffic and improve port security.

Storm control protects a VPLS bridge from excessive Layer 2 traffic on an access circuit (AC). When multiple hosts or routers connect to the same LAN, protocol traffic rates can increase, creating security risks and potential disruptions. Packet floods to a VPLS bridge may degrade network performance and disrupt bridge operation—this condition is known as a traffic storm.

Storm control suppresses traffic when packet levels reach configured threshold values. This helps prevent disruption of the VPLS bridge and enhances Layer 2 port security.

Key details include:

- Storm control limits excessive traffic on an AC under a VPLS bridge.
- Configurable threshold levels determine when traffic suppression begins.
- Separate threshold levels can be set for different traffic types.
- The feature helps reduce traffic storms and protect network performance.
- Use the **storm-control** command to enable storm control on the VPLS bridge.

Supported traffic types

Storm control supports broadcast, multicast, and unknown-unicast thresholds on each VPLS bridge port.

On each VPLS bridge port, both packet-per-second and bits-per-second rate are supported. You can configure up to three storm control thresholds, one for each supported traffic type. If you do not configure a threshold for a traffic type, storm control is not enabled on that port for that traffic type.

- Broadcast traffic: Packets with a destination MAC address equal to FFFF.FFFF.FFFF.
- Multicast traffic: Packets with a destination MAC address not equal to the broadcast address, but with the multicast bit set to 1.
- Unknown unicast traffic: Packets with a destination MAC address not yet learned.

How traffic storm control works

Traffic storm control monitors traffic during a fixed one-second interval and drops traffic that exceeds the configured threshold.

A traffic storm occurs when excessive packet flooding on a LAN increases protocol traffic and degrades network performance. Traffic storm control helps protect the VPLS bridge and provides Layer 2 port security on the access circuit.

Summary

The key components involved in traffic storm control are:

- The ingress port: Receives the incoming traffic that the router monitors.
- The router: Measures the traffic level against the configured storm-control threshold.
- The storm-control threshold: Defines the allowed traffic rate for the configured traffic type on the bridge port.

Traffic storm control evaluates incoming traffic continuously and enforces the configured limit during each monitoring interval.

Workflow

Traffic storm control involves these stages:

1. The router monitors incoming traffic levels on a port during a fixed one-second hardware interval.
2. During that interval, the router compares the measured traffic level with the storm-control threshold configured for the bridge port and traffic type.
3. When the incoming traffic reaches the configured threshold, the router drops traffic of that type until the end of the current interval.
4. At the beginning of the next interval, the router allows traffic of the specified type again and repeats the monitoring cycle.

Result

Traffic storm control limits excess traffic on the bridge port and helps prevent a traffic storm from disrupting the VPLS bridge.

Feature behavior for traffic storm control

Traffic storm control operates on bridge ports and has subinterface, policing, and bundle-member behavior that affects enforcement.

- Storm control configuration is allowed at the bridge port level.
- Configuration on a Layer 2 subinterface applies to the main interface as well.
- Policer is applied to the main port. Thus, policing applies to all subinterfaces that are within a bridging service.
- When multiple subinterfaces of a common Ethernet port have storm control configured, only one subinterface is used for storm control configuration. Usually, the first subinterface that is added is picked up.
- When applied to a bundle AC, policing occurs independently on each main port that is a member of the bundle. Aggregate BUM traffic can be up to the configured rate times the number of bundle members.

Supported traffic types

Storm control supports broadcast, multicast, and unknown-unicast thresholds on each VPLS bridge port.

On each VPLS bridge port, both packet-per-second and bits-per-second rate are supported. You can configure up to three storm control thresholds, one for each supported traffic type. If you do not configure a threshold for a traffic type, storm control is not enabled on that port for that traffic type.

- Broadcast traffic: Packets with a destination MAC address equal to FFFF.FFFF.FFFF.

- Multicast traffic: Packets with a destination MAC address not equal to the broadcast address, but with the multicast bit set to 1.
- Unknown unicast traffic: Packets with a destination MAC address not yet learned.

Restrictions for traffic storm control

Outlines the key operational and support restrictions for configuring and enforcing traffic storm control.

Traffic storm control is subject to important restrictions to ensure predictable and supported bridging behavior. Follow these requirements:

- Use traffic storm control only on supported bridge-port configurations.
- Storm control supports main ports only.
- Storm control configuration operates at the bridge-port level, not the bridge-domain level.
- Storm control does not support pseudowire-level operations.
- Do not assume the hardware applies the exact configured rate.
- Storm control converts packets-per-second (PPS) configurations to a kbps value, assuming a 256-byte packet size.
- When you configure the storm control rate at the attachment circuit level, hardware rate limiter values may deviate by up to five percent from the configured value.

Configure traffic storm control

Configure storm control on a bridge-domain attachment circuit and review the resulting statistics behavior.

Enable and verify storm control on bridge-domain attachment circuits to manage broadcast traffic and observe drop statistics.

Storm control is disabled by default on attachment circuits and must be explicitly enabled for each traffic type. Storm control statistics are only available on line cards based on Q200 silicon.

Before you begin

Follow these steps to configure storm control on the attachment circuit:

Procedure

1. Configure storm control on the attachment circuit.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# storm-control broadcast kbps 4500
Router(config-l2vpn-bg-bd-ac)# commit
```

2. Review the running configuration.

Example

```
configure
 l2vpn
  bridge group BG1
   bridge-domain BD1
    interface HundredGigE0/0/0/0
     storm-control broadcast kbps 4500
    !
```

3. Review the source storm-control statistics guidance.

Example

```
Storm control statistics are present as part of the AC bridging statistics
only on the AC where storm control is configured.
The storm control statistics are the aggregate drop statistics across all ACs
 that share the same main port.
```

Storm control statistics are not available on Line Card based on Q100 Silicon.

Storm control is successfully configured on the bridge-domain attachment circuit, and you can verify drop statistics if using a Q200 silicon line card.

GTP load balancing

Explains GTP load balancing, describing the GTP protocol's operation, behavioral patterns, and guidelines for optimizing traffic distribution within Layer 2 bridging contexts.

A GTP load-balancing feature is a transit-router hashing feature that

- adds the GTP tunnel endpoint identifier to the load-balancing calculation
- uses TEID-based entropy for GTP traffic with otherwise limited unique fields, and
- distributes traffic more evenly across equal-cost links and bundles.

> **Note**
>
> The Cisco 8010 Series Routers do not support this feature. For a list of supported features on the Cisco 8010 Series Routers, see *Compatibility Matrix for Cisco 8010 Series Routers*.

Table 23: Feature history table

Feature Name	Release Information	Feature Description
GTP Load Balancing	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.

Feature Name	Release Information	Feature Description
GTP Load Balancing	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8711-48Z-M routers.
GTP Load Balancing	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8712-MOD-M
GTP Load Balancing	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
GTP Load Balancing	Release 7.3.2	In addition to the source IP address, destination IP address, and port number, this functionality enables using the unique tunnel endpoint identifier (TEID) to compute load balancing (or hashing) of traffic in tunnels between endpoints. The load balancing occurring at the TEID is unique for each traffic flow and achieves better distribution of traffic over equal-cost links. It also helps in load balancing GTP traffic over bundles at transit routers. By default, this functionality is enabled on the Cisco 8000 Series routers, and you cannot disable it.

Key attributes of GTP

Provides key features and functions of the GTP protocol used in wireless networks.

The GPRS Tunneling Protocol (GTP) is a tunnel control and management protocol used by General Packet Radio Service (GPRS) support nodes in wireless networks. GTP enables signaling and user-data transport, especially for delivering mobile data.

Key features of GTP include:

- Tunnel management: GTP-C specifies control and management procedures for creating, modifying, and deleting tunnels between GPRS support nodes.
- User data transportation: GTP-U uses a tunneling mechanism to carry user data packets efficiently across the wireless network.
- Signaling and data separation: GTP distinguishes between control messages for signaling and actual user-data transfer, enhancing network flexibility and scalability.

How GTP load balancing works

Describes how GTP load balancing uses the TEID and other packet fields to improve traffic distribution when the usual Layer 3 and Layer 4 fields do not provide enough entropy.

This process describes how the router improves load balancing for GTP-U traffic in mobile transport networks, ensuring more even distribution across equal-cost links.

Summary

The key components involved in GTP load balancing are:

- Transit router: Evaluates packet fields and calculates the load-balancing hash for traffic distribution.
- Standard Layer 3 and Layer 4 hash fields: Provide the baseline inputs for the hash calculation.
- GTP tunnel endpoint identifier (TEID): Adds entropy for GTP-U traffic when the standard fields are not unique enough.

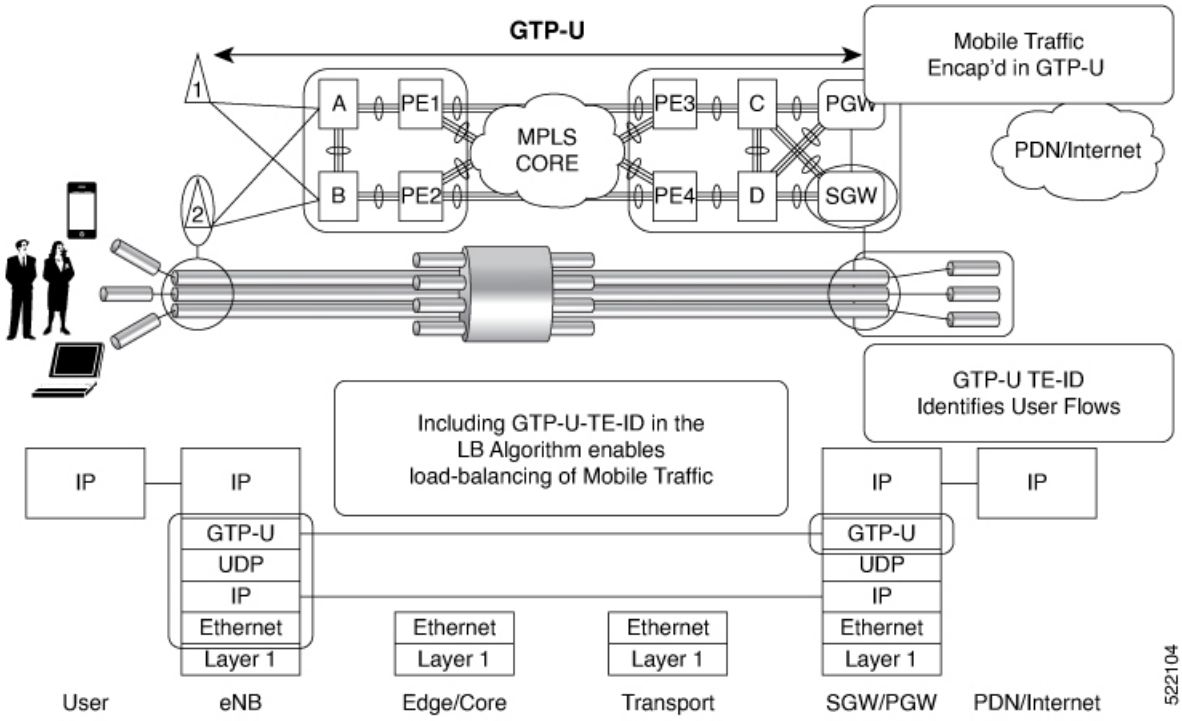
GTP load balancing enables routers to distribute traffic more effectively, especially when traditional Layer 3 and Layer 4 fields are not sufficient for creating unique traffic flows.

Table 24: Global Layer 3 flow-based load-balancing fields

Source address
Destination address
Router ID
Source port
Destination port

These fields are not always unique enough for GTP traffic, so the router uses the GTP tunnel endpoint identifier to improve traffic distribution.

Workflow



PGW – Packet Data Network Gateway
SGW – Serving Gateway
eNB - evolved Node B

GTP load balancing involves these stages:

1. The router evaluates the standard Layer 3 flow-based load-balancing fields for the incoming packet.
2. If the packet is TCP or UDP and the destination port is the GTP-U port 2152, the router includes the GTP TEID in the hash computation to add entropy for each traffic flow.
3. If the TEID is present, the router supports tunnel-endpoint load balancing for GTP version 1 and version 2 packets. For GTP version 0, the router relies only on the standard fields because version 0 does not include a TEID.

Result

The router distributes GTP-U traffic more evenly across equal-cost links. GTP-C packets, which use destination port 2123, continue to use only the global Layer 3 flow-based load balancing.

Guidelines for GTP load balancing

GTP load balancing applies only to supported packet types, destination ports, and encapsulation fields.

Use GTP load balancing only for supported packet and encapsulation combinations.

- GTP load balancing is supported only when the UDP or TCP destination port is 2152.
- GTP load balancing is performed on IPv4 or IPv6 incoming packets with GTP payloads.
- For MPLS packets with GTP payload, the load-balancing hash is based on the label stack and the GTP TEID. The maximum limit on the label stack is 14.
- For IPv4 packets with GTP payload, the hash is based on Router ID, source IP, destination IP, Layer 4 protocol field, source port, destination port, and GTP TEID.
- For IPv6 packets with GTP payload, the hash is based on Router ID, source IP, destination IP, flow label, Layer 4 protocol field, source port, destination port, and GTP TEID.

Do not expect to disable this feature or use it on a GPRS tunnel originator:

- GTP load balancing is enabled by default and cannot be disabled.
- For GTP hashing, the Cisco 8000 Series routers are transit routers but not GPRS tunnel originators.

8 Pseudowire over MPLS and Point-to-Point Service Modes

Topics:

- Point-to-point Layer 2 services
- Pseudowire over MPLS
- Pseudowire modes over MPLS
- Local switching between attachment circuits

Outlines the concepts, requirements, configuration, and operational modes for deploying pseudowire over MPLS and point-to-point Layer 2 services, including supported topologies, service modes, redundancy, and local switching between attachment circuits.

Point-to-point Layer 2 services

Introduces point-to-point Layer 2 services, detailing local switching mechanisms, attachment circuit configurations, and pseudowire concepts for connecting remote sites over MPLS infrastructure.

A point-to-point Layer 2 service is a transport circuit emulation that

- makes two end nodes appear directly connected over a point-to-point link
- connects two sites by using local switching, attachment circuits, or pseudowires, and
- is also called an MPLS Layer 2 VPN service.

Local switching

Explains local switching as a point-to-point internal circuit on a router and distinguishes it from pseudowire transport across an MPLS network.

Local switching is a point-to-point internal circuit that

- switches Layer 2 data between two attachment circuits on the same device
- keeps the switching function within a single router, and
- is also known as local connect.

Attachment circuits

Explains attachment circuits as CE-facing connection points that link CE devices to provider edge devices in point-to-point Layer 2 services.

Attachment circuits are physical or logical ports or circuits that

- connect a CE device to a provider edge devices
- provide the local service endpoint for a pseudowire or local switching connection, and
- must be active for the related Layer 2 service to forward traffic.

Pseudowires

Explains pseudowires as virtual point-to-point circuits between provider edge routers and positions them as MPLS-backed transport for Layer 2 services.

A pseudowire is a virtual point-to-point circuit that:

- connects one provider edge router to another provider edge router
- links the attachment circuits connected at each provider edge router over an MPLS network, and
- transports Layer 2 service traffic across a packet switched network.

Pseudowire over MPLS

Describes pseudowire over MPLS operation, covering route processor failover requirements, traffic transport topologies, static cross-connect circuit functionality and requirements, as well as procedures for configuring static and dynamic point-to-point cross-connects.

A pseudowire over MPLS is a tunneling mechanism, also known as Ethernet-over-MPLS, that

- connects two provider edge devices and links the attachment circuits at each provider edge device over the MPLS network
- provides a common intermediate format to transport IPv4, IPv6, MPLS, and Ethernet services over a packet switched network, and
- encapsulates Ethernet protocol data units with MPLS labels to forward them through an MPLS-enabled Layer 3 core network.

A pseudowire (PW) is a point-to-point connection between two provider edge (PE) devices that connects two attachment circuits (ACs).

A packet switched network (PSN) is a network that forwards packets, such as IPv4, IPv6, MPLS, and Ethernet.

Ethernet protocol data units (PDUs) are encapsulated by pseudowire over MPLS with MPLS labels before they are forwarded across the MPLS network.

Feature history
Table 25: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire over MPLS	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Pseudowire over MPLS	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Pseudowire over MPLS	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8712-MOD-M

Feature Name	Release Information	Feature Description
Pseudowire over MPLS	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700);Modular Systems (8800 [ASIC: P100]) (select variants only*) *This feature is now supported on: • 8212-32FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 88-LC1-36EH
Pseudowire over MPLS	Release 7.3.15	This feature allows you to tunnel two L2VPN Provider Edge (PE) devices to transport L2VPN traffic over an MPLS core network. MPLS labels are used to transport data over the pseudowire.

Limitations for Pseudowire over MPLS

Ensure proper handling of LDP-signaled pseudowire sessions during route processor failover when logging pseudowires is configured under Layer 2 VPN services with MPLS.

During route processor failover with logging pseudowire configured under Layer 2 VPN services, you must ensure these requirements:

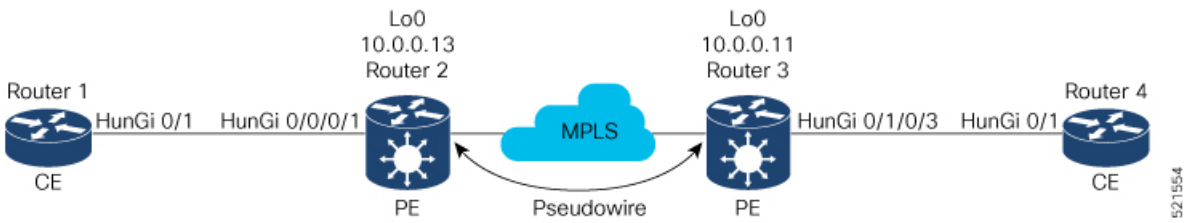
- Cisco IOS XR software displays syslog messages indicating that LDP-signaled Layer 2 VPN VPWS pseudowire sessions temporarily go down and then return to operational state.
- The session changes do not affect traffic forwarding, because pseudowire forwarding remains programmed in hardware.
- Traffic continues to flow despite any temporary syslog messages about session status.

How pseudowire over MPLS works

Describes how Layer 2 VPN traffic moves across pseudowire over MPLS, including attachment circuits, provider edge devices, MPLS labels, and remote delivery.

The source topology shows how Layer 2 VPN traffic is transported using pseudowire over MPLS.

Figure 10: Pseudowire over MPLS topology



Summary

The key components involved in the process are:

- CE devices: Connect to provider edge devices by using attachment circuits.
- Attachment circuits: Carry Layer 2 traffic between the CE device and the provider edge device.
- Provider edge devices: Host the pseudowire and manage MPLS labels.
- MPLS core network: Carries the labeled pseudowire traffic between provider edge devices.

Layer 2 VPN traffic is transported across an MPLS network using pseudowire technology, connecting customer edge devices through provider edge routers and labeled tunnels to deliver traffic efficiently and securely.

Workflow

The pseudowire over MPLS traffic transport process involves these stages:

1. Router 1 sends traffic to Router 2 through the attachment circuit.
2. Router 2 adds the MPLS pseudowire label and sends the traffic to Router 3 through the pseudowire.
3. Each provider edge device uses an MPLS label to reach the remote provider edge loopback.
4. The provider edge device can learn the Interior Gateway Protocol label through MPLS Label Distribution Protocol or MPLS Traffic Engineering.
5. One provider edge device advertises the MPLS label to the other provider edge device for pseudowire identification.
6. Router 3 identifies traffic with the MPLS label and removes the label.
7. Router 3 sends the traffic to the attachment circuit that connects to Router 4.

Result

The Layer 2 VPN traffic reaches the remote attachment circuit through pseudowire over MPLS. The point-to-point connection can use static or dynamic configuration.

What's next

Configure a static or dynamic point-to-point cross-connect based on the deployment requirements.

How static cross-connect circuits work

Describes how static point-to-point cross-connect circuits become active without control-plane signaling and why manual end-to-end service verification is required after configuration.

Static point-to-point connections using cross-connect circuits in a Layer 2 VPN do not use control-plane signaling and do not depend on remote configuration.

Summary

Static cross-connect circuits provide point-to-point connections in a Layer 2 VPN without control-plane signaling or reliance on remote configuration. The key issues affecting static connections are:

- The active status cannot validate the remote provider edge configuration.
- The static connection does not know the remote attachment circuit or virtual circuit status.
- The pseudowire can show UP locally even if traffic is lost.

Workflow

These stages describe how static cross-connect circuits become active and why operational verification remains necessary.

1. The local attachment circuit becomes active on the provider edge router.
2. MPLS data-plane reachability exists to the neighbor IP address.

3. The router marks the static cross-connect circuit active based on local and transport readiness.
4. The operator manually verifies end-to-end connectivity because IP reachability alone does not prove service health.

Result

The static connection forwards traffic after local and transport conditions are met. However, operators must manually verify end-to-end service to avoid traffic loss.

Requirement: Static cross-connect circuit readiness

Outlines the configuration requirements and operational limitations that must be met before static point-to-point cross-connect circuits are deployed in Layer 2 VPNs.

You must verify both local configuration readiness and transport readiness before relying on a static point-to-point cross-connect circuit.

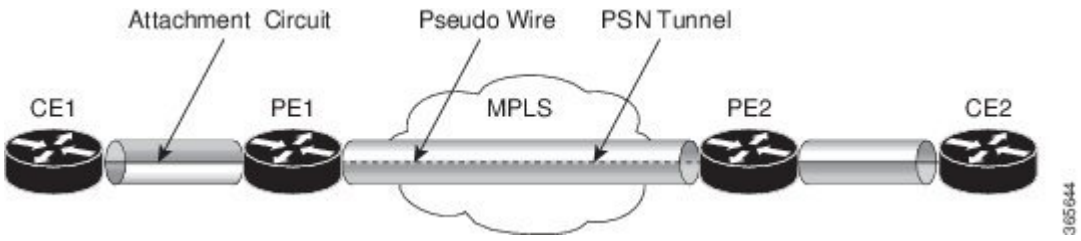
- Configure the customer edge and provider edge routers so they operate in the network.
- Use a cross-connect circuit name that identifies a pair of provider edge routers and is unique within the cross-connect group.
- Assign each segment, attachment circuit, or pseudowire to only one cross-connect circuit.
- Use a globally unique static virtual-circuit local label in only one pseudowire.
- Manually match pseudowire IDs, MPLS labels, and related parameters at both ends to prevent traffic issues.
- Static pseudowire connections do not use LDP for signaling; ensure all signaling-related parameters are explicitly configured.

Static cross-connect circuit topology

Provides the topology reference used to configure static cross-connect circuits in a Layer 2 VPN between two provider edge routers.

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

Figure 11: Static cross-connect circuits in a Layer 2 VPN



Configure static point-to-point cross-connects

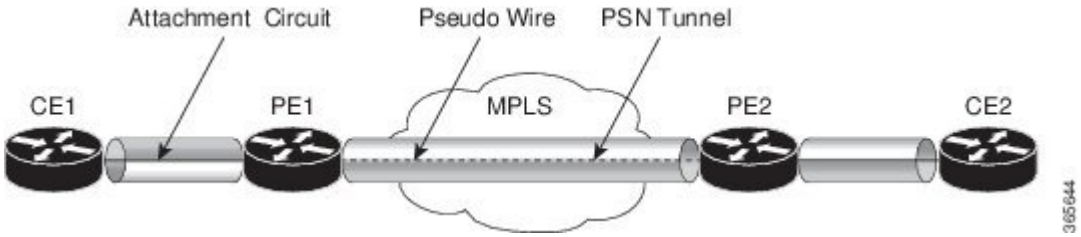
Configure static point-to-point cross-connect circuits in a Layer 2 VPN and verify that each provider edge router reports the cross-connect state as up.

Use this task to configure static point-to-point cross-connects in a Layer 2 VPN.

Static point-to-point cross-connects use manually matched pseudowire IDs and MPLS labels on each provider edge router.

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

Figure 12: Static cross-connect circuits in a Layer 2 VPN



Procedure

1. Configure the static cross-connect circuit on PE1.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigEt0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.3 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 50 remote 40
Router(config-l2vpn-xc-p2p-pw)# commit
```

2. Configure the static cross-connect circuit on PE2.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/2/0/0.4
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.4 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 40 remote 50
Router(config-l2vpn-xc-p2p-pw)# commit
```

3. Review the running configuration on PE1 and PE2.

   Example

```
On PE1
!
l2vpn
 xconnect group XCON1
  p2p xc1
   interface HundredGigE0/1/0/0.1
   neighbor ipv4 10.0.0.3 pw-id 100
    mpls static label local 50 remote 40
!

On PE2
!
l2vpn
 xconnect group XCON2
  p2p xc1
   interface HundredGigE0/2/0/0.4
   neighbor ipv4 10.0.0.4 pw-id 100
```

```
    mpls static label local 40 remote 50
  !
```

4. Use the **show l2vpn xconnect** command to verify the static cross-connect circuit on PE1.

   Example

```
Router# show l2vpn xconnect
Tue Apr 12 20:18:02.971 IST
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description
      ST
------------------------   ------------------------------
------------------------------
XCON1      xc1        UP   Hu0/1/0/0.1            UP       10.0.0.3  100
UP
------------------------------------------------------------------------------------------
```

5. Use the **show l2vpn xconnect** command to verify the static cross-connect circuit on PE2.

   Example

```
Router# show l2vpn xconnect
Tue Apr 12 20:18:02.971 IST
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description
      ST
------------------------   ------------------------------
------------------------------
XCON2      xc1        UP   Hu0/2/0/0.4            UP       10.0.0.4  100
UP
------------------------------------------------------------------------------------------
```

The static cross-connect circuit is configured on both provider edge routers, and the verification output shows the cross-connect state as UP.

Configure dynamic point-to-point cross-connects

Configure dynamic point-to-point cross-connects in a Layer 2 VPN where Label Distribution Protocol is available and operational for pseudowire signaling.

Use this task to configure a dynamic point-to-point cross-connect, allowing flexible Layer 2 VPN connectivity using signaling rather than static label configuration.

Dynamic cross-connects use signaling protocols, such as LDP, instead of static label configuration for establishing point-to-point circuits within Layer 2 VPNs.

Before you begin

For dynamic cross-connects, Label Distribution Protocol must be up and running.

Procedure

1. Configure the dynamic point-to-point cross-connect.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group vlan_grp_1
Router(config-l2vpn-xc)# p2p vlan1
Router(config-l2vpn-xc-p2p)# interface HunGigE 0/0/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

2. Review the running configuration.

   Example

```
configure
  l2vpn
   xconnect group vlan_grp_1
    p2p vlan1
    interface HunGigE 0/0/0/0.1
    neighbor 10.0.0.1 pw-id 1
!
```

The dynamic point-to-point cross-connect is configured and appears in the running configuration.

Pseudowire modes over MPLS

Explains available pseudowire over MPLS service modes, including Ethernet port, VLAN, VLAN passthrough, redundancy, and Inter-AS modes; outlines configuration steps and requirements for each mode, including redundancy and manual switchover procedures.

A pseudowire mode over MPLS is a transport option that

- defines how the provider edge router handles Ethernet frames and VLAN tags across the pseudowire
- supports Ethernet port mode over a virtual connection type 5 pseudowire, and
- supports VLAN mode and VLAN passthrough mode over virtual connection type 4 pseudowires.

Ethernet port mode

Explains Ethernet port mode as the pseudowire over MPLS mode that tunnels a port and transports VLAN manipulation over a virtual connection type 5 pseudowire.

Ethernet port mode is a pseudowire over MPLS mode that

- connects both ends of a pseudowire to Ethernet ports
- tunnels the port over a virtual connection type 5 pseudowire, and
- transports VLAN manipulation over the type 5 pseudowire, whether tagged or untagged.

Feature history
Table 26: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire VC Type 5	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Pseudowire VC Type 5	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8712-MOD-M
Pseudowire VC Type 5	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700);Modular Systems (8800 [ASIC: P100]) (select variants only*) *This feature is now supported on: • 8212-32FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 88-LC1-36EH
Pseudowire VC Type 5	Release 7.3.15	With this feature, Ethernet port mode is supported for pseudowire over MPLS. The virtual connection (VC) type 5 is known as an Ethernet port-based PW. In this mode, both ends of a pseudowire are connected to Ethernet ports and allow a complete ethernet trunk to be transported. The ingress PE transports frames received on a main interface or subinterface. This feature nullifies the need for a dummy tag and reduces overhead. In addition, frame tagging is no longer necessary.

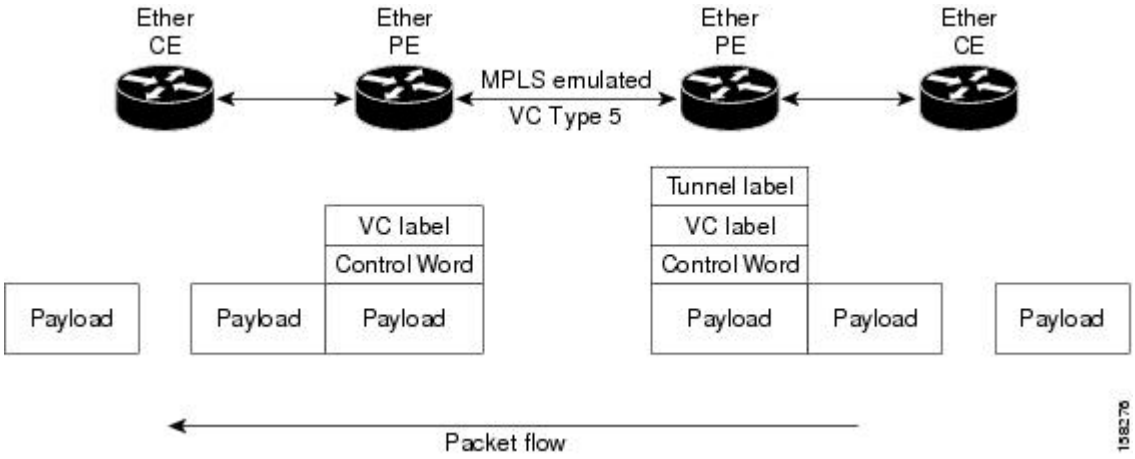
Ethernet port mode packet flow

The ingress provider edge router transports frames received on a main interface.

The ingress provider edge router can also transport frames after subinterface tags are removed.

Subinterface tags are removed when the packet is received on a subinterface.

Figure 13: Ethernet port mode packet flow



Configure Ethernet port mode

Configure Ethernet port mode for pseudowire over MPLS and verify that the detailed Layer 2 VPN output reports PW type Ethernet.

Use this task to configure Ethernet port mode for pseudowire over MPLS.

Ethernet port mode uses a virtual connection type 5 pseudowire to create a Layer 2 VPN across MPLS.

Before you begin

Confirm the provider edge interfaces and neighbor pseudowire IDs before you configure the service.

Procedure

1. Configure Ethernet port mode on PE1.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/0/0/1.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.11 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# commit
```

2. Configure Ethernet port mode on PE2.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/3.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.13 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# commit
```

3. Review the Ethernet port mode running configuration.

   Example

```
PE1 configuration
l2vpn
 xconnect group grp1
  p2p xc1
   interface HundredGigE0/0/0/1.2
   neighbor 10.0.0.11 pw-id 222

PE2 configuration
l2vpn
 xconnect group grp1
  p2p xc1
   interface HundredGigE0/1/0/3.2
   neighbor 10.0.0.13 pw-id 222
```

4. Use the **show l2vpn xconnect group grp1 detail** command to verify the Ethernet port mode configuration.

   Example

```
Router# show l2vpn xconnect group grp1 detail
Group grp1, XC xc1, state is up; Interworking none
  AC: HundredGigE0/0/0/1.2, state is up
    Type VLAN; Num Ranges: 1
    VLAN ranges: [2, 2]
    MTU 1504; XC ID 0x840006; interworking none
  PW: neighbor 10.0.0.11, PW ID 222, state is up ( established )
    Encapsulation MPLS, protocol LDP
    Source address 10.0.0.13
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    PW Status TLV in use
      MPLS         Local                          Remote
      Label        16026                          16031
      Interface    HundredGigE0/0/0/1.2           HundredGigE0/1/0/3.2
      MTU          1504                           1504
      PW type      Ethernet                       Ethernet
```

   The PW type Ethernet value indicates a virtual connection type 5 pseudowire.

The Ethernet port mode pseudowire is configured, and verification output reports PW type Ethernet.

VLAN mode

Explains VLAN mode as the pseudowire over MPLS mode that uses virtual connection type 4 to transport VLAN-based traffic with VLAN tag handling.

VLAN mode is a pseudowire over MPLS mode that

- configures each VLAN on a CE-to-PE link as a separate Layer 2 VPN connection by using virtual connection type 4
- transports a VLAN tag over the pseudowire by pushing a dummy tag at the attachment-circuit ingress, and
- removes the dummy tag before egress on the remote router when the dummy tag has been added.

Feature history

Table 27: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire VC Type 4	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Pseudowire VC Type 4	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Pseudowire VC Type 4	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) • 8011-4G24Y4H-I • 8712-MOD-M
Pseudowire VC Type 4	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700);Modular Systems (8800 [ASIC: P100]) (select variants only*) • 8212-32FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 88-LC1-36EH
Pseudowire VC Type 4	Release 7.3.15	With this feature, VLAN mode is supported for pseudowire over MPLS. A virtual connection (VC) type 4 is the VLAN-based PW. The ingress PE does not remove the incoming VLAN tags that are to be transported over the PW. VC type 4 inserts an extra dummy tag with VLAN 0 onto the frame which is removed on the other side. This mode helps the service provider to segregate traffic for each customer based on the VLAN.

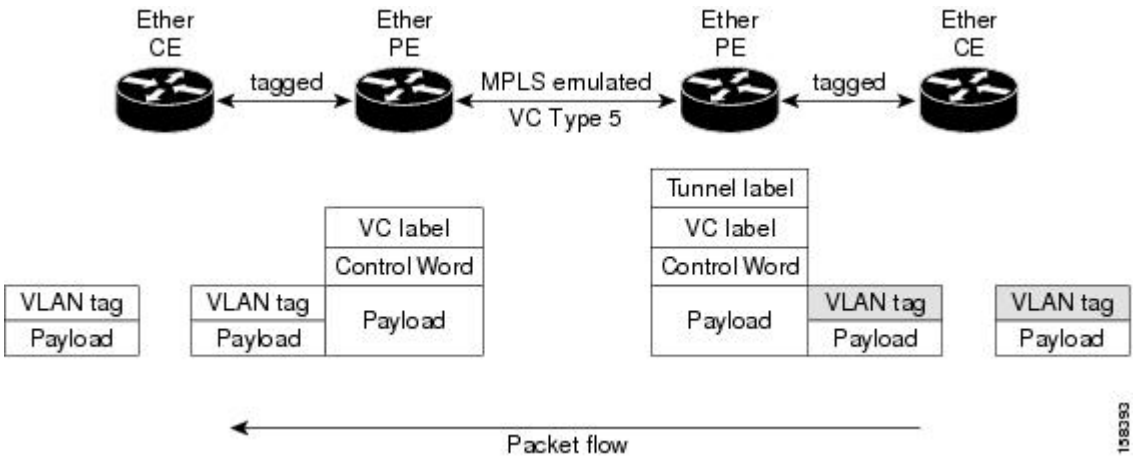
VLAN mode packet flow

The Ethernet provider edge router associates an internal VLAN tag to the Ethernet port.

The internal VLAN tag switches traffic internally from the ingress port to the pseudowire.

The provider edge router removes the internal VLAN tag before moving traffic into the pseudowire.

Figure 14: VLAN mode packet flow



At the egress VLAN provider edge router, the provider edge router associates a VLAN tag to the frames coming out of the pseudowire.

The provider edge router sends traffic on an Ethernet trunk port after it switches the traffic internally.

Note

Because the port is in trunk mode, the VLAN provider edge router does not remove the VLAN tag.

It forwards the frames through the port with the added tag.

VLAN mode TPID handling

Outlines how VLAN mode handles Tag Protocol Identifier values for dummy VLAN tags on imposition and disposition provider edge routers.

Account for Tag Protocol Identifier (TPID) handling whenever VLAN mode adds or removes a dummy VLAN tag. Follow these requirements:

- On the pseudowire imposition provider edge router, the pushed TPID of the dummy VLAN tag uses the innermost popped VLAN tag TPID.
- If no VLAN tag is popped on the ingress Layer 2 interface, the Tag Protocol Identifier on the dummy VLAN is 0x8100.
- On the disposition provider edge router, egress VLAN tag push copies the TPID of the dummy VLAN tag to the innermost pushed VLAN tag.
- If no egress VLAN push is configured on the egress Layer 2 interface, the dummy VLAN tag is discarded.

Configure VLAN mode for pseudowire over MPLS

Configure VLAN mode for a pseudowire over MPLS and verify that Layer 2 VPN output reports PW type Ethernet VLAN.

Use this task to configure VLAN mode for pseudowire over MPLS.

VLAN mode uses a pseudowire class with MPLS encapsulation and transport-mode VLAN.

Before you begin

Review the VLAN mode Tag Protocol Identifier handling requirement before you configure the service.

Procedure

1. Configure VLAN mode under the pseudowire class and xconnect.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class VLAN
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode vlan
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.11 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# pw-class VLAN
Router(config-l2vpn-xc-p2p-pw)# commit
```

2. Review the VLAN mode running configuration.

   Example

```
l2vpn
 pw-class VLAN
  encapsulation mpls
   transport-mode vlan
  !
 !
 xconnect group grp1
  p2p xc1
   neighbor 10.0.0.11 pw-id 222
    pw-class VLAN
   !
  !
 !
!
```

3. Use the **show l2vpn xconnect group grp1 detail | i " PW type"** command to verify the VLAN mode configuration.

   Example

```
Router# show l2vpn xconnect group grp1 detail | i " PW type"
PW type Ethernet VLAN, control word disabled, interworking none
      PW type      Ethernet VLAN                  Ethernet VLAN
```

   The PW type Ethernet VLAN value indicates a virtual connection type 4 pseudowire.

The VLAN mode pseudowire is configured, and the verification output reports PW type Ethernet VLAN.

VLAN passthrough mode

Explains VLAN passthrough mode as the pseudowire transport mode that carries the attachment-circuit VLAN tag manipulation result without adding a dummy VLAN tag.

VLAN passthrough mode is a VLAN transport mode that

- negotiates a virtual connection type 4 Ethernet VLAN pseudowire
- transports what comes out of the attachment circuit after VLAN tag manipulation, and
- leaves at least one VLAN tag on the frame without adding a dummy VLAN 0 tag.

The **transport mode vlan passthrough** command is configured under the pseudowire class to enable VLAN passthrough mode.

Pseudowire redundancy

Explains pseudowire redundancy as a backup-path mechanism for pseudowire over MPLS services and describes how the provider edge router switches traffic after primary pseudowire failure.

A pseudowire redundancy is a backup-path mechanism that

- establishes a secondary pseudowire to support the primary pseudowire
- enables the provider edge router to reroute traffic to the secondary pseudowire upon primary failure, and
- allows automatic restoration of the primary path once it becomes available.

The primary pseudowire fails when the provider edge router fails or when a network outage occurs.

Feature history

The feature history table lists release support for pseudowire redundancy.

Table 28: Feature History Table

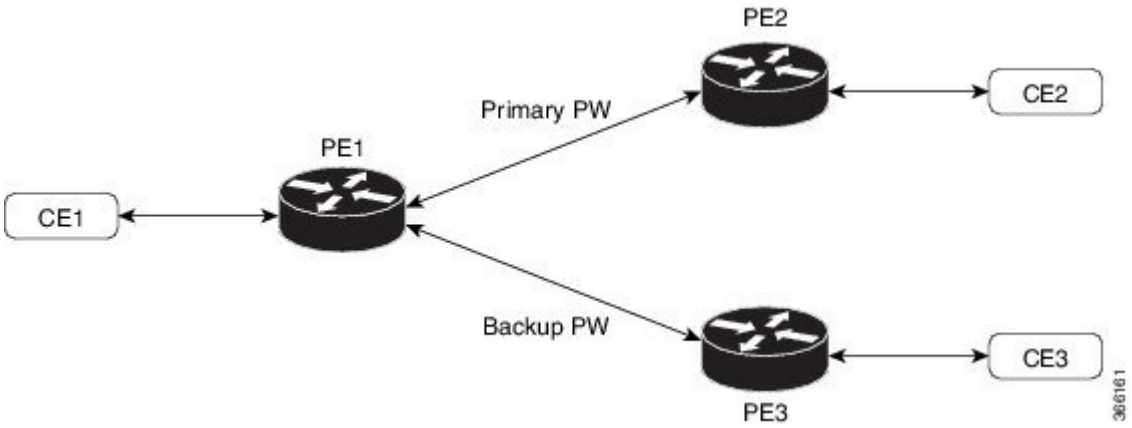
Feature Name	Release Information	Feature Description
Pseudowire Redundancy	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Pseudowire Redundancy	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The PW redundancy functionality is now extended to the Cisco 8712-MOD-M routers.
Pseudowire Redundancy	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) * The PW redundancy functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
Pseudowire Redundancy	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) Pseudowire redundancy enhances network reliability by providing backup paths for pseudowires in case of a failure, ensuring continuous data transmission. This feature allows for the establishment of primary and secondary pseudowires, which can automatically switch traffic to the backup if the primary path fails. * This functionality is now extended to routers with the 88-LC1-36EH line cards.

Pseudowire redundancy

The pseudowire redundancy feature allows you to configure a redundant pseudowire that backs up the primary pseudowire. When the primary pseudowire fails, the PE router switches to the redundant pseudowire. You can elect to have the primary pseudowire resume operation after it becomes functional. The primary pseudowire fails when the PE router fails or when there is a network outage.

Figure 15: Pseudowire redundancy



Forcing a manual switchover to the backup pseudowire

Provides the command-use behavior for forcing a router to switch to the backup pseudowire or return to the primary pseudowire.

Use the **l2vpn switchover** command in EXEC mode to manually force the router to switch to the backup pseudowire or return to the primary pseudowire. The command performs these functions:

- Forces the router to switch to the backup pseudowire.
- Switches back to the primary pseudowire using the same command.
- Initiates manual switchover only if the specified peer is available.
- Moves the cross-connect to a fully active state when executed.

Configure pseudowire redundancy

Configure primary and backup pseudowires on provider edge routers so traffic can use a redundant pseudowire when the primary pseudowire fails.

Set up pseudowire redundancy for a Layer 2 VPN cross-connect to ensure continuous connectivity in case of pseudowire failure.

The configuration defines a primary pseudowire and a backup pseudowire under the point-to-point cross-connect on provider edge (PE) routers.

Before you begin

- 2000 active and 2000 backup PWs are supported.
- Only MPLS Label Distribution Protocol is supported.

Procedure

1. Configure the primary pseudowire and backup pseudowire on PE1.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# backup neighbor 192.168.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw-backup)# commit
```

2. Configure the pseudowire on PE2.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

3. Configure the pseudowire on PE3.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

4. Use the **show l2vpn xconnect group XCON_1** command to verify that the configured pseudowire redundancy is up.

```
Router (PE1)# show l2vpn xconnect group XCON_1
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                         Segment 2
Group      Name       ST   Description            ST       Description
     ST
------------------------   ------------------------------
------------------------------
XCON_1     XCON1_P2P2 UP   Hu0/1/0/0.1            UP       172.16.0.1     1000
   UP
                                                           Backup
                                                           192.168.0.1
1000   SB
------------------------------------------------------------------------------


Router (PE2)# show l2vpn xconnect group XCON_1
Tue Jan 17 15:36:12.327 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                         Segment 2
Group      Name       ST   Description            ST       Description
     ST
------------------------   ------------------------------
------------------------------
XCON_1     XCON1_P2P2 UP   BE100.1                UP       10.0.0.1     1000
  UP
------------------------------------------------------------------------------


Router (PE3)# show l2vpn xconnect group XCON_1
Tue Jan 17 15:38:04.785 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                         Segment 2
Group      Name       ST   Description            ST       Description
     ST
------------------------   ------------------------------
------------------------------
XCON_1     XCON1_P2P2 DN   BE100.1                UP       10.0.0.1     1000
  SB
------------------------------------------------------------------------------

Router# show l2vpn xconnect summary
Number of groups: 3950
Number of xconnects: 3950
  Up: 3950  Down: 0  Unresolved: 0 Partially-programmed: 0
  AC-PW: 3950  AC-AC: 0  PW-PW: 0 Monitor-Session-PW: 0
Number of Admin Down segments: 0
Number of MP2MP xconnects: 0
  Up 0 Down 0
  Advertised: 0 Non-Advertised: 0
Number of CE Connections: 0
  Advertised: 0 Non-Advertised: 0
Backup PW:
```

```
  Configured   : 3950
  UP           : 0
  Down         : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby      : 3950
  Standby Ready: 0
Backup Interface:
  Configured   : 0
  UP           : 0
  Down         : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby      : 0
```

Inter-AS modes

Explains Inter-AS mode for L2VPN pseudowire services and how it supports VPN connectivity across multiple autonomous systems or service-provider domains.

Inter-AS mode is a peer-to-peer mode for L2VPN pseudowire that

- allows VPNs to operate through multiple providers or multidomain networks using Layer 2 VPN cross-connect
- enables VPLS autodiscovery to function across multiple Border Gateway Protocol autonomous systems, and
- provides end-to-end VPN connectivity across different geographical locations.

An autonomous system is a single network or group of networks controlled by a common system administration group and using a single, clearly defined routing protocol.

Feature history

The feature history table lists release support for Inter-AS mode for L2VPN pseudowire.

Table 29: Feature History Table

Feature Name	Release Information	Feature Description
Inter-AS Mode for L2VPN Pseudowire	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Inter-AS Mode for L2VPN Pseudowire	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Feature Name	Release Information	Feature Description
Inter-AS Mode for L2VPN Pseudowire	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Inter-AS Mode for L2VPN Pseudowire	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The Inter-AS mode functionality is now extended to the Cisco 8712-MOD-M routers.
Inter-AS Mode for L2VPN Pseudowire	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) * The Inter-AS mode functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Inter-AS Mode for L2VPN Pseudowire	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) * The Inter-AS mode functionality is now extended to routers with the 88-LC1-36EH line cards.
Inter-AS Mode for L2VPN Pseudowire	Release 7.3.15	Inter-AS is a peer-to-peer type that allows VPNs to operate through multiple providers or multi-domain networks using L2VPN cross-connect. This mode allows VPLS autodiscovery to operate across multiple BGP autonomous systems and enables service providers to offer end-to-end VPN connectivity over different geographical locations.

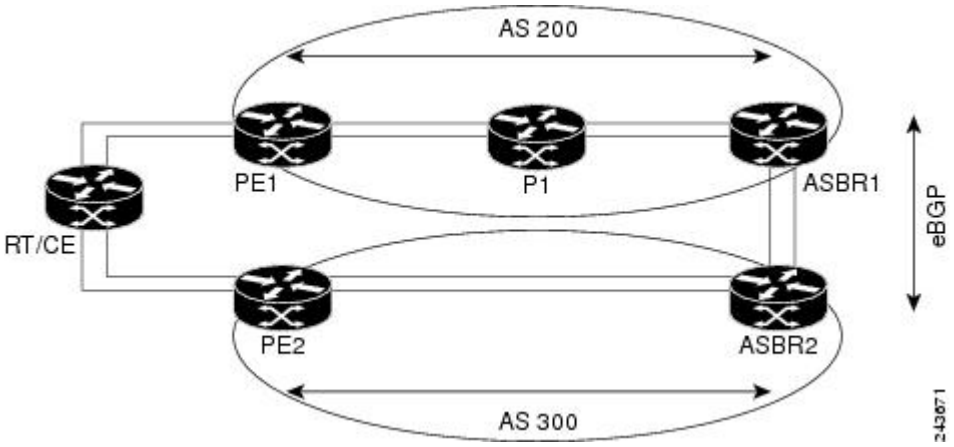
Inter-AS deployment context

As VPNs grow, their requirements expand. In some cases, VPNs must reside on different autonomous systems in different geographic areas. Some VPNs also need to extend across multiple service providers. Regardless of complexity and location, the connection between autonomous systems must be seamless.

Ethernet over MPLS supports both single and multiple autonomous system topologies:

- In the single autonomous system topology, the pseudowire resides within the same autonomous system and connects provider edge routers at both ends of the point-to-point Ethernet over MPLS cross-connects.
- In multiple autonomous system topologies, provider edge routers can reside in two different autonomous systems using internal and external Border Gateway Protocol peering

Figure 16: Ethernet over MPLS over Inter-AS basic double autonomous system topology



Configure Inter-AS mode

Configure Inter-AS mode for Layer 2 VPN pseudowire services by enabling MPLS label distribution, BGP autodiscovery, and Layer 2 VPN signaling.

Configure Inter-AS mode across provider edge routers to enable Layer 2 VPN pseudowire deployment using MPLS Label Distribution Protocol, BGP autodiscovery, and signaling.

This procedure enables key protocols and configuration parameters required for Layer 2 VPN service across autonomous systems, including route target planning and provider edge (PE) router setup.

Before you begin

Confirm that the autonomous system design and provider edge router addresses are planned before you configure Inter-AS mode.

Procedure

1. Configure Inter-AS mode on PE1.

Example

```
Router# configure
Router(config)# mpls ldp
Router(config-ldp)# router-id 10.0.0.1
Router(config-ldp)# interface HundredGigE0/2/0/3
Router(config-ldp-if)# exit
Router(config-ldp)# router bgp 100
Router(config-bgp)# bgp router-id 10.0.0.1
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn vpls-vpws
```

```
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group gr1
Router(config-l2vpn-xc)# mp2mp mp1
Router(config-l2vpn-xc-mp2mp)# vpn-id 100
Router(config-l2vpn-xc-mp2mp)# l2-encapsulation vlan
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# rd auto
Router(config-l2vpn-xc-mp2mp-ad)# route-target 192.0.2.2:100
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/1.1
remote-ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/1.1
remote-ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

2. Configure Inter-AS mode on PE2.

   Example

```
Router# configure
Router(config)# mpls ldp
Router(config-ldp)# router-id 172.16.0.1
Router(config-ldp)# interface HundredGigE0/3/0/0
Router(config-ldp-if)# exit
Router(config-ldp)# router bgp 100
Router(config-bgp)# bgp router-id 172.16.0.1
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)# neighbor 10.0.0.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn vpls-vpws
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group gr1
Router(config-l2vpn-xc)# mp2mp mp1
Router(config-l2vpn-xc-mp2mp)# vpn-id 100
Router(config-l2vpn-xc-mp2mp)# l2-encapsulation vlan
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# rd auto
Router(config-l2vpn-xc-mp2mp-ad)# route-target 192.0.2.2:100
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/2.1
remote-ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/2.2
remote-ce-id 1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

3. Review the running configuration.

Inter-AS mode is configured on both provider edge routers, enabling MPLS Label Distribution Protocol, Border Gateway Protocol autodiscovery, and Layer 2 VPN signaling for pseudowire services.

```
PE1 configuration
mpls ldp
 router-id 10.0.0.1
 interface HundredGigE0/2/0/3
 !
router bgp 100
 bgp router-id 10.0.0.1
 address-family l2vpn vpls-vpws
 neighbor 172.16.0.1
  remote-as 200
  update-source Loopback0
  address-family l2vpn vpls-vpws
!
l2vpn
 xconnect group gr1
  mp2mp mp1
    vpn-id 100
    l2-encapsulation vlan
    autodiscovery bgp
     rd auto
     route-target 192.0.2.2:100
     signaling-protocol bgp
      ce-id 1
       interface HunGigE0/1/0/1.1 remote-ce-id 2
       interface HunGigE0/1/0/1.2 remote-ce-id 3

PE2 configuration
mpls ldp
 router-id 172.16.0.1
 interface HundredGigE0/3/0/0
 !
router bgp 100
 bgp router-id 172.16.0.1
 address-family l2vpn vpls-vpws
 neighbor 10.0.0.1
  remote-as 100
  update-source Loopback0
  address-family l2vpn vpls-vpws
!
l2vpn
 xconnect group gr1
  mp2mp mp1
    vpn-id 100
    l2-encapsulation vlan
    autodiscovery bgp
     rd auto
     route-target 192.0.2.2:100
     signaling-protocol bgp
      ce-id 2
       interface HunGigE0/1/0/2.1 remote-ce-id 3
       interface HunGigE0/1/0/2.2 remote-ce-id 1
```

Local switching between attachment circuits

Details local switching between attachment circuits by introducing the concept and providing configuration procedures for enabling seamless traffic forwarding within a single device.

A local switching connection between attachment circuits is a point-to-point Layer 2 service that

- exchanges Layer 2 data from one attachment circuit to another on the same device
- operates like a bridge domain that has only two bridge ports, and
- forwards traffic from one port of the local connection to the other port.

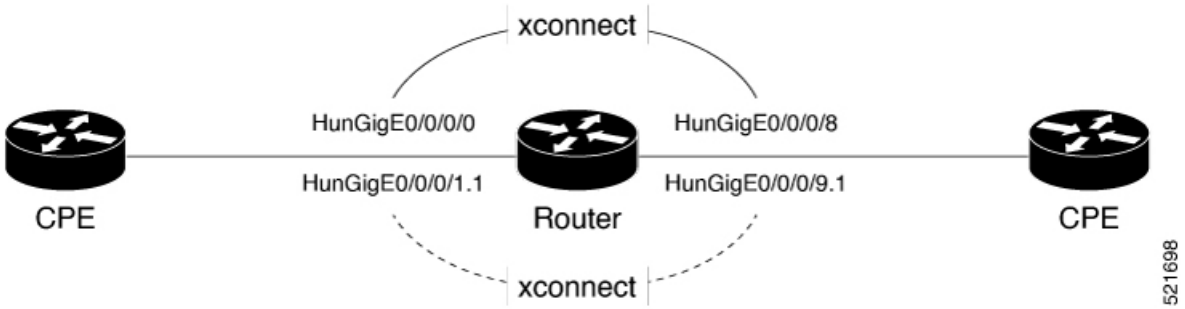
Feature history
Table 30: Feature History Table

Feature Name	Release Information	Feature Description
Support of Tagged or Untagged VLAN on Physical and Bundle AC with VLAN Rewrite	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Support of Tagged or Untagged VLAN on Physical and Bundle AC with VLAN Rewrite	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Support of Tagged or Untagged VLAN on Physical and Bundle AC with VLAN Rewrite	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8712-MOD-M
Support of Tagged or Untagged VLAN on Physical and Bundle AC with VLAN Rewrite	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Support of Tagged or Untagged VLAN on Physical and Bundle AC with VLAN Rewrite	Release 7.3.15	This feature supports tagged or untagged VLAN on physical and bundle interfaces. The tagged VLAN allows you to send and receive the traffic for multiple VLANs whereas the untagged VLAN allows you to send and receive the traffic for a single VLAN. The multiple VLANs are used to differentiate traffic streams so that the traffic can be split across different services.

Layer 2 local switching characteristics

Figure 17: Local switching between attachment circuits



Layer 2 local switching connections have these characteristics:

- Because there is no bridging in a local connection, there is neither MAC learning nor flooding.
- Attachment circuits in a local connection are not in the UP state if the interface state is DOWN.
- Local switching attachment circuits use Layer 2 trunk main interfaces, bundle interfaces, and Ethernet Flow Points.
- Same-port local switching allows Layer 2 data switching between two circuits on the same interface.

Configure local switching between attachment circuits

Configure an attachment-circuit to attachment-circuit point-to-point cross-connect and verify that the local switching cross-connect is up after the commit succeeds.

Use this task to configure point-to-point cross-connect between attachment circuits.

This task creates Layer 2 interfaces, establishes a point-to-point connection, and attaches the Layer 2 interfaces to the connection for local switching.

- Create Layer 2 interfaces.
- Create a cross-connect group and point-to-point connection.
- Attach the Layer 2 interfaces to the point-to-point connection.

Procedure

1. Configure Layer 2 transport and encapsulation on the VLAN subinterfaces.

Example

```
Router# configure
Router(config)# interface HunGigE 0/0/0/1.1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# exit
Router(config)# interface HunGigE 0/0/0/9.1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# commit
```

2. Configure local switching on the VLAN subinterfaces.

Example

```
Router(config)# l2vpn
Router(config-l2vpn-xc)# p2p XCON1_P2P1
Router(config-l2vpn-xc-p2p)# interface HunGigE0/0/0/1.1
Router(config-l2vpn-xc-p2p)# interface HunGigE0/0/0/9.1
Router(config-l2vpn-xc-p2p)# commit
Router(config-l2vpn-xc-p2p)# exit
```

3. Review the local switching running configuration.

Example

```
configure

interface HunGigE 0/0/0/1.1 l2transport
 encapsulation dot1q 5
!
interface HunGigE 0/0/0/9.1 l2transport
  encapsulation dot1q 5
 !

l2vpn
   p2p XCON1_P2P1
    interface HunGigE0/0/0/1.1
    interface HunGigE0/0/0/9.1
    !
   !
 !
```

4. Use the **show l2vpn xconnect brief** command to verify that the configured cross-connect is up.

Example

```
router# show l2vpn xconnect brief

Locally Switching

  Like-to-Like                          UP        DOWN         UNR

    EFP                                  1           0           0

    Total                                1           0           0


  Total                                  1           0           0
```

```
Total: 1 UP, 0 DOWN, 0 UNRESOLVED
```

The local switching cross-connect is configured successfully, and verification output shows one locally switched EFP in the UP state.

9 Pseudowire Headend and Advanced Pseudowire Features

Topics:

- Virtual circuit connection verification features
- Pseudowire headend
- GIL prune behavior for PWHE interfaces
- Multisegment pseudowires

Explains advanced pseudowire concepts, covering headend interface architecture, virtual circuit verification, multisegment pseudowires, GIL prune behavior, hardware requirements, configuration tasks, and operational behaviors for robust Layer 2 VPN connectivity.

Virtual circuit connection verification features

Introduces virtual circuit connection verification capabilities, detailing mechanisms and operational packet handling required to monitor, validate, and troubleshoot L2VPN connectivity using robust verification processes.

Virtual circuit connection verification is a Layer 2 VPN operations, administration, and maintenance feature that

- runs an IP-based provider edge-to-provider edge keepalive protocol across a specified pseudowire
- verifies that pseudowire data-path forwarding does not contain faults, and
- uses a control channel that is negotiated when the pseudowire is established.

Feature history

The feature history table lists release support for this feature.

Table 31: Feature History Table

Feature Name	Release Information	Feature Description
Virtual Circuit Connection Verification on L2VPN	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Virtual Circuit Connection Verification on L2VPN	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The VCCV functionality is now extended to the Cisco 8712-MOD-M routers.
Virtual Circuit Connection Verification on L2VPN	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only* *The VCCV functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Virtual Circuit Connection Verification on L2VPN	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) Virtual Circuit Connection Verification (VCCV) enhances network reliability by enabling operators to detect and troubleshoot faults in the pseudowire data path, ensuring uninterrupted data transmission. It utilizes an IP-based keepalive protocol between provider edges (PEs) across a specified pseudowire, with VCCV packets managed on a dedicated control channel. *This functionality is now extended to routers with the 88-LC1-36EH line cards.

Virtual circuit connection verification packet handling

Provides details on the packet types, control channel operation, and counter handling for Virtual Circuit Connection Verification (VCCV).

Virtual Circuit Connection Verification (VCCV) is an L2VPN Operations, Administration, and Maintenance (OAM) feature that enables network operators to run IP-based provider edge-to-provider edge (PE-to-PE) keepalive protocols across a specified pseudowire, ensuring the pseudowire data path is free from faults. The disposition PE receives VCCV packets on a control channel associated with the pseudowire. Control channel type and connectivity verification type are negotiated during pseudowire establishment between PEs in 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 8000 series routers support 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, which is the reply mode in IPv4. The reply may be forwarded as IP, MPLS, or a combination of both.

VCCV ping counters are:

- Counted in MPLS forwarding on the egress side
- On the ingress side, sourced by the route processor and not counted as MPLS forwarding counters

Pseudowire headend

Details pseudowire headend interface architecture, highlighting hardware requirements, interface behaviors, benefits, supported topologies, traffic flow types, encapsulation and decapsulation processes, configuration guidelines, restrictions, and procedures for setup.

A Pseudowire headend interface is a virtual interface that

- terminates access pseudowires into a Layer 3 VRF, a global Layer 3 domain, or a Layer 2 domain
- integrates legacy Layer 2 services into IP or MPLS packet-switched networks, and
- emulates attachment-circuit behavior on the service provider edge.

Feature history

The feature history table lists release support for this feature.

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire Headend	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100]) (select variants only*) *This feature is supported on Cisco 88-LC1-16H16F-EM line cards.

Feature Name	Release Information	Feature Description
Pseudowire Headend	Release 26.2.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) This feature requires egress feature capability on supported platforms. Use the **hw-module profile edge-mode** CLI command to enable the feature. The feature introduces these changes: CLI: • **hw-module profile edge-mode** • **show hw-module profile edge-mode** *This feature is now supported on: • 8711-48Z-M • 8712-MOD-M
Pseudowire Headend	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) Pseudowire Headend (PWHE) is a virtual interface that allows termination of access PWs into a Layer 3 (VRF or global) domain or into a Layer 2 domain. PWHE enables integration of legacy Layer 2 services into packet-switched networks (PSNs) like IP or MPLS networks, so that users can integrate their older devices into newer networks without upgrading their hardware. This is possible because PWHE allows the termination or encapsulation of the frames from the attachment circuit into packets that can be transmitted over the PSN. *This feature is supported only on: • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Requirement: Pseudowire headend hardware support

Outlines the hardware support conditions for pseudowire headend interfaces.

To ensure supported feature behavior for pseudowire headend interfaces., follow these requirements:

- Only line cards and routers with the P100-based Silicon One ASIC support pseudowire headend functionality.
- Review the requirement when planning or configuring pseudowire architectures.
- The requirement preserves source restrictions, limits, and deployment conditions for the feature.
- Adjust the design or configuration before deployment if any condition is not met.

Pseudowire headend interface behavior

Provides details on the behavior, attributes, and provisioning features for pseudowire headend interfaces in IP-MPLS networks.

Pseudowire headend interfaces (PWHE) in IP-MPLS packet-switched networks (PSNs) enable transparent transport of payloads and facilitate integration of legacy Layer 2 services. The following summarizes their features and functions:

- Lightweight tunnel integration: Pseudowires act as simple, manageable tunnels that return customer traffic to core networks efficiently.
- Flexible termination: PWHE interfaces terminate access pseudowires into Layer 3 (VRF and global) domains or Layer 2 domains, supporting integration of older devices into newer networks without hardware upgrades.
- Frame conversion: PWHE allows termination of frames from the attachment circuit and conversion to packets suitable for PSN transport.
- Feature provisioning: PWHE supports configuration of features such as Quality of Service (QoS), access control lists (ACL), and Layer 3 VPNs on a per-interface basis, enhancing service provider flexibility.
- Attachment-circuit emulation: When PWHE is configured on a router, its interface emulates the behavior of an attachment circuit (AC) and terminates pseudowires—enabling Layer 2 VPN deployments even when a physical AC is not present.

Benefits of pseudowire headend

Lists the key benefits and features of a pseudowire headend.

A pseudowire headend helps network providers improve flexibility, reduce costs, and scale services. The primary advantages include:

- Dissociates the customer facing interface (CFI) of the service PE from the underlying physical transport media of the access or aggregation network.
- Reduces capital expenditure (CapEx) in the access or aggregation network and service provider edge (PE) devices.
- Distributes and scales the customer facing Layer 2 user-network interfaces (UNI) set.
- Implements a uniform method of OAM functionality.
- Allows providers to extend or expand Layer 3 service footprints.
- Provides a method for terminating customer traffic into a next generation network (NGN).

How pseudowire headend works

Describes how Pseudowire Headend changes the access-to-provider-edge topology and how Generic Interface Lists support PWHE deployment.

n a PWHE topology, the access network consists of multiple customer edge (CE) devices connected to the access PE (A-PE) via single links. The S-PE is connected to various provider routers using multiple links and line cards, enhancing redundancy and scalability. PWHE integrates L2-PE and S-PE functions and uses BGP with MPLS label distribution (RFC 3107) instead of IGP, simplifying the network and bypassing certain connectivity restrictions.

Summary

The key components involved in the process are:

- Layer 2 provider edge (L2-PE): Originates pseudowires (PWs) from the access network and connects to the service provider edge.

- Service provider edge (S-PE): Terminates PWs and emulates the behavior of the access circuit (AC) for PWHE, combining functions previously split between L2-PE and S-PE.
- Provider routers (P1 and P2): Serve as intermediate nodes between the access provider edge (A-PE) and S-PE, offering connectivity via multiple links and line cards.
- Generic Interface List (GIL): Groups multiple PWs for streamlined configuration and management, applying changes across all associated PW interfaces.

PWHE changes the traditional access-to-provider-edge network topology by terminating access pseudowires directly on the service provider edge, simplifying Layer 2 VPN deployments. GIL enable efficient management of multiple pseudowire interfaces associated with PWHE.

Workflow

Figure 18: Pseudowire network without PWHE

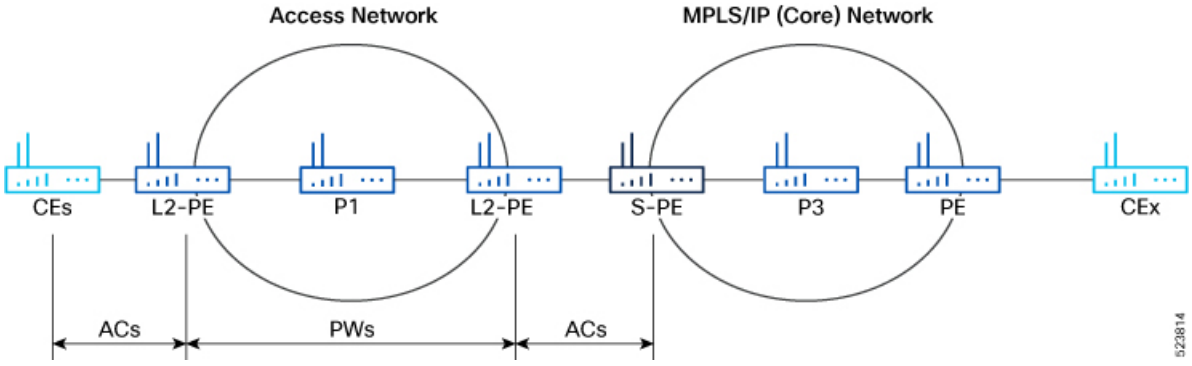


Figure 19: Pseudowire network with PWHE

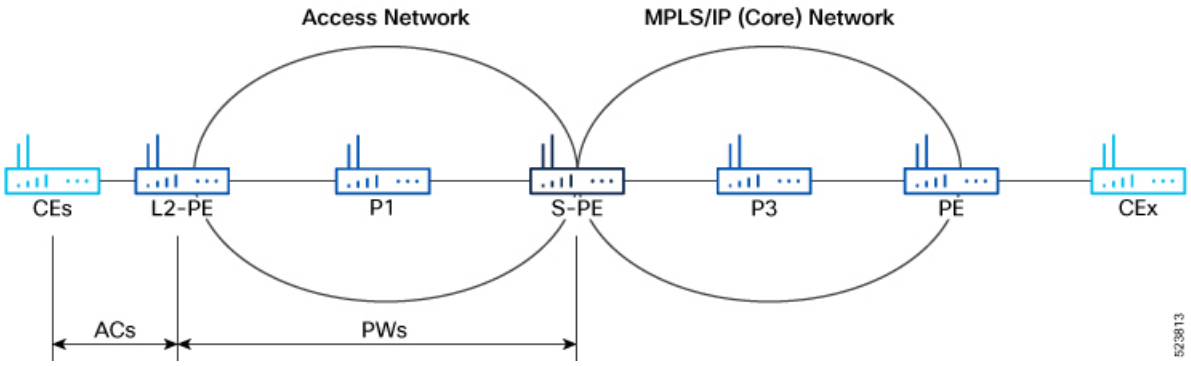
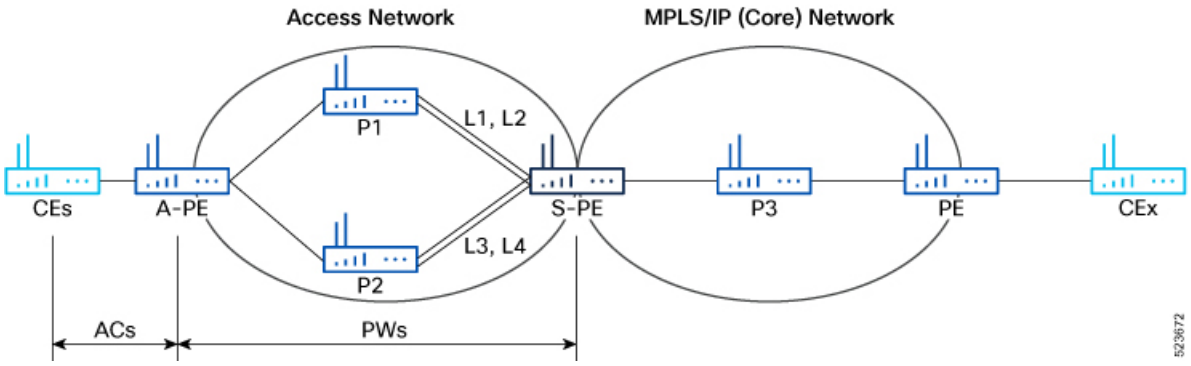


Figure 20: PWHE deployment



1. Access network setup: Multiple provider routers (P1 and P2) are deployed between A-PE and S-PE, each connected via separate links and line cards.

2. Pseudowire connection establishment: S-PE connects to P1 using links L1 and L2, and to P2 using links L3 and L4; these links attach to distinct line cards for resilience.
3. Cross-connect configuration: For each CE-A-PE link, a cross-connect (AC-PW) is configured on A-PE, enabling pseudowire traffic between CE and service provider.
4. PWHE functionality integration: S-PE combines L2-PE and AC functions, terminating pseudowires and using ARP resolution for customer IP addresses, with packets operating in bridged interworking mode (VC type 5).
5. GIL deployment and management: Multiple PWs are grouped under a generic interface list. Configuration changes to a GIL automatically apply to all associated PW interfaces, streamlining PWHE deployments.

Traffic flow types on PWHE interfaces

Lists the traffic flow types that are involved in PWHE interfaces.

There are two types of traffic flow involved in the PWHE interfaces.

- PWHE decapsulation/disposition flow: This flow occurs when traffic travels from the access side to the core. PWHE ingress features are executed on the PWHE access-facing line card.
- PWHE encapsulation/imposition flow: This flow occurs when traffic travels from the core to the access side. PWHE egress features are executed on the PWHE access-facing line card.

How PWHE decapsulation works

Describes pwhe decapsulation traffic flow and packet-handling behavior.

The packets from access side to core side are decapsulated and the ingress features are executed.

Summary

The key components involved in the process are:

- Access device: Sends packets into the PWHE router using encapsulated formats.
- PWHE router: Performs decapsulation by removing headers/labels and reconstructing the packet.
- Core network: Receives the fully reconstructed packets and forwards them appropriately.

PWHE decapsulation handles the forwarding of packets from the access side to the core network by removing multiple encapsulation layers and reconstructing packets for delivery.

Workflow

These stages describe how PWHE decapsulation work.

1. The PWHE router receives a packet from the access side.
2. The PWHE router removes the outer L2 header or VLAN tag.
3. The PW label is removed from the packet.
4. The inner L2 header is removed, revealing the payload for later forwarding.
5. The PWHE router reconstructs the packet with the appropriate transport label, service label, and L2 header needed by the core network.
6. The reconstructed packet is delivered into the core network.

Result

The process completes when the pseudowire decapsulation and forwarding behavior matches the intended network path, ensuring seamless packet delivery from the access side to the core network.

How PWHE encapsulation works

Describes how PWHE encapsulation processes packets, including traffic flow and packet-handling behaviors between the core and access sides.

The packets from core side to access side are encapsulated and the egress features are executed.

Summary

The key components involved in the process are:

- Core network device: Receives packets and prepares them for encapsulation and forwarding.
- PWHE encapsulation engine: Adds necessary pseudowire and transport labels, reconstructs the packet, and prepares it for egress.
- Access network device: Receives the reconstructed packet and handles its final delivery into the access network.

PWHE encapsulation handles the forwarding of packets from the core side to the access side in a network by encapsulating Ethernet frames, reconstructing them with relevant labels, and delivering them to their destination.

Workflow

These stages describe how PWHE encapsulation work.

1. Encapsulation of L2 header: The core network device receives a packet and the PWHE encapsulation engine encapsulates the inner Layer 2 header.
2. Packet reconstruction: The encapsulation engine reconstructs the packet by adding a PW label, one or more transport labels, and an outer Layer 2 header.
3. Packet delivery to access network: The reconstructed packet is forwarded to the access network device, which handles the final delivery.

Result

PWHE encapsulation results in seamless packet forwarding from the core to the access network, ensuring that traffic is properly labeled, encapsulated, and handled according to the specified forwarding path.

Generic interface lists

Explains how a Generic Interface List groups interfaces for PWHE forwarding and configuration inheritance.

A Generic interface lists is a PWHE interface grouping construct that

- groups multiple outgoing interfaces under one logical list
- applies updates to all associated pseudowire interfaces, and
- helps scale PWHE deployments that use multiple paths to an access provider edge.

Additional reference information

A generic interface list (GIL) is a list of physical or bundle interfaces used in a PWHE connection. The GIL supports only main interfaces, not subinterfaces. It is bi-directional and restricts both receive and transmit interfaces on access-facing line cards, with no impact on the core-facing side.

A GIL 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 GIL and expects PWHE packets to arrive only on line cards that contain GIL members. If packets arrive at a line card without GIL members, they are dropped.

Restrictions for pseudowire headend

Outlines restrictions that apply to PWHE deployments.

- These features are not supported on PWHE:
  - ISIS as IGP for access core
  - PW classVC label 4
  - Segment Routing Traffic Engineering (SR-TE) in the access core for Layer 2 VPN
  - TE tunnel as preferred path in access core for Layer 2 VPN
  - Traffic with Internet Mix (IMIX)
  - The commands **load-balancing flow src-dst-ip** and **flow-label both**
  - PWHE load balancing by VC label or by Flow-Aware Transport (FAT)
  - EVPN multihoming mode
  - PWHE MTU
- The load balancing hashing is performed only by PWHE link numbers.
- When a packet arrives at PWHE Layer 3 subinterface, the software aggregate count is done on PWHE main interface.
- Do not use mixed-mode Egress Traffic Management (ETM): avoid combining ETM and non-ETM members within a GIL.
- Do not use the ECMP path list as a superset of GIL interfaces.
- If you configure features like QoS and ACL on a GIL, they apply as follows:
  - For PWHE traffic received on GIL, all the features configured on the PWHE interface are applicable.
  - For other traffic received on GIL, all the features configured on the GIL interface are applicable.
- You can attach ACL in PWHE interface for both ingress and egress, for IPv4 and IPv6.
- You can attach hybrid-ACL (Level 2) in PWHE interface for only ingress, for IPv4 and IPv6.

Pseudowire headend configuration guidelines

States the requirements you must meet before configuring pseudowire headend (PWHE).

You must meet all these requirements before configuring a PWHE:

- The generic interface list members must be the superset of the ECMP path list to the Access Provider Edge (A-PE).
- Only eight generic interface lists are supported per A-PE neighbor address.
- Eight Layer 3 links per generic interface list are supported.
- Only PW-Ether interfaces can be configured as PWHE L2 or L3 subinterfaces.
- Cross-connects that contain PW-Ether main interfaces can be configured as VC-type 5.
- PW-Ether interfaces and subinterfaces can be configured with both IPv4 and IPv6. The packet-switched network must be capable of routing IPv4 and IPv6 packets to encapsulate and transport these frames over a pseudowire.
- Pseudowire redundancy, preferred path, local switching or L2TP are not supported for cross-connects configured with PWHE.

- The TE and LDP applications work on physical interfaces and therefore do not allow PWHE configuration.
- Address family, CDP, and MPLS configurations are not allowed on PWHE interfaces.
- For PWHE, eBGP, static routes, OSPF, and ISIS are supported with both IPv4 and IPv6. Routing Information Protocol (RIP) is supported only with IPv4; IPv6 is not supported.
- You must attach a different generic interface list for PW-Ether interfaces with different remote neighbors. This means creating a separate and dedicated generic interface list for each remote neighbor or peer whose remote neighbors are different routers or devices. The generic interface list may have the same set of outgoing interfaces.

Configure pseudowire headend

Configure PWHE on A-PE and S-PE routers, including PWHE cross-connects, L2 subinterfaces, and L3 subinterfaces.

Set up pseudowire headend (PWHE) interfaces and cross-connects, including Layer 2 and Layer 3 subinterfaces, on Access Provider Edge (A-PE) and Service Provider Edge (S-PE) routers.

This task uses configuration examples for provider edge roles (A-PE and S-PE) and PWHE interfaces.

Procedure

1. Configure L2 interface.

   Example

```
Router(config)# interface hundredGigE 0/1/0/3
Router(config-if)# l2transport
Router(config-if-l2)# root
```

2. Configure PWHE cross-connect.

   Example

```
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface hundredGigE 0/1/0/3
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 10.1.1.1 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

3. Create a generic interface list (GIL).

   Example

```
Router(config)# generic-interface-list txlist
Router(config-gen-if-list)# interface hundredGigE 0/1/0/1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/2
Router(config-gen-if-list)# commit
```

4. Configure pw-ether interface and attach GIL

Example

```
Router(config)# interface pw-ether1
Router(config-if)# ipv4 address 10.1.1.1/24
Router(config-if)# ipv6 address 2001:DB8::2/64
Router(config-if)# attach generic-interface-list txlist
Router(config-if)# commit
```

5. Configure cross-connect to include pw-ether interface.

Example

```
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface pw-ether1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.2.2.2 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

A-PE configuration guidance:

- Configure the A-PE with PWHE cross-connect to include the L2 interface and PW towards the S-PE. Cross-connect (xconnect) establishes the connection between access provider and service provider edges.

S-PE configuration guidance:

- On the S-PE, create the generic interface list (GIL), configure the pw-ether interface and attach GIL, and configure cross-connect to include the pw-ether interface (use the **transport-mode ethernet** command for Ethernet traffic).

6. Configure L2 subinterface.

A-PE configuration for L2 subinterface:

Example

```
Router(config)# interface hundredGigE 0/1/0/3.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# rewrite ingress tag pop 1 symmetric
```

S-PE configuration for L2 subinterface:

Example

```
Configure L2 subinterface
Router(config)# interface PW-Ether1.2 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric
```

7. Configure cross-connect and assign the L2 subinterface.

Example

```
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode vlan
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface pw-ether1.2
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.2.2.2 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

There is no need to configure cross-connect for the subinterface, as the main PWHE interface configuration is applied to the subinterface.

8. Configure Layer 3 (L3) subinterface.

Example

```
Router(config)# interface pw-ether 1.1
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# ipv4 address 10.1.1.1/24
Router(config-subif)# commit
```

Except for the A-PE, repeat the other configurations as described for the L2 interface.

There is no need to configure cross-connect for the subinterface, as the main PWHE interface configuration is applied to the subinterface.

9. Use the **show generic-interface-list idb name txlist** command to verify the details of GIL.

Example

```
Router# show generic-interface-list idb name txlist

GIL name: txlist, ifhandle: 0xf000014
  State: Down Immediate
  Members:
    HundredGigE0/1/0/2
    HundredGigE0/1/0/1
```

Use the **show l2vpn xconnect group pw-he xc-name pw-ether1** command to verify that the status of xconnect group with PWHE Up.

Example

```
Router# show l2vpn xconnect group pw-he xc-name pw-ether1
Mon Mar 11 21:26:48.281 EDT
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed,
LU = Local Up, RU = Remote Up, CO = Connected, (SI) = Seamless Inactive

XConnect                   Segment 1       Segment 2
Group     Name         ST  Description ST Description              ST
------------------------ ------------------------------
------------------------------
```

```
pw-he pw-ether1       UP PE1          UP EVPN 1,1,192.0.2.10 UP
-------------------------------------------------------------------------------
```

Use the **show l2vpn pwhe interface pw-ether 1 detail** command to verify that the status of PWHE interface.

Example

```
Router# show l2vpn pwhe interface pw-ether 1 detail

Interface: PW-Ether1 Interface State: Up, Admin state: Up
Interface handle 0xf000054
MTU: 1514
BW: 10000 Kbit
Interface MAC addresses: 1859.f57d.0008
Label: 24041
Internal ID: None
L2-overhead: 0
VC-type: 5
CW: Y
Hash: 0xf5f5 [Success]
```

The pseudowire headend configuration is complete when the router commits your configurations successfully. PWHE interfaces and cross-connects should be operational and verified via the show commands.

GIL prune behavior for PWHE interfaces

Explains GIL prune behavior for PWHE interfaces, focusing on the interaction between generic interface lists and multicast/broadcast traffic management within advanced pseudowire deployments.

A GIL prune behavior for PWHE interfaces is a network forwarding mechanism that

- aligns the PWHE underlay with a Generic Interface List (GIL)
- sends PWHE traffic only to interfaces where PWHE is present or replicated, and
- limits PWHE replication to the necessary line card locations.

Feature history

The feature history table lists release support for this feature.

Table 33: Feature History Table

Feature Name	Release Information	Feature Description
Enhance network efficiency and scalability with GIL pruning for PWHE interfaces	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8404-SYS-D router.

Feature Name	Release Information	Feature Description
Enhance network efficiency and scalability with GIL pruning for PWHE interfaces	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8011-12G12X4Y-A/D
Enhance network efficiency and scalability with GIL pruning for PWHE interfaces	Release 24.4.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can now manage hardware resources for a Pseudowire Headend (PWHE) interface more efficiently by limiting PWHE replication to the line card locations where the interfaces listed in the Generic Interface List (GIL) are physically present. This optimization ensures that resource usage is confined to only the necessary line cards. The router internally synchronizes the PWHE underlay with the GIL using a mechanism known as GIL pruning. The GIL consists of a subset of core-facing IGP/LDP-enabled interfaces expected to transmit pseudowire traffic for the PWHE interface. This feature is enabled by default and does not require any user configuration. *This feature is supported on: • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

GIL pruning behavior

GIL pruning is a network forwarding mechanism that

- aligns PWHE underlay with a GIL
- ensures that PWHE traffic is sent only to the interfaces where PWHE is present or replicated, and
- enhances hardware resource usage and allows increased scale of PWHE deployment by limiting the replication of PWHE to necessary interfaces.

This feature eliminates the need for manual synchronization between interfaces in GIL and PWHE underlay next hops.

- The router automatically aligns PWHE underlay with the GIL, ensuring that PWHE traffic is only sent to interfaces where PWHE is present or replicated.
- The feature reduces unnecessary hardware resource usage by ensuring PWHE traffic is only forwarded to necessary interfaces.
- The network performs more efficiently and reliably with optimized forwarding paths and reduced resource wastage.
- The feature ensures that forwarding chains are aligned with the updated interface lists, maintaining high performance and reducing potential points of failure.

The feature is enabled by default; no specific configuration is required. You must configure a Generic Interface List (GIL) for PWHE interfaces.

> **Note**
>
> This feature supports only MPLS LDP as the underlay for PWHE interfaces.

Multisegment pseudowires

Outlines multisegment pseudowire architecture, describing component functions, connectivity solutions, operational benefits, core workflows, and configuration procedures to enable scalable end-to-end L2VPN services through segmented pseudowires.

A multisegment pseudowire is a type of pseudowire that

- connects two or more pseudowire segments across multiple provider edge devices
- extends Layer 2 VPN services across multiple network segments, cores, or autonomous systems, and
- uses switching provider edge devices to connect pseudowire segments between endpoint provider edge devices.

Multisegment pseudowire details

The feature history table lists release support for this feature.

Table 34: Feature History Table

Feature Name	Release Information	Feature Description
Multisegment pseudowires	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Modular Systems (8800 [LC ASIC: P100]) Multisegment pseudowires improve network scalability and flexibility by extending Layer 2 services across multiple network segments. End-to-end stitching of pseudowires lets data flow across multiple provider networks as a single virtual connection. This capability simplifies service deployment and broadens network reach.

Main components and functions of multisegment pseudowires

Provides an overview of the main components and their specific roles in multisegment pseudowires, including pseudowire segments, multisegment pseudowires, and pseudowire stitching.

Multisegment pseudowires (MS-PWs) are constructed from several core components, each with specific roles and characteristics:

- Pseudowire (PW):

    - Establishes a tunnel between two PE routers.
    - Carries Layer 2 payload encapsulated as MPLS data.
    - PE routers at each end are called terminating PE routers (T-PEs).
- Multisegment pseudowire (MS-PW):
    - Consists of two or more PW segments joined together.
    - Behaves as a single point-to-point pseudowire.
    - Uses switching PE routers (S-PEs) to connect segments.
- Pseudowire stitching:
    - Configuration technique that connects independent pseudowires.
    - Achieved by cross-connects on S-PE routers.
    - Enables end-to-end service continuity as a single PW.

Overcoming connectivity barriers with multisegment pseudowire

Provides details about how multisegment pseudowire architecture overcomes connectivity barriers in networks divided by regions or autonomous systems.

Multisegment pseudowire addresses connectivity challenges in networks where end-to-end pseudowires span multiple regions, such as distinct IGP areas or BGP autonomous systems. Key facts include:

- Terminating PE (T-PE) devices cannot always directly communicate across regions due to lack of mutual IP visibility.
- Multisegment pseudowire divides the path into multiple segments, each connecting a pair of PE nodes within the same region.
- The architecture introduces two classes of PE nodes:
    - Terminating PE (T-PE): Ends the pseudowire connection.
    - Switching PE (S-PE): Interconnects segments between regions.
- Concatenating pseudowire segments enables end-to-end connectivity, even across diverse network domains.
- This approach overcomes limitations of targeted LDP session establishment between distant PE nodes.

Benefits of multisegment pseudowire

Lists the primary advantages of using multisegment pseudowire technology in large-scale network environments.

Multisegment pseudowires (MS-PWs) enable network operators to achieve greater flexibility and scalability through these benefits:

- Allow splitting large networks into smaller segments—such as core and metro—while still providing seamless end-to-end connectivity.
- Support scalability by enabling up to 254 pseudowire segments in a single MS-PW, accommodating complex network architectures.
- Facilitate interconnection across multiple cores or autonomous systems, including integration between different carrier networks.

How multisegment pseudowires work

Describes the workflow for multisegment pseudowire signaling and forwarding.

MS-PWs are used when a single pseudowire cannot span the entire distance between two customer sites due to limitations like multiple provider domains or different underlying transport technologies. They provide flexibility and scalability in delivering Layer 2 VPN services.

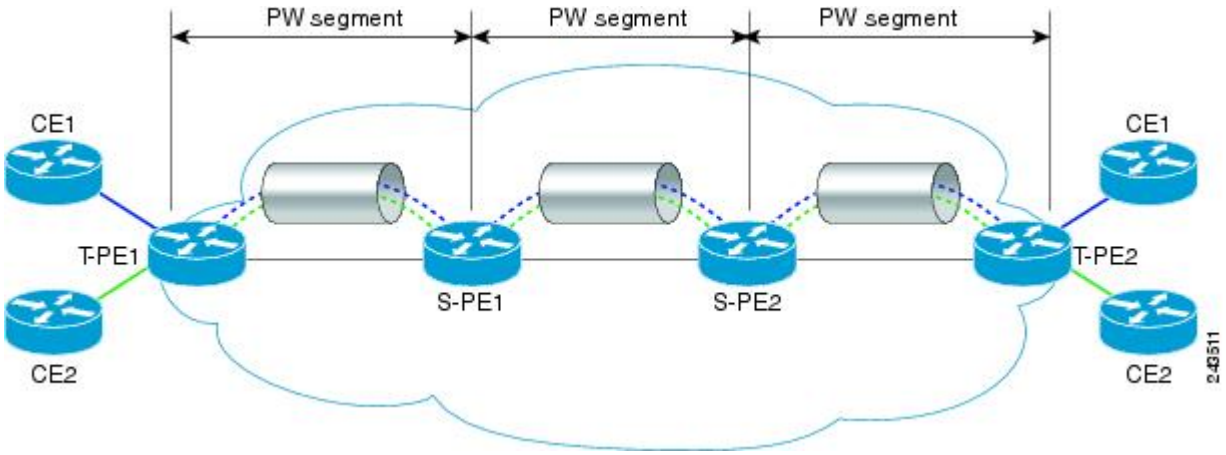
Summary

The key components involved in the process are:

- PE router: Initiates or terminates the pseudowire, providing connectivity between customer edge devices and the provider's network.
- Pseudowire switching (S-PE) point: Acts as an intermediary that connects two or more pseudowire segments, facilitating continuity across multiple segments.
- Pseudowire signaling protocol: Handles setup, maintenance, and teardown of the pseudowire segments across the network.
- CE device: Connects to the PE router and sends/receives Layer 2 traffic transported over the MS-PW.

MS-PWs enable the extension of Layer 2 VPN services across multiple provider domains or segments by connecting two or more pseudowire segments through one or more switching points. This allows service providers to deliver end-to-end Layer 2 connectivity over a network that spans different administrative or technology domains.

Workflow



These stages describe how multisegment pseudowires work.

1. Establishment of pseudowire segments—each PE router establishes a pseudowire segment with the adjacent switching point or PE router using a signaling protocol such as LDP.
2. Pseudowire switching point configuration—the switching point is configured to connect the incoming and outgoing pseudowire segments, forming a continuous path.
3. Signaling and label exchange—PE routers and switching points exchange signaling messages to communicate pseudowire parameters and labels, ensuring correct packet forwarding across segments.
4. Data transmission—once established, user data from the CE device is encapsulated and transmitted over the MS-PW, traversing each segment and switching point until it reaches the remote CE device.
5. Maintenance and teardown—the signaling protocol monitors the status of the MS-PW and manages any required maintenance or teardown operations if connectivity changes or failures occur.

Result

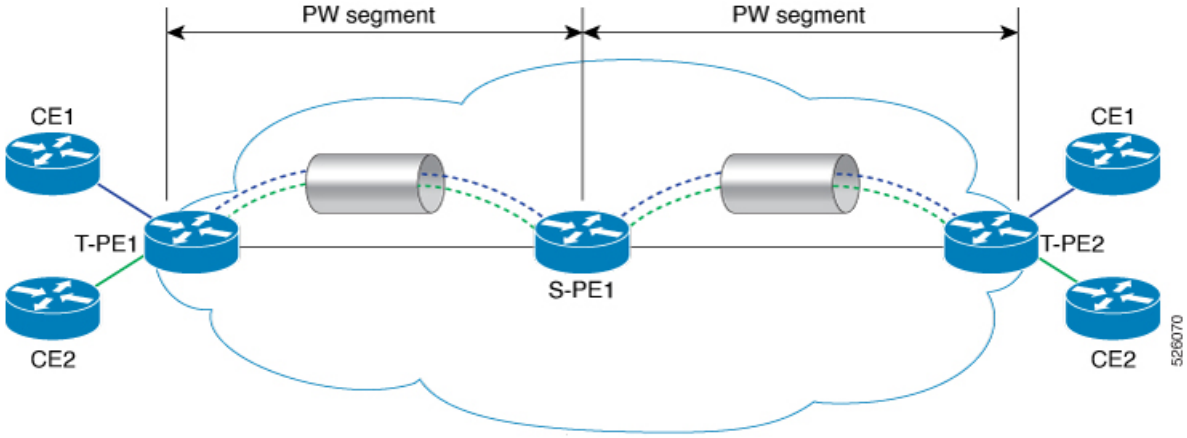
The MS-PW process delivers seamless end-to-end Layer 2 VPN connectivity across multiple network segments and domains, enabling service providers to offer flexible and scalable Ethernet and other Layer 2 services.

Configure multisegment pseudowires

Configure multisegment pseudowires between two terminating provider edges (T-PE1 and T-PE2) with a stitching provider edge (S-PE1), using MPLS encapsulation, pseudowire classes, and xconnect groups. Includes verification guidance and example configurations.

Establish a multisegment pseudowire (MS-PW) service between two terminating PE routers (T-PE1 and T-PE2) and a stitching PE (S-PE1), using pseudowire stitching and MPLS encapsulation.

This task is typically performed to interconnect service provider edge routers across multiple network segments. T-PE routers terminate attachment circuits and use the control word, while S-PE performs pseudowire stitching—linking the segments without attachment circuits. It is implemented using pseudowire classes and xconnect groups.



Before you begin

Confirm that the PWHE interface and related underlay are planned before you begin.

Procedure

1. Configure S-PE1 for pseudowire stitching for MS-PW segments.

Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class T-PE1
Router(config-l2vpn-pw-class)# encapsulation mpls
Router(config-l2vpn-pw-class)# exit
Router(config-l2vpn)# pw-class T-PE2
Router(config-l2vpn-pw-class)# encapsulation mpls
Router(config-l2vpn-pw-class)# exit
Router(config-l2vpn)# xconnect group MS-PW
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.16 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# pw-class T-PE1
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.17 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# pw-class T-PE2
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# exit
```

```
Router(config-l2vpn-xc)# p2p xc2
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.16 pw-id 2
Router(config-l2vpn-xc-p2p-pw)# pw-class T-PE1
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.17 pw-id 2
Router(config-l2vpn-xc-p2p-pw)# pw-class T-PE2
Router(config-l2vpn-xc-p2p-pw)# commit
```

Define pseudowire classes with MPLS encapsulation for T-PE1 and T-PE2, then create an xconnect group with point-to-point (p2p) cross-connects for each segment, specifying both T-PE1 and T-PE2 as neighbors, assigning unique PW IDs, and referencing the correct pseudowire class.

2. Configure T-PE1 to terminate and originate MS-PW segments.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class T-PE1
Router(config-l2vpn-pw-class)# encapsulation mpls
Router(config-l2vpn-pw-class)# control-word
Router(config-l2vpn-pw-class)# exit
Router(config-l2vpn)# xconnect group MS-PW
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether111.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.18 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# pw-class T-PE1
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# p2p xc2
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether111.2
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.18 pw-id 2
Router(config-l2vpn-xc-p2p-pw)# pw-class T-PE1
Router(config-l2vpn-xc-p2p-pw)# commit
```

   Define a pseudowire class with MPLS encapsulation and control word enabled, then create an xconnect group with two p2p cross-connects, each specifying the local interface, neighbor IP, PW ID, and referencing the pseudowire class.

3. Configure T-PE2 to terminate and originate MS-PW segments.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class T-PE2
Router(config-l2vpn-pw-class)# encapsulation mpls
Router(config-l2vpn-pw-class)# control-word
Router(config-l2vpn-pw-class)# exit
Router(config-l2vpn)# xconnect group MS-PW
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether333.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.18 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# pw-class T-PE2
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# p2p xc2
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether333.2
```

```
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.18 pw-id 2
Router(config-l2vpn-xc-p2p-pw)# pw-class T-PE2
Router(config-l2vpn-xc-p2p-pw)# commit
```

Define a pseudowire class with MPLS encapsulation and control word, then configure an xconnect group with two p2p cross-connects, each specifying the local interface, neighbor IP, PW ID, and pseudowire class.

4. Use the show commands to verify the MS-PW configuration.

Example

```
Router# show l2vpn xconnect detail
S-PE1:
Group MS-PW, XC xc1, State: Up, Type: P2P
  PW: neighbor 192.0.2.16, pw-id 1, state: Up (pw-class T-PE1)
  PW: neighbor 192.0.2.17, pw-id 1, state: Up (pw-class T-PE2)
Group MS-PW, XC xc2, State: Up, Type: P2P
  PW: neighbor 192.0.2.16, pw-id 2, state: Up (pw-class T-PE1)
  PW: neighbor 192.0.2.17, pw-id 2, state: Up (pw-class T-PE2)
Note: No local attachment circuits configured (PW-to-PW stitching)
T-PE1:
Group MS-PW, XC xc1, State: Up, Type: P2P
  AC: Bundle-Ether111.1, state: Up
  PW: neighbor 192.0.2.18, pw-id 1, state: Up (pw-class T-PE1)
Group MS-PW, XC xc2, State: Up, Type: P2P
  AC: Bundle-Ether111.2, state: Up
  PW: neighbor 192.0.2.18, pw-id 2, state: Up (pw-class T-PE1)
T-PE2:
Group MS-PW, XC xc1, State: Up, Type: P2P
  AC: Bundle-Ether333.1, state: Up
  PW: neighbor 192.0.2.18, pw-id 1, state: Up (pw-class T-PE2)
Group MS-PW, XC xc2, State: Up, Type: P2P
  AC: Bundle-Ether333.2, state: Up
  PW: neighbor 192.0.2.18, pw-id 2, state: Up (pw-class T-PE2)
```

Example

```
Router# show mpls ldp neighbor brief
Peer LDP Identifier    Transport Address  State       Uptime
192.0.2.16:0           192.0.2.16          OPERATIONAL  00:34:12
192.0.2.17:0           192.0.2.17          OPERATIONAL  00:34:12
192.0.2.18:0           192.0.2.18          OPERATIONAL  00:34:12
```

Example

```
Router# show mpls forwarding
Local  Outgoing    Prefix             Bytes Label   Outgoing   Next Hop
Label  Label       or ID              Switched      Interface
16001  17001       192.0.2.16:1        100000        Te0/0/0/1  192.0.2.16
16002  17002       192.0.2.17:1        100000        Te0/0/0/2  192.0.2.17
17001  Pop Label   192.0.2.18:1        100000        BE111.1    local
17002  Pop Label   192.0.2.18:2        100000        BE111.2    local
```

Example

```
Router# show mpls l2transport vc detail
S-PE1:
```

```
Local interface: none (PW stitching)
VC ID: 1, peer 192.0.2.16, state: UP
  Encapsulation: MPLS, pw-class: T-PE1
VC ID: 1, peer 192.0.2.17, state: UP
  Encapsulation: MPLS, pw-class: T-PE2
VC ID: 2, peer 192.0.2.16, state: UP
  Encapsulation: MPLS, pw-class: T-PE1
VC ID: 2, peer 192.0.2.17, state: UP
  Encapsulation: MPLS, pw-class: T-PE2
T-PE1:
Local interface: Bundle-Ether111.1 up, VC ID: 1
  Peer 192.0.2.18, state: UP, PW class: T-PE1, encapsulation: MPLS,
control-word: enabled

Local interface: Bundle-Ether111.2 up, VC ID: 2
  Peer 192.0.2.18, state: UP, PW class: T-PE1, encapsulation: MPLS,
control-word: enabled
T-PE2:
Local interface: Bundle-Ether333.1 up, VC ID: 1
  Peer 192.0.2.18, state: UP, PW class: T-PE2, encapsulation: MPLS,
control-word: enabled

Local interface: Bundle-Ether333.2 up, VC ID: 2
  Peer 192.0.2.18, state: UP, PW class: T-PE2, encapsulation: MPLS,
control-word: enabled
```

Use the **show l2vpn xconnect detail** command to check the state of XCs, PWs, and ACs to confirm the end-to-end operational status of the L2VPN service.

Use the **show mpls ldp neighbor brief** command to verify the state and uptime of LDP neighbors to ensure stable MPLS LDP peering sessions.

Each router shows its direct LDP neighbors only.

Use the **show mpls forwarding** command to examine label entries to confirm correct MPLS label switching and next-hop forwarding for specific traffic.

Use the **show mpls l2transport vc detail** command to check the state of each VC and its local interface to verify the operational status of L2 virtual circuits.

The multisegment pseudowire service is successfully established. All routers confirm operational status for the cross-connects, pseudowires, and virtual circuits. Verification outputs provide evidence of fully configured MS-PWs across segments.

10 Traffic Engineering, Load Balancing, and Protection for L2 Services

Topics:

- Preferred tunnel path
- How MPLS PW traffic load balancing works on P routers
- L2VPN traffic load balancing on PE routers
- G.8032 Ethernet ring protection switching
- VPLS preferred path over SR-TE policy

Outlines traffic engineering principles, load balancing strategies, and protection mechanisms for L2 services, providing guidance on optimizing service delivery, ensuring network resilience, and enhancing performance across various deployment scenarios.

Preferred tunnel path

Introduces preferred tunnel path concepts, highlighting key benefits, deployment restrictions, and step-by-step configuration for MPLS-TE tunnels and VPLS integration to ensure optimal Layer 2 forwarding, including end-to-end service setup.

Preferred tunnel path is a configuration mechanism that

- influences transport path selection for Layer 2 service traffic
- maps pseudowires to specific traffic engineering tunnels, and
- targets network traffic paths through specific tunnels.

Feature history

The feature history table lists release support for this feature.

Table 35: Feature History Table

Feature Name	Release Information	Feature Description
VPLS over preferred TE and MPLS OAM	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
VPLS over preferred TE and MPLS OAM	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
VPLS over preferred TE and MPLS OAM	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) The VPLS over preferred TE and MPLS OAM feature with MPLS OAM capabilities helps you troubleshoot MPLS networks. *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
VPLS over preferred TE and MPLS OAM	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) VPLS over preferred TE and MPLS OAM is now supported on the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
VPLS over preferred TE and MPLS OAM	Release 24.3.1	Introduced in this release on: Fixed Systems (8700 [ASIC: P100])(select variants only*) The VPLS over preferred TE and MPLS OAM feature with MPLS OAM capabilities helps you troubleshoot MPLS networks. *This feature is now supported on: • 8711-32FH • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
VPLS over preferred TE and MPLS OAM	Release 24.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100])(select variants only*) *This feature which uses MPLS OAM capabilities to troubleshoot MPLS networks is now supported on the Cisco 8212-48FH-M routers.
VPLS over preferred TE and MPLS OAM	Release 24.1.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature which uses MPLS OAM capabilities to troubleshoot MPLS networks is now supported on the Cisco 88-LC1-36EH line cards.

Feature Name	Release Information	Feature Description
VPLS over preferred TE and MPLS OAM	Release 7.5.2	Based on your network traffic pattern, you can configure the preferred Traffic Engineering (TE) tunnel path between Provider Edge (PE) routers participating in the same Virtual Private LAN Services (VPLS). You optimize network resource utilization and performance when you set an explicit path on the PE router to direct traffic flow to a specific destination PE router. With VPLS, you now have MPLS-OAM capabilities for troubleshooting MPLS networks: • MPLS LSP Ping • MPLS LSP Traceroute • Flow-Aware Transport (FAT) Pseudowires (PW) This functionality adds the following command: control-word

Preferred tunnel path benefits

Lists the operational benefits of mapping VPLS pseudowires to preferred traffic engineering tunnel paths.

Mapping VPLS pseudowires to explicit, preferred traffic engineering tunnel paths on a PE router offers key operational advantages for MPLS networks:

- Optimizes network resource utilization and performance by directing traffic flows to specific destination PE routers using explicit path configuration.
- Supports fallback enable options to maintain service continuity when the preferred tunnel path is unavailable.
- Enables efficient troubleshooting in MPLS networks by leveraging MPLS Operations, Administration, and Maintenance (OAM) capabilities.

Preferred tunnel path restrictions

Outlines the restrictions that apply when configuring a preferred tunnel path for MPLS encapsulation.

If you plan or configure a preferred tunnel path, ensure the following requirement:

- The preferred tunnel path configuration applies only to MPLS encapsulation.

Configure preferred tunnel path

Configure a preferred tunnel path for a pseudowire by attaching a tunnel interface to the pseudowire class. This ensures traffic traverses the designated tunnel and disables fallback paths if the intended tunnel is unavailable.

Specify a preferred MPLS tunnel for a pseudowire, ensuring traffic uses the designated path and disables fallback alternatives.

Use this task to define a preferred MPLS Traffic Engineering (TE) tunnel for a pseudowire. When configured, the pseudowire directs traffic through the specified tunnel interface and prevents traffic from using alternate routes if the preferred tunnel is unavailable.

Before you begin

- Confirm that MPLS encapsulation is enabled.
- Verify that the required tunnel interface exists and is operational.

Follow these steps to configure a preferred tunnel path for a pseudowire:

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters the global configuration mode.

2. Enter L2VPN configuration mode.

   Example

   ```
   Router(config)# l2vpn
   ```

   Enters the L2VPN configuration mode.

3. Define the pseudowire class to configure tunnel preference.

   Example

   ```
   Router(config-l2vpn)# pw-class PATH1
   ```

   Defines the pseudowire class, PATH1, to configure tunnel preferences.

4. Specify MPLS as the encapsulation type for the pseudowire.

   Example

   ```
   Router(config-l2vpn-pwc)# encapsulation mpls
   ```

   Specifies MPLS as the encapsulation type for the pseudowire.

5. Set the preferred tunnel interface and disable fallback options.

   Example

   ```
   Router(config-l2vpn-pwc-mpls)# preferred-path interface tunnel-te1 fallback
   disable
   ```

Here, interfacetunnel-te1 specifies the preferred MPLS TE tunnel and fallback disable prevents using alternate paths if the preferred tunnel is unavailable.

6. Commit and save the configuration.

Example

```
Router(config-l2vpn-pwc-mpls)# commit
```

Saves and applies the configuration changes.

Configure MPLS-TE tunnel for VPLS

Configure the MPLS traffic engineering tunnel that provides the preferred path for VPLS traffic.

Set up an MPLS-TE tunnel to ensure a preferred and optimized path for VPLS traffic across the network.

This task outlines the CLI commands required to configure an MPLS-TE tunnel for VPLS. The tunnel enables traffic engineering by directing VPLS packets through a specified path, improving network reliability and control.

Before you begin

- Confirm that MPLS traffic engineering is enabled.
- Ensure routing and interface addressing are planned.
- Identify the tunnel interface name and destination IP address.

Follow these steps to configure an MPLS-TE tunnel for VPLS:

Procedure

1. Enter the tunnel interface configuration mode.

Example

```
Router(config)# interface tunnel-te1
```

Enters the configuration mode for the MPLS-TE tunnel interface.

2. Set the tunnel to use an IPv4 unnumbered loopback interface.

Example

```
Router(config-if)# ipv4 unnumbered Loopback0
```

Sets the tunnel to use an IPv4 unnumbered loopback interface.

3. Specify the signaled bandwidth for the tunnel.

Example

```
Router(config-if)# signalled-bandwidth 50
```

Specifies the signaled bandwidth for the tunnel.

4. Define the tunnel destination IP address.

   Example

   ```
   Router(config-if)# destination 10.12.12.12
   ```

   Defines the destination IP address for the tunnel.

5. Specify the explicit path option for the tunnel.

   Example

   ```
   Router(config-if)# path-option 1 explicit name FC1
   ```

   Specifies the explicit path option.

6. Save and apply configuration changes.

   Example

   ```
   Router(config)# commit
   ```

   Saves and applies the configuration changes.

Configure VPLS over preferred TE tunnel

Configure VPLS service traffic to use a preferred traffic engineering tunnel for optimal redundancy and controlled traffic flow. This procedure ensures your VPLS bridge-domain routes traffic through the designated MPLS-TE tunnel.

Assign a preferred MPLS-TE tunnel to carry VPLS service traffic, optimizing network redundancy and performance.

You perform this task to ensure VPLS traffic follows a specified traffic engineering tunnel, supporting redundancy and traffic optimization in a service provider environment.

Before you begin

- Confirm that an MPLS-TE tunnel is configured and available.
- Ensure that your VPLS bridge-domain design is complete and meets service requirements.

Follow these steps to configure VPLS over a preferred TE tunnel:

Procedure

1. Define the Layer 2, specify the VLAN encapsulation for the interface, and configure ingress tag rewrite for VLAN.

   Example

   ```
   Router# configure
   Router(config)# interface FourHundredGigE0/0/0/0.1 l2transport
   Router(config-subif)# encapsulation dot1q 100
   Router(config-subif)# rewrite ingress tag pop 1 symmetric
   ```

2. Exit subinterface configuration mode.

Example

```
Router(config-subif)# exit
```

3. Create pseudowire class for the VPLS pseudowire and specify MPLS encapsulation for the pseudowire.

   Example

   ```
   Router(config)# l2vpn
   Router(config-l2vpn)# pw-class c
   Router(config-l2vpn-pwc)# encapsulation mpls
   ```

4. Enable the control word and flow-label-based load balancing for the pseudowire.

   Example

   ```
   Router(config-l2vpn-pwc-mpls)# control-word
   Router(config-l2vpn-pwc-mpls)# load-balancing
   Router(config-l2vpn-pwc-mpls-load-bal)# flow-label both
   ```

5. Specify the preferred MPLS-TE tunnel path for the pseudowire.

   Example

   ```
   Router(config-l2vpn-pwc-mpls)# preferred-path interface tunnel-te1
   ```

6. Exit pseudowire class configuration mode.

   Example

   ```
   Router(config-l2vpn-pwc-mpls)# exit
   Router(config-l2vpn-pwc)# exit
   ```

7. Define the VPLS bridge group and bridge domain, associate the access interface with the bridge domain.

   Example

   ```
   Router(config-l2vpn)# bridge group bg bridge-domain bd
   Router(config-l2vpn-bg-bd)# interface FourHundredGigE0/0/0/0.1
   Router(config-l2vpn-bg-bd-ac)# exit
   ```

8. Define a pseudowire neighbor for the bridge domain and associate the pseudowire class with the neighbor pseudowire.

   Example

   ```
   Router(config-l2vpn-bg-bd)# neighbor 10.12.12.12 pw-id 100
   Router(config-l2vpn-bg-bd-pw)# pw-class c
   ```

9. Verify the VPLS over preferred TE tunnel configuration.

Example

```
Router(PE1)# show l2vpn xconnect group XCON_1
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description
      ST
------------------------   ------------------------------
------------------------------
XCON_1     XCON1_P2P2 UP   Hu0/1/0/0.1            UP       172.16.0.1
1000   UP
                                                           Backup
                                                           192.168.0.1
1000   SB
------------------------------------------------------------------------------------

Router(PE2)# show l2vpn xconnect group XCON_1
Tue Jan 17 15:36:12.327 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description
      ST
------------------------   ------------------------------
--------------------------
XCON_1     XCON1_P2P2 UP   BE100.1                UP       10.0.0.1     1000
   UP
------------------------------------------------------------------------------------

Router(PE3)# show l2vpn xconnect group XCON_1
Tue Jan 17 15:38:04.785 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description
      ST
------------------------   ------------------------------
------------------------------
XCON_1     XCON1_P2P2 DN   BE100.1                UP       10.0.0.1     1000
   SB
------------------------------------------------------------------------------------

Router# show l2vpn xconnect summary
Number of groups: 3950
Number of xconnects: 3950
  Up: 3950  Down: 0  Unresolved: 0 Partially-programmed: 0
  AC-PW: 3950  AC-AC: 0  PW-PW: 0 Monitor-Session-PW: 0
Number of Admin Down segments: 0
Number of MP2MP xconnects: 0
  Up 0 Down 0
  Advertised: 0 Non-Advertised: 0
Number of CE Connections: 0
  Advertised: 0 Non-Advertised: 0
Backup PW:
  Configured   : 3950
  UP           : 0
  Down         : 0
  Admin Down   : 0
```

```
  Unresolved    : 0
  Standby       : 3950
  Standby Ready: 0
Backup Interface:
  Configured    : 0
  UP            : 0
  Down          : 0
  Admin Down    : 0
  Unresolved    : 0
  Standby       : 0
```

VPLS service traffic is routed via the preferred MPLS-TE tunnel. The configuration changes are committed, and verification output confirms the expected state.

How MPLS PW traffic load balancing works on P routers

Explains MPLS pseudowire traffic load balancing on P routers by describing control word and flow label mechanisms, load-balancing topologies, and provides configuration procedures for effective traffic distribution and increased network efficiency.

An L2VPN PE sends frames over an MPLS pseudowire (PW) by encapsulating each Ethernet frame into an MPLS frame. The frame includes at least one PW label and often an IGP label to reach the remote PE. The MPLS network uses one of several available paths to transport the frame to the remote PE.

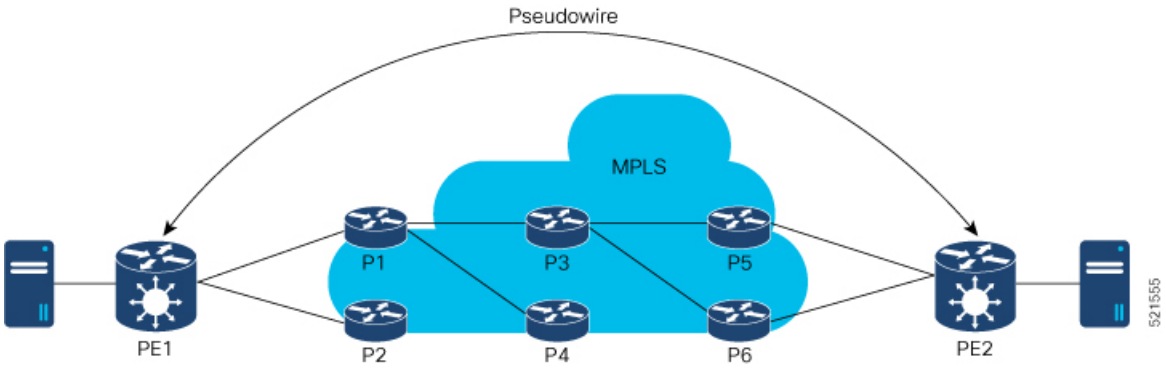
Summary

The key components involved in the process are:

- L2VPN provider edge (PE) routers: Encapsulate Ethernet frames into MPLS frames and initiate the pseudowire traffic across the network.
- Provider routers (P routers): Select next-hop routers and balance traffic over eligible links using path selection and hashing algorithms.
- IGP topology and MPLS TE tunnel paths: Determine the available equal-cost paths through the network for traffic distribution.

MPLS pseudowire (PW) traffic load balancing ensures that provider routers distribute traffic efficiently across multiple paths while maintaining packet order for each traffic flow.

Workflow



These stages describe how MPLS PW traffic load balancing works on P routers.

1. Path selection by PE: PE1 selects P1 or P2 as the first MPLS P router toward PE2.

2. Next-hop selection by P routers: If PE1 selects P1, P1 then chooses P3 or P4 for the next hop.
3. Determining available paths: The IGP topology and MPLS TE tunnel path determine which paths are eligible for traffic.
4. Balancing traffic loads: Providers balance traffic across multiple links to prevent.
5. Hash-based path assignment: The core hashing algorithm assigns pseudowire traffic to specific paths.
6. Handling high-bandwidth flows: Multiple high-bandwidth pseudowires can map to the same physical link, which may cause congestion.
7. Preserving packet order: Packets from one traffic flow must follow the same path to prevent out-of-order frames.
8. Enhancing load balancing: Control word and flow label methods are applied to improve the accuracy of MPLS PW load balancing.

Result

MPLS PW traffic is efficiently distributed across all eligible paths in the provider network, ensuring packets from a given flow remain in order and network resources are optimally utilized.

Load balance MPLS PW traffic using control word and flow label

Explains the methods used to optimize load balancing for MPLS pseudowire traffic, including control word and flow label, and summarizes feature support across hardware platforms.

Control word and flow label load balancing is a pseudowire forwarding method that

- prevents misidentification of Ethernet pseudowire packets as IP packets
- identifies individual packet flows within a pseudowire, and
- improves traffic distribution across ECMP paths or link-bundled paths in the core.

Feature history

The feature history table lists release support for this feature.

Table 36: Feature History Table

Feature Name	Release Information	Feature Description
Load Balance MPLS PW Traffic using Flow-Label	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Load Balance MPLS PW Traffic using Control-Word	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Feature Name	Release Information	Feature Description
Load Balance MPLS PW Traffic using Flow-Label	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Load Balance MPLS PW Traffic using Control-Word	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Load Balance MPLS PW Traffic using Flow-Label	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Load Balance MPLS PW Traffic using Flow-Label	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The load balancing with flow-label functionality is now extended to the Cisco 8712-MOD-M routers.
Load Balance MPLS PW Traffic using Control-Word	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The load balancing with control-word functionality is now extended to the Cisco 8712-MOD-M routers.
Load Balance MPLS PW Traffic using Flow-Label	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The load balancing with flow-label functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
Load Balance MPLS PW Traffic using Control-Word	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The load balancing with control-word functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Load Balance MPLS PW Traffic using Flow-Label	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The load balancing with flow-label functionality is now extended to routers with the 88-LC1-36EH line cards.
Load Balance MPLS PW Traffic using Control-Word	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The load balancing with control-word functionality is now extended to routers with the 88-LC1-36EH line cards.
Load Balance MPLS PW Traffic using Flow-Label	Release 7.3.15	The flow-label provides the capability to identify individual flows within a pseudowire and provides routers the ability to use these flows to load balance traffic. Individual flows are determined by the hashing algorithm configured under L2VPN. Similar packets with the same source and destination addresses are all said to be in the same flow. A flow-label is created based on indivisible packet flows entering a pseudowire and is inserted as the lowermost label in the packet. Routers can use the flow-label for load balancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core. The **flow-label** keyword is added.

Feature Name	Release Information	Feature Description
Load Balance MPLS PW Traffic using Control-Word	Release 7.3.15	This feature allows the router to correctly identify the Ethernet PW packet over an IP packet, thus preventing the selection of wrong equal-cost multipath (ECMP) path for the packet that leads to the misordering of packets. This feature inserts the control word keyword immediately after the MPLS label to separate the payload from the MPLS label over a PW. The control word carries layer 2 control bits and enables sequencing. The **control-word** keyword is added.

Load balancing using control word

Provides details on how the control word feature prevents label switching routers from misidentifying Ethernet pseudowire packets as IP packets and ensures correct ECMP path selection.

The control word feature addresses issues in MPLS networks where label switching routers (LSRs) may misidentify Ethernet pseudowire packets as IP packets, resulting in incorrect load balancing and packet misordering. This feature is commonly used in Ethernet over MPLS (EoMPLS) environments to ensure reliable and efficient packet transport.

Key attributes and benefits:

- Prevents LSRs from mistaking Ethernet pseudowire packets with MAC addresses beginning with hexadecimal 0x4 or 0x6 for IPv4 or IPv6 packets.
- Ensures correct load balancing across equal-cost multipath (ECMP) paths by avoiding hash calculations on misidentified packets.
- Inserts a control word immediately after MPLS labels under a pseudowire (PW) class for point-to-point pseudowires, resolving misordering of packets.
- Facilitates seamless migration of legacy ATM and Frame Relay services to MPLS or IP core networks without service interruption.
- Allows two L2VPN Provider Edge (PE) devices at different sites to reliably transport Layer 2 VPN traffic over an MPLS core.

Usage notes:

- The **control-word** keyword must be configured under a pw-class attached to a point-to-point pseudowire.
- This feature is especially important when encapsulating and transporting Ethernet frames over pseudowires with MPLS.

Load balancing using flow label

Provides details on how flow label identifies individual pseudowire flows to enable improved load distribution.

Routers can use the flow label to achieve more granular load balancing of pseudowire traffic. The flow label enables identification of individual packet flows within a pseudowire, allowing improved distribution across available paths.

Key points:

- Routers typically load balance traffic based on the lowermost label, which is identical for all flows on a given pseudowire—leading to asymmetric load balancing.
- A flow refers to a sequence of packets sharing the same source and destination pair.
- The flow label provides the capability to identify individual flows within a pseudowire and is inserted as the lowermost label in the packet.
- Routers use the flow label for load balancing, enabling better traffic distribution across ECMP or link-bundled paths in the core.

Benefits:

- Enables effective load distribution in networks using pseudowires.
- Reduces risk of asymmetric traffic patterns.
- Improves network resource utilization.

How MPLS PW traffic load balancing works

Describes how MPLS pseudowire (PW) traffic is distributed over ECMP and bundle links using a reference topology.

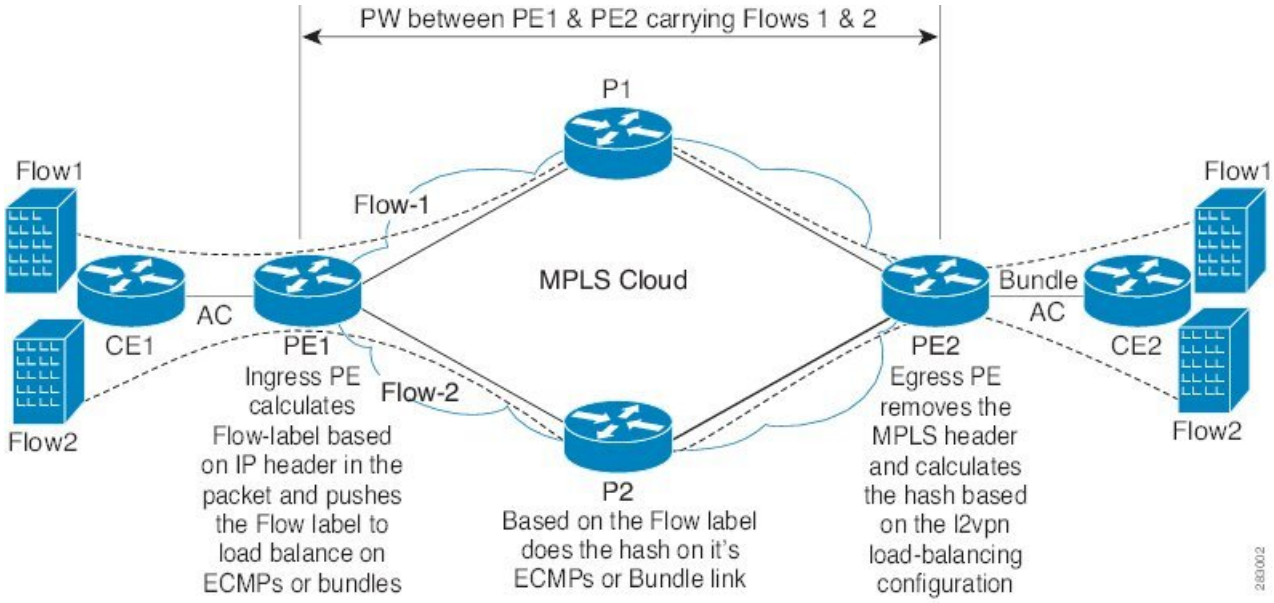
This example illustrates two flows distributing over ECMPs and bundle links.

Summary

The key components involved in the process are:

- MPLS pseudowires (PWs): Virtual circuit connections that encapsulate customer traffic across the provider network.
- ECMP paths: Multiple network paths with equal cost that allow traffic to be distributed for load sharing.
- Bundle links: Groups of physical links that operate as a single logical connection, further enabling flow distribution within an ECMP path.

Workflow



These stages describe how MPLS PW traffic load balancing works.

1. Topology establishment: The network is configured with ECMP paths and bundle links between ingress and egress provider edge (PE) routers.

2. PW traffic initiation: Pseudowire tunnels are created, and multiple traffic flows (such as customer Ethernet frames) enter the provider network at the ingress PE.
3. Flow hashing and selection: The ingress router uses flow-based hashing algorithms to select which ECMP path and which bundle link a specific flow follows.
4. Traffic distribution: Individual flows are distributed across the available ECMP and bundle links, maximizing available bandwidth and enabling redundancy.
5. Traffic egress: At the remote PE, packets are reassembled and forwarded to the customer edge.

Result

This process ensures that MPLS PW traffic is efficiently balanced across the network, improving resource utilization and providing better fault tolerance through the use of ECMP and bundle links.

Configure load balancing using control word and flow label

Configure a pseudowire class with control word and flow label load balancing.

Set up load balancing for pseudowire traffic using both control word and flow label to optimize data distribution across multiple links.

Use this task to ensure traffic is efficiently balanced when deploying pseudowire-based Layer 2 VPNs. The control word and flow label enhance granular load distribution between endpoints.

Before you begin

- Confirm that the pseudowire class and xconnect group are planned.
- Ensure all prerequisite configurations and connectivity checks are complete.

Follow these steps to configure load balancing with control word and flow label:

Procedure

1. Define the Layer 2 VPN pseudowire class and enable required features.

   Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class path1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc)# control-word
Router(config-l2vpn-pwc-mpls)# load-balancing flow-label both
```

2. Set up the xconnect group and peer connection.

   Example

```
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p vlan1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/0/0/1.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.2 pw-id 2000
Router(config-l2vpn-xc-p2p-pw)# pw-class path1
Router(config-l2vpn-xc-p2p-pw)# commit
```

3. Add a point-to-point VLAN instance.

Example

```
Router(config-l2vpn-xc)# p2p vlan1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/0/0/1.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.2 pw-id 2000
Router(config-l2vpn-xc-p2p-pw)# pw-class path1
Router(config-l2vpn-xc-p2p-pw)# commit
```

4. Specify the neighbor and assign the pseudowire class.

Example

```
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.2 pw-id 2000
Router(config-l2vpn-xc-p2p-pw)# pw-class path1
Router(config-l2vpn-xc-p2p-pw)# commit
```

5. Review the running configuration to verify that settings are applied

Example

```
l2vpn
 pw-class path1
  encapsulation mpls
   control-word
   load-balancing
    flow-label both
  !
 !
 xconnect group grp1
  p2p vlan1
   interface HundredGigE0/0/0/1.2
   neighbor ipv4 10.0.0.2 pw-id 2000
    pw-class path1
   !
```

This section shows the running configuration.

Load balancing using control word and flow label is successfully configured when the commit command completes and the running configuration reflects the desired setup.

L2VPN traffic load balancing on PE routers

Details L2VPN traffic load balancing techniques on PE routers, covering supported VLAN tag formats, VPWS and VPLS service interactions, BUM traffic handling, non-IP hash configurations, and hash algorithms for L2 and L3 interface traffic.

L2VPN traffic load balancing on PE routers is a forwarding behavior that

- balances VPWS and VPLS traffic across outgoing interfaces
- uses different hash inputs for different traffic groups, and
- applies to ECMP paths and link aggregation bundle members.

Feature history

The feature history table lists release support for this feature.

Table 37: Feature History Table

Feature Name	Release Information	Feature Description
VLAN tag format support for load balancing for line cards and routers	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8404-SYS-D router.
VLAN tag format support for load balancing for line cards and routers	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8711-48Z-M
VLAN tag format support for load balancing for line cards and routers with the A100 and K100-based Silicon One ASICs	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) Introduced support for the following VLAN tags on line cards and routers with the A100 and K100-based Silicon One ASICs: • Single VLAN tag 0x88A8 • QinQ with outer 0x8100 and inner 0x8100 • QinQ with outer 0x9100 and inner 0x8100 Introduced support for BUM traffic in VPLS service load balancing. *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
Introduced New VLAN Tag Format Support for Load Balancing	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) Introduced support for the following VLAN tags on line cards and routers with Q100, Q200, and P100 based Silicon One ASIC: • Single VLAN tag 0x88A8 • QinQ with outer 0x8100 and inner 0x8100 • QinQ with outer 0x9100 and inner 0x8100 Introduced support for BUM traffic in VPLS service load balancing.
Extended L2VPN Traffic Load Balancing Support for line cards and routers with the P100 based Silicon One ASIC	Release 24.1.1	Introduced support for the following VLAN tags on line cards and routers with P100 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ outer 0x88A8 and inner 0x8100 Introduced support for QinQ with outer 0x8100 and inner 0x8100 on line cards and routers with Q200 based Silicon One ASIC.
Extended L2VPN Traffic Load Balancing Support for line cards and routers with the Q200 based Silicon One ASIC	Release 7.5.1	Introduced support for the following VLAN tags on line cards and routers with Q200 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ outer 0x88A8 and inner 0x8100
L2VPN Traffic Load Balancing on PE Router	Release 3.7.1	Distributes L2 VPN traffic across multiple physical links or paths. Introduced support for the following VLAN tags on line cards and routers with Q100 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ with outer 0x88A8 and inner 0x8100

PE router load-balancing behavior

A PE router balances L2VPN traffic across network paths using established load-balancing methods. Cisco 8000 routers use different methods for VPWS traffic, unicast VPLS traffic, and BUM traffic. Load balancing works independently of whether connectivity is point-to-point or multipoint. Routers apply load balancing over ECMP paths and LAG bundle members.

Supported VLAN tag formats for load balancing

Lists the VLAN tag formats supported by Cisco 8000 for load balancing, including applicable software releases for each platform.

The Cisco 8000 platform load balances traffic when it is either not VLAN tagged or is tagged in a supported VLAN format. The following table lists the VLAN tag formats and the minimum software release required for each Cisco 8000 platform:

Table 38: Supported VLAN Tag Formats

VLAN Format	Q100 supports VLAN from Release	Q200 supports VLAN from Release	P100 supports VLAN from Release	K100 supports VLAN from Release	A100 supports VLAN from Release
Single VLAN Tag 0x8100	3.7.1	7.5.1	24.1.1	24.1.1	25.1.1
Single VLAN Tag 0x88A8	24.3.1	24.3.1	24.3.1	24.3.1	25.1.1
QinQ outer 0x88A8, inner 0x8100 (double VLAN tags)	3.7.1	7.5.1	24.1.1	24.1.1	25.1.1
QinQ outer 0x8100, inner 0x8100 (double VLAN tags)	24.3.1	24.1.1	24.3.1	24.3.1	25.1.1
QinQ outer 0x9100, inner 0x8100 (double VLAN tags)	24.3.1	24.3.1	24.3.1	24.3.1	25.1.1

Load balancing scenarios for VPWS and VPLS unicast traffic

Provides detailed reference information about how the router handles load balancing for VPWS service and VPLS unicast traffic, including packet processing, outgoing interface selection, and relevant hash mechanisms.

The router performs load balancing on outgoing interfaces and bundle members in the following scenarios:

- Ethernet frames entering from a Layer 2 main or sub-interface:
  - Switched out to another Layer 2 interface that is part of a bundle, or
  - Routed out to Layer 3 interfaces with MPLS encapsulation.
- MPLS labeled PW and EVPN traffic entering from a Layer 3 interface:
  - After label disposition, customer Ethernet frames may be:

- Switched out to a Layer 2 main or sub-interface that is part of a bundle, or
- Routed out to Layer 3 interfaces with MPLS encapsulation.

The router parses packets to identify required headers for generating a load balance hash, which determines the path for routing traffic across the network. The hashing process varies based on whether the traffic is received on Layer 2 or Layer 3 interfaces.

For further information on load balance hash mechanisms:

- For load balance hash on traffic received from L2 interfaces, refer to Hash for Traffic Received on L2 Interface.
- For load balance hash on traffic received from L3 interfaces, refer to Hash for Traffic Received on L3 Interface.

BUM traffic in VPLS service load balancing

Lists the source and load balancing behaviors of BUM traffic in VPLS service scenarios.

The router performs BUM Traffic load balancing on outgoing interfaces and bundle members in these scenarios:

- Sending traffic to an L2 interface on a bundle
  - The router does not perform load balancing over bundle members.
  - All traffic sent to an L2 main or sub-interface is pinned to a single bundle member, selected based on the main or sub-interface ID.
- Sending BUM traffic over an MPLS pseudowire (PW)
  - Traffic is routed to L3 interfaces.
  - The router performs ECMP and bundle load balancing on PW traffic.
  - Load balancing uses hashing based on:
    - PW VC label and flow label
    - 12 bytes of the PW payload. If control word (CW) is enabled, this includes 4 bytes of CW and 8 bytes of the inner Ethernet MAC address.
- Sending BUM traffic to an EVPN MPLS network
  - Traffic is encapsulated with:
    - EVPN label
    - ETREE leaf label or Ethernet Segment split horizon label
    - MPLS encapsulated traffic is routed to L3 interfaces.
    - ECMP and bundle load balancing are performed on EVPN MPLS encapsulated traffic.
  - Load balancing uses hashing based on:
    - EVPN label
    - ETREE leaf label or Ethernet Segment split horizon label
    - 12 bytes of the PW payload. If CW is enabled, this includes 4 bytes of CW and 8 bytes of the inner Ethernet MAC address.

Impact of disabling MPLS non-IP hash mode

Provides details on the behavior and impact on load balancing when MPLS non-IP hash mode is disabled.

When MPLS non-IP hash mode is disabled in a configuration:

- The Ethernet MAC address of BUM (Broadcast, Unknown-unicast, Multicast) traffic is not used for load balance hashing.
- As a result, the router relies on other header information for hashing decisions.
- This change can affect how BUM traffic is distributed across the network, potentially leading to altered load balancing behavior compared to configurations where non-IP hash mode is enabled.

Purpose: This reference helps you understand the changes in traffic distribution and hashing behavior when MPLS non-IP hash mode is disabled. Use this information to assess network design choices and troubleshoot relevant load balancing outcomes.

Hash fields and load balancing for traffic received on L2 interfaces

Provides details about hash field selection, IP and non-IP classification, and load balancing behaviors for traffic received on Layer 2 (L2) interfaces.

Traffic classification for hash calculation

For traffic received on an L2 interface, hashing depends on the headers present in the packet stack. Routers generate the load balance hash using designated fields based on packet stack headers.

IP traffic on L2 interface

An Ethernet frame is classified as IP traffic if:

- The Ethernet header contains no more than two VLAN tags.
- The Ethernet header either has no VLAN tag or has VLAN tags in a supported format. For details, see Supported VLAN Tag Formats for Load Balancing.
- No more than ten MPLS labels are placed in front of the IP header.

Non-IP traffic on L2 interface

Traffic that does not meet the above criteria is classified as non-IP traffic on the L2 interface.

Hash fields used for load balancing

- L2 hash fields: Source MAC address, destination MAC address, and outer VLAN ID.
- L3 hash fields: Source IP address, destination IP address, and source and destination ports from the Layer 4 header.

Load balancing for IP traffic

- Q100 and Q200 Silicon One ASIC platforms:
  - Before Release 24.2.1, L2 hash fields were used. L3 fields were included for limited traffic types.
  - Starting with Release 24.2.1, both L2 and L3 fields are used for all IP traffic.
- P100, K100, and A100 Silicon One ASIC platforms:
  - Before Release 24.2.1, L2 hash fields were used. L3 fields were included for limited traffic types.

- Starting with Release 24.2.1, both L2 and L3 fields are used for all IP traffic.

Load balancing for non-IP traffic

All non-IP traffic on the L2 interface is load balanced using only L2 hash fields.

If any designated hash fields are missing, the router substitutes the field values with zeros.

Hash criteria for traffic received on L3 interfaces

Lists the hash criteria and classification rules for traffic received on Layer 3 interfaces, including pseudowire configuration, Ethernet frame structure, and ASIC-specific behaviors.

For traffic received on an L3 interface hashing depends on several criteria, including the headers present in the packet stack and whether the Control Word (CW) and Flow Label (FL) are enabled on the pseudowire (PW).

When ethernet frames entering a L3 interface, the router identifies the IP header within the packet based on the packet stack headers.

IP over PW Traffic

An ethernet frame is classified as IP over PW traffic if it meets the following criteria:

- The frame contains an outer ethernet header and an inner ethernet header.
- No more than two MPLS labels are placed between the outer ethernet header and the inner ethernet header.
- There is no control word in front of the inner ethernet header.
- The inner ethernet header contains no more than two VLAN tags.
- The inner ethernet header has no VLAN tag or has VLAN tags in supported format as listed in Supported VLAN Tag Formats for Load Balancing.
- Behind the inner ethernet header, there are no more than ten MPLS labels placed in front of the IP header.

Non-IP over PW Traffic

All traffic that does not meet the IP over PW traffic criteria is classified as non-IP over PW traffic.

Inner Ethernet L2 Hash Fields

The inner ethernet L2 hash fields include the source and destination MAC addresses, and the first VLAN tag of the inner ethernet frame.

Inner Ethernet L2 Hash Fields for NPU2.0 ASIC

The inner ethernet L2 hash fields include the source and destination MAC addresses of the inner ethernet frame.

Inner Ethernet L3 Hash Fields

The inner ethernet L3 hash fields include L3 and L4 headers in the inner ethernet frame. They are source and destination IP addresses, the source and destination ports of the layer 4 header.

Control Word Presence L2 Hash Fields

The 12 bytes of the PW payload, which include 4 bytes of CW followed by the 8 bytes of inner ethernet frame MAC address.

MPLS Hash Fields of Outer Ethernet Frame

All MPLS labels in front of inner ethernet header.

PW Disposition Traffic Load Balancing

1. PW disposition traffic load balance when control word and flow label are both disabled.

IP over PW Traffic Load Balancing

a. On Q100 and Q200 based Silicon One ASIC:

IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

Before Release 24.2.1, inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.

Starting from Release 24.2.1, both inner ethernet L2 hash fields and inner ethernet L3 hash fields are used in hashing.

b. On P100, K100, and A100 based Silicon One ASIC:

Before Release 24.2.1, IP over PW traffic load balance hash uses the inner ethernet L2 hash fields. Inner Ethernet L3 hash fields are also added in the hashing for limited types of traffic.

From Release 24.2.1 to Release 24.3.1, IP over PW traffic load balance hash uses the Inner Ethernet L2 Hash Fields and Inner Ethernet L3 Hash Fields.

Starting from Release 24.3.2, IP over PW traffic load balance hash uses the Inner Ethernet L2 Hash Fields for NPU2.0 ASIC and Inner Ethernet L3 Hash Fields.

Non-IP over PW Traffic Load Balancing

Non-IP over PW traffic load balance hash always uses the inner ethernet L2 hash fields.

2. PW disposition traffic load balance when control word is enabled and flow label disabled.

In this case, all PW traffic is non-IP over PW traffic. Load balance hash uses control word presence L2 hash fields.

3. PW disposition traffic load balance when control word is disabled and flow label enabled.

a. On Q100 and Q200 based Silicon One ASIC:

IP over PW Traffic Load Balancing

IP over PW traffic load balancing hash uses the following fields:

- Before Release 24.2.1, hash uses the inner ethernet L2 hash fields. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.
- In Release 24.2.1, inner ethernet L2 hash fields and inner ethernet L3 hash fields are both used in hashing.
- Starting from Release 24.3.1, inner ethernet L3 hash fields and MPLS hash fields of outer ethernet frame are used in hashing.

Non-IP over PW Traffic Load Balancing

Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

Starting from Release 24.3.1 release, MPLS hash fields of outer ethernet frame are used in hashing.

b. On P100, K100, and A100 based Silicon One ASIC:

IP over PW Traffic Load Balancing

IP over PW traffic load balance hash uses the following fields:

- Before Release 24.4.1, hash uses the inner ethernet L2 hash fields. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.
- Starting from Release 24.4.1, inner ethernet L3 hash fields and MPLS hash fields of outer ethernet frame are used in hashing.

Non-IP over PW Traffic Load Balancing

Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

Starting from Release 24.3.1 release, MPLS hash fields of outer ethernet frame are used in hashing.

4. PW disposition traffic load balance when control word and flow label are both enabled.

   In this case, all PW traffic is Non-IP over PW traffic.

   Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the control word presence L2 hash fields.

   Starting from Release 24.3.1, MPLS hash fields of outer ethernet frame are used in hashing.

G.8032 Ethernet ring protection switching

Describes G.8032 Ethernet Ring Protection Switching, including operational principles, timers, single link failure protection and recovery, configuration procedures for ERPS profiles and instances, restrictions, and practical topology and interconnection examples.

G.8032 Ethernet ring protection switching is a resiliency protocol that

- recovers from link or node failures in Ethernet ring topologies
- uses RAPS messages to coordinate protection switching, and
- helps maintain loop-free traffic forwarding in the ring.

Feature history

The feature history table lists release support for this feature.

Table 39: Feature History Table

Feature Name	Release Information	Feature Description
G.8032 Ethernet Ring Protection Switching	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
G.8032 Ethernet Ring Protection Switching	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
G.8032 Ethernet Ring Protection Switching	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
G.8032 Ethernet Ring Protection Switching	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The G.8032 Ethernet Ring Protection Switching functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
G.8032 Ethernet Ring Protection Switching	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The G.8032 Ethernet Ring Protection Switching functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
G.8032 Ethernet Ring Protection Switching	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) Ethernet Ring Protection Switching (ERPS) protocol, defined in ITU-T G.8032, provides protection for Ethernet traffic in a ring topology, while ensuring that there are no loops within the ring at the Ethernet layer. The loops are prevented by blocking either a predetermined link or a failed link. *This feature introduces the **ethernet ring g8032** and **ethernet ring g8032 profile** commands. This feature is supported on routers with the 88-LC1-36EH line cards.

ERPS recovery behavior

- ERPS helps Ethernet ring topologies recover from link or node failures.
- During a link failure, ERPS reroutes traffic to provide continuous connectivity.
- ERPS simplifies network management and operates independently of control planes.

How G.8032 Ethernet ring protection switching works

Describes the concepts, roles, behaviors, and administrative operations involved in G.8032 Ethernet ring protection switching.

Each Ethernet ring node uses two independent links to connect to its neighbors, forming a physical ring topology suitable for high availability. The ring avoids loops by blocking the RPL under normal operation.

Summary

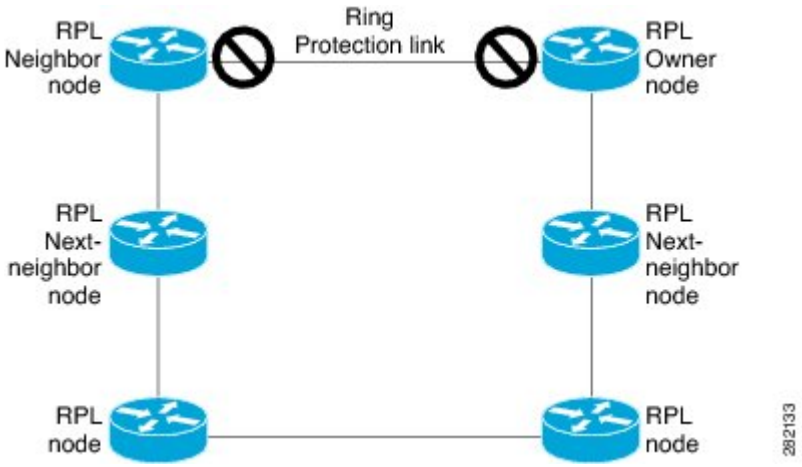
The key components involved in the process are:

- Ethernet ring node: A device connected to adjacent ring nodes using two independent links; participates in protection switching.
- Ring protection link (RPL): A specific link designated to protect against loops in the Ethernet ring by being selectively blocked or unblocked.
- Control protocols and messages (RAPS, CFM): Mechanisms that monitor link health, coordinate role actions, and trigger protection switching during failures or maintenance.

G.8032 Ethernet ring protection switching ensures loop-free, resilient connections in Ethernet ring topologies by coordinating node roles, protective mechanisms, and control protocols.

Workflow

Figure 21: G.8032 Ethernet Ring



The process involves these stages:

1. Normal operation and loop avoidance:
   - The RPL is blocked by the RPL owner, preventing traffic loops on the ring while all other links carry traffic.
   - Nodes continuously send and monitor CFM Continuity Check Messages (CCM) to detect link health.
2. Failure detection and switching:
   - When a link fails, nodes adjacent to the failure detect it using line status and CFM.
   - These nodes immediately block ports facing the failure and generate RAPS signal fail (SF) messages, which propagate through the ring.
   - Upon receiving RAPS SF, the RPL owner unblocks the RPL to restore traffic continuity.
3. Recovery and reversion:
   - When the failed link is restored, its neighboring nodes send RAPS no request (NR) messages.
   - On receiving RAPS NR, the RPL owner blocks the RPL again and informs the ring via RAPS NR, root blocked (RB) messages.
   - All other nodes unblock any previously blocked ports, fully restoring the original topology.
4. Administrative operations:

- Force Switch (FS): Allows operators to force the blocking of a ring-port for immediate maintenance, even during existing failure conditions.
- Manual Switch (MS): Allows manual port blocking unless an FS or SF condition is present. MS is overridden by new FS or SF events.
- Clear command: Cancels existing FS or MS actions; used by the RPL owner to clear non-revertive modes.

Result

The Ethernet ring remains loop-free during normal and fault conditions, automatically restores traffic paths after failures, and allows administrative intervention for maintenance and optimization.

G.8032 ERPS timers

Lists the delay, guard, and hold-off timers that prevent race conditions and unnecessary switching operations.

Types of ERPS timers

G.8032 Ethernet Ring Protection Switching (ERPS) uses several types of timers to avoid race conditions and unnecessary switching operations. These timers help ensure network stability and improve ring protection. The timers and their purposes are outlined below:

- Delay timers: Used by the RPL Owner to verify that the network has stabilized before blocking the RPL.
- Wait-to-Restore (WTR) timer: Activated after a signal failure (SF) condition to ensure the SF isn't intermittent.
  - Operator-configurable.
  - Default time interval: 5 minutes (range: 1 to 12 minutes).
- Wait-to-Block timer: Used after the FS/MS command to verify that no background condition exists.

  May be shorter than the Wait-to-Restore timer.
- Guard timer:
  - Used by all nodes when changing state to block latent or outdated messages from causing unnecessary state changes.
  - Operator-configurable.
  - Default time interval: 500 ms (range: 10 to 2000 ms).
- Hold-off timer:
  - Used by the underlying Ethernet layer to filter out intermittent link faults.
  - Operator-configurable.
  - Default time interval: 0 seconds (range: 0 to 10 seconds).
  - Faults are reported to the ring protection mechanism only if this timer expires.

ERPS operation during link failure

During a link failure, G.8032 ERPS performs one of the following actions to provide continuous connectivity:

- Traffic rerouting: If ERPS is unable to recover from a link failure, it reroutes traffic.

  For more information, see *Protection switching during single link failure* section.
- Wait to recover: ERPS waits to recover from the link failure to prevent unnecessary switching operations.

For more information, see *recovery from single link failure* section.

How ERPS protection switching works during single link failures

Describes how ERPS reroutes traffic during a single link failure in the Ethernet ring.

This example describes the protection switching process during a single link failure.

Summary

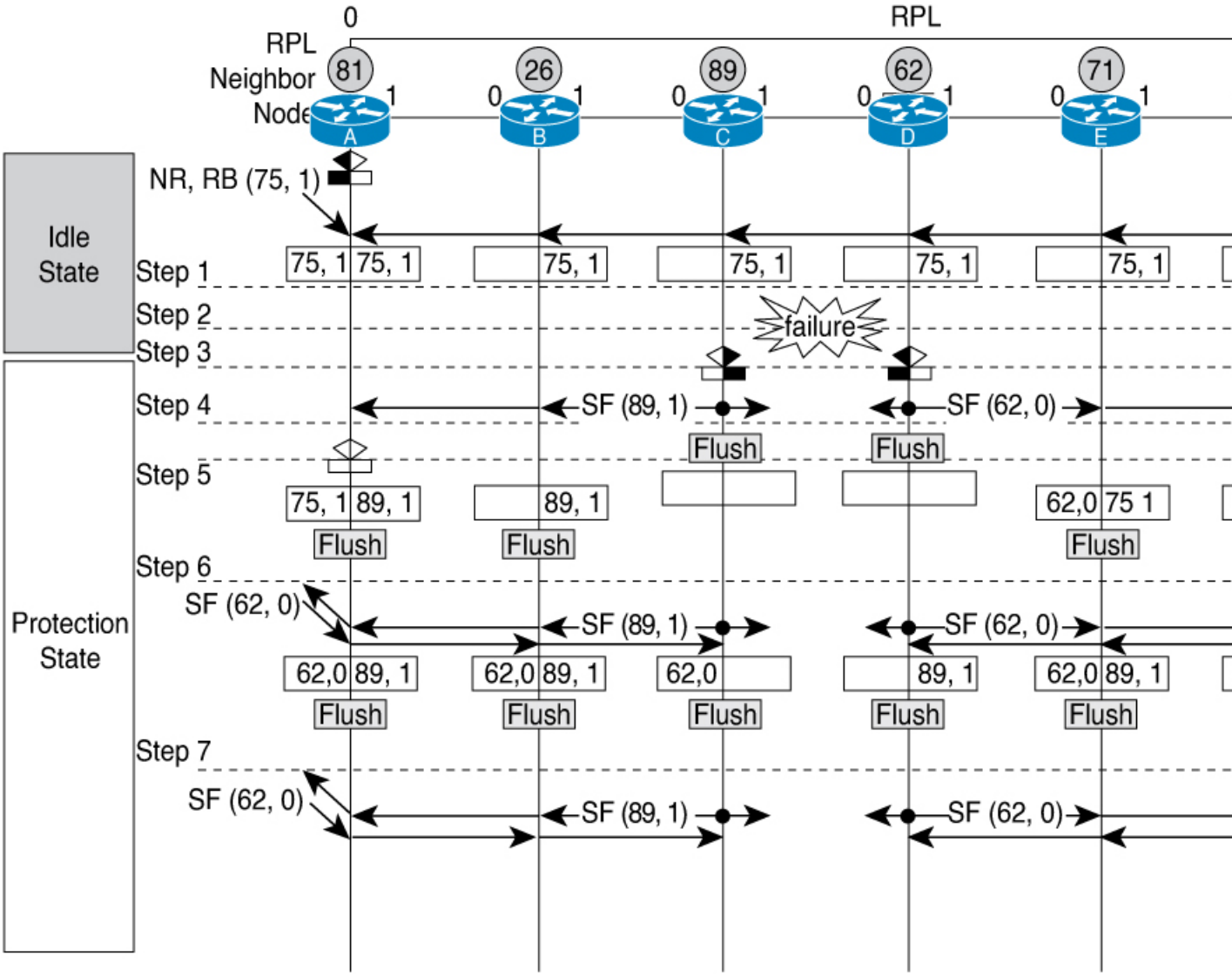
The key components involved in the process are:

- Ethernet ring nodes: Each node in the ring is identified by a unique node ID and port ID. Seven nodes (A to G) participate in the example discussed.
- RPL owner and neighbor nodes: Node G is the RPL (Ring Protection Link) owner, and node A is the RPL neighbor. The RPL is the link between these two nodes.
- Ring Automatic Protection Switching (RAPS) messages: Notification messages used for signaling link and node status during failures.

ERPS rapidly reroutes traffic to maintain network continuity when a single link fails in an Ethernet ring topology. The process is designed to minimize downtime and ensure reliable data transmission between ring nodes.

Workflow

Figure 22: Protection switching during G.8032 single link failure



The process involves the following stages:

1. The Ethernet ring operates normally, with traffic blocked at both ends of the RPL link between nodes A and G.
2. A single link failure occurs between ring nodes C and D.
3. Nodes C and D detect a local signal failure (SF) condition and, after a holdoff interval, block the failed port and flush their forwarding databases (FDB).
4. Nodes C and D begin sending RAPS (SF) messages periodically, including node IDs and blocked port ring (BPR) pairs, on both ring ports.

   For example, node C sends an SF (89,1) message with its node ID and blocked port information.
5. All nodes receiving a RAPS (SF) message perform an FDB flush.

   When the RPL owner (G) and neighbor (A) receive a RAPS (SF) message, each unblocks its end of the RPL and also flushes its FDB.
6. All nodes receiving a second RAPS (SF) message repeat the FDB flush, due to the Node ID and BPR-based mechanism.
7. Once the SF condition stabilizes, RAPS (SF) messages continue, but trigger no further actions.

Result

Traffic is rerouted through the now unblocked protection path, ensuring continued network operation while the failed link remains isolated until it is restored.

Recovery from single link failure

Describes how ERPS restores normal ring operation after a single link failure is cleared, including the key network components and the sequence of recovery stages.

This process applies to ethernet rings composed of multiple nodes, where ERPS ensures network stability and quick restoration of traffic after a single link failure. The protocol prevents loops and recovers from faults by coordinated actions among ring nodes.

Summary

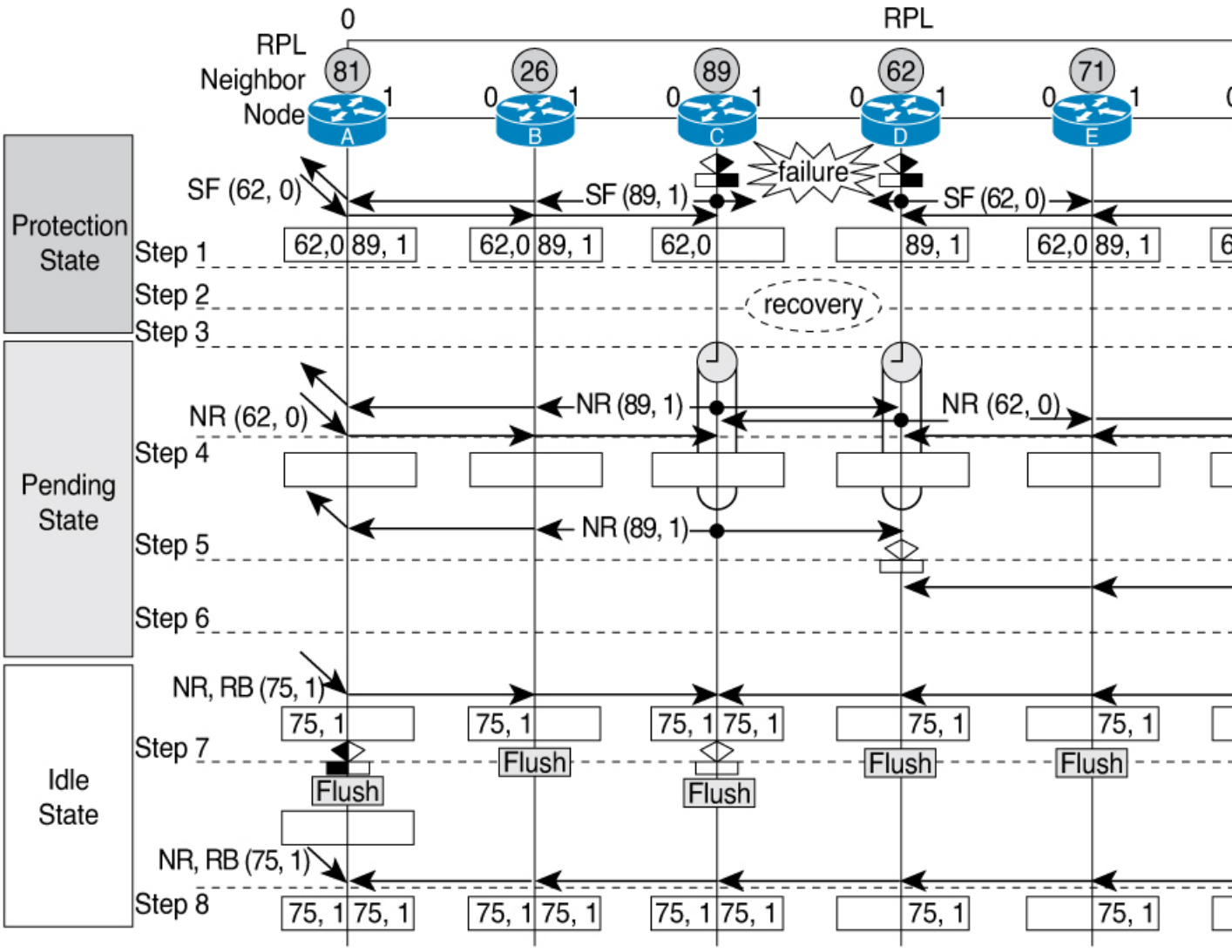
The key components involved in the process are:

- Ethernet ring nodes (A–G): Network devices organized in a ring topology, each assigned unique node IDs and port IDs.
- RPL owner node (G): The node responsible for managing the Ring Protection Link (RPL) and initiating recovery actions.
- RPL neighbor node (A): The node adjacent to the RPL owner, working with the owner to control ring traffic and respond during recovery.
- Ring protection link (RPL): The physical link between nodes A and G, where traffic is blocked at both ends to maintain ring integrity.
- Guard timer and WTR timer: Mechanisms used to manage the timing of message transmissions and port recovery during the process.
- RAPS messages (No Request and RB): Control messages exchanged between nodes to coordinate recovery and unblock ports.

ERPS restores the normal operation of ethernet rings affected by a single link failure by systematically managing blocked ring ports, timers, and messages among ring nodes to reestablish stable traffic flow.

Workflow

Figure 23: Single link failure recovery (Revertive operation)



The process involves these stages:

1. The Ethernet ring consists of seven nodes (A-G), each assigned a unique node ID and port ID. Nodes A and G are designated as the RPL neighbor and owner, respectively, connected by the RPL.
2. Traffic is initially blocked at both ends of the RPL to protect the ring.
3. A link failure occurs between ring nodes C and D.
4. After the failure is cleared, nodes C and D detect the clearing of the SF (Signal Failure) condition. They start the guard timer and transmit periodic RAPS No Request (NR) messages on both ring ports.

> **Note**
>
> The guard timer prevents reception of other RAPS messages during this period.

5. When ring nodes receive RAPS (NR) messages, the node ID and Blocked Port Record (BPR) pair of the receiving port is deleted, and the RPL owner node (G) starts the Wait-To-Restore (WTR) timer.

6. When the guard timer expires on nodes C and D, they accept new RAPS messages. Node D receives a RAPS (NR) message with a higher Node ID from node C and unblocks its non-failed ring port.
7. After the WTR timer expires, the RPL owner node (G) blocks its end of the RPL, sends a RAPS (NR, RB) message with the node ID and BPR pair, and performs an FDB (Forwarding Database) flush.
8. Node C receives the RAPS (NR, RB) message, removes the block on its ring ports, and stops sending RAPS (NR) messages. The RPL neighbor node (A) also receives the RAPS (NR, RB) message and blocks its end of the RPL. Nodes A to F flush their FDBs due to the Node ID and BPR mechanism.
9. The ring resumes stable operation, with all ports unblocked and traffic restored.

Result

The ethernet ring returns to stable normal operation after the RPL owner blocks the RPL and all ring nodes flush their forwarding entries, ensuring network integrity and uninterrupted connectivity.

G.8032 Ethernet ring protection switching restrictions

Outlines the required restrictions when deploying G.8032 Ethernet ring protection switching in your network.

Follow these requirements when using G.8032 Ethernet ring protection switching:

- You must not configure G.8032 ERPS and CFM down-mep on the same sub-interface. If you enable it, then the router displays a syslog message, as shown in the following example:

  You must not configure G.8032 ERPS and CFM down-mep on the same sub-interface. If you enable both on the same sub-interface, the router displays a syslog message similar to the following example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# ethernet ring g8032 test
Router(config-l2vpn-erp)# port0 interface FourHundredGigE0/0/0/6
Router(config-l2vpn-erp)# port1 interface FourHundredGigE0/0/0/10
Router(config-l2vpn-erp)# instance 1
Router(config-l2vpn-erp-inst)# profile test
Router(config-l2vpn-erp-inst)# rpl port0 owner
Router(config-l2vpn-erp-inst)# inclusion-list vlan-ids 1,100
Router(config-l2vpn-erp-inst)# aps-channel
Router(config-l2vpn-erp-inst-aps)# port0 interface FourHundredGigE0/0/0/6.1
Router(config-l2vpn-erp-inst-aps)# port1 interface FourHundredGigE0/0/0/10.1

Router(config)# interface FourHundredGigE0/0/0/6.1 l2transport
Router(config-if)# encapsulation dot1q 1
Router(config-if)# ethernet cfm
Router(config-if-cfm)# mep domain domain1 service link1 mep-id 2

%PLATFORM-SPITFIRE_CFM-3-G8032_VIOLATION : G8032 has been configured for
interface FourHundredGigE0/0/0/6.1
where CFM configuration exists. G8032 config is disabled.
```

  Instead configure the CFM down-mep on the main interface and configure the G.8032 ERPS on the sub-interface.

- Linecards and fixed routers with Q100 and Q200 based Silicon One ASICs do not support G.8032 Ethernet ring protection switching.

Configure G.8032 Ethernet ring protection switching

Explains the configuration elements required for G.8032 Ethernet ring protection switching operation in a network.

G.8032 Ethernet ring protection switching is a network resiliency protocol that

- enables rapid switching between alternate communication paths when a failure occurs
- uses ERPS profiles, instances, parameters, and TCN propagation to control ring behavior, and
- leverages CFM to monitor ring links and bridge domains to define the Layer 2 topology.

Additional reference information

To configure the G.8032 operation, you have to configure ERPS and CFM separately as follows:

- To configure G.8032 Ethernet Ring Protection Switching, configure ERPS and CFM as follows:
  - ERPS configuration:
    - Set the ERPS profile, instance, parameters, and TCN propagation.
    - Designate a (sub)interface as the APS channel.
    - Assign a (sub)interface for CFM monitoring.
  - Check whether an interface is an RPL link and, if so, indicate its RPL node type.

  For configuration details, see:
  - Configure ERPS profile
  - Configure an ERPS instance
  - Configure ERPS parameters
  - Configure TCN propagation
- CFM configuration:
  - Configure CFM with EFD to monitor the ring links.
  - For configuration details, see Configuring CFM MEP.

  > **Note**
  >
  > Each monitored link requires a Maintenance Association (MEP) configured.

- Bridge domains:
  - Create bridge domains to define the Layer 2 topology.
  - RAPS channels are managed in a dedicated management bridge domain, separate from data bridge domains.
- Behavior characteristics (optional):

  Apply attributes to the ERPS instance that differ from default values.

Configure an ERPS profile

Configure an ERPS profile to improve ring resilience and prevent unnecessary switching on the network.

Configure an ERPS profile to enable G.8032 Ethernet ring protection, set timing parameters, and define ring behavior. Perform this task when you need to improve ring resilience and prevent unnecessary switching using ERPS profiles.

Before you begin

Confirm that you have planned the ERPS profile name and timing values.

Follow these steps to configure an ERPS profile:

Procedure

1. Configure a new G.8032 ERPS profile.

   Example

   ```
   Router# configure
   Router(config)# Ethernet ring g8032 profile p1
   ```

   Enables G.8032 ring mode, and enters G.8032 configuration submode.

2. Sets the hold-off timer.

   Example

   ```
   Router(config-g8032-ring-profile)# timer hold-off 5
   ```

   Specifies a time interval (in seconds) for the guard, hold-off, and wait-to-restore timers.

   The hold-off timer prevents unnecessary switching due to short-lived failures on the ring.

3. Specify a non-revertive ring instance.

   Example

   ```
   Router(config-g8032-ring-profile)# non-revertive
   Router(config-g8032-ring-profile)# commit
   ```

   Enables the router to use the current path until an administrator manually reverts to the original path.

You successfully configure an ERPS profile. The configuration is committed, and verification output shows the expected state.

Configure an ERPS instance for a bridge domain

Configure an ERPS instance for a bridge domain.

Set up an ERPS instance to provide Ethernet ring protection switching for a bridge domain in your Layer 2 VPN.

Use this task when you want to configure Ethernet ring protection switching (G.8032) for enhanced network redundancy in a Layer 2 VPN environment.

Before you begin

- Plan your bridge domain and ring port details.
- Ensure you have access to Layer 2 VPN configuration mode on your router.

Follow these steps to configure an ERPS instance for a bridge domain:

Procedure

1. Configure the layer 2 VPN with a bridge group and bridge domain for R-APS channels.

   Example

   ```
   Router# configure
   Router(config)# l2vpn
   Router(config-l2vpn)# bridge group cisco
   Router(config-l2vpn-bg)# bridge-domain bd1
   ```

2. Assign interfaces to each bridge domain.

   For R-APS channels (ports 0 and 1):

   Example

   ```
   Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0.1
   Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/1.1
   ```

3. Configure a bridge domain for data traffic and assign an interface to a bridge domain.

   Example

   ```
   Router(config-l2vpn-bg)# bridge-domain bd2
   Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0.10
   ```

4. Configure an ethernet ring and define an ERPS instance

   Example

   ```
   Router(config-l2vpn)# ethernet ring g8032 r1
   Router(config-l2vpn-erp)# instance 1
   ```

5. Set up an ERPS profile.

   Example

   ```
   Router(config-l2vpn-erp-instance)# profile p1
   ```

   Specifies associated Ethernet ring G.8032 profile.

6. Specify the RPL port and designate it as a neighbor.

   Example

   ```
   Router(config-l2vpn-erp-instance)# rpl port0 neighbor
   ```

   Specifies one ring port on the local node as RPL owner, neighbor, or next-neighbor.

7. Configure VLAN inclusion for the ERPS instance, enable Automatic Protection Switching (APS) channel, and set the priority level for the APS protocol.

Example

```
Router(config-l2vpn-erp-instance)# inclusion-list vlan-ids e-g
Router(config-l2vpn-erp-instance)# aps-channel
Router(config-l2vpn-erp-instance-aps)# level 5
```

8. Assign ports to the G.8032 APS channel interface:

   For port0:

   Example

   ```
   Router(configl2vpn-erp-instance-aps)# port0 interface GigabitEthernet 0/0/0/0.1
   ```

   For port1:

   Example

   ```
   Router(config-l2vpn-erp-instance-aps)# port1 interface GigabitEthernet
   0/0/0/1.1
   ```

Once your configuration is committed, the ERPS instance is active and can be verified to ensure expected state and protection switching behavior.

Configure ERPS parameters

Configure ERPS parameters for the ring instance to protect Ethernet rings and monitor link status.

Set up ERPS parameters to safeguard ring links and configure monitoring.

Perform this task when you need to enable ring protection and monitoring on an existing ERPS instance.

Before you begin

Confirm that the ERPS instance exists in your device configuration.

Follow these steps to configure ERPS parameters:

Procedure

---

1. Configure an ethernet ring in L2VPN configuration mode.

   Example

   ```
   Router# configure
   Router(config)# l2vpn
   Router(config-l2vpn)# ethernet ring g8032 r1
   ```

   Enters L2VPN configuration mode, enables G.8032 ring mode, and enters G.8032 configuration submode.

2. For each ring port you wish to protect:
   a) Enable G.8032 ERPS for the specified port.

      Example

      ```
      Router(config-l2vpn-erp)# port0 interface GigabitEthernet 0/0/0/0
      ```

b) Specify a port to monitor the G.8032 ERPS and detect ring link failure.

Example

```
Router(config-l2vpn-erp-port0)# monitor port0 interface 0/0/0/0.5
```

c) Exit from port configuration submode.

Example

```
Router(config-l2vpn-erp-port0)# exit
```

d) Repeat for additional ports as needed.

For example,

Example

```
Router(config-l2vpn-erp)# port1 interface GigabitEthernet 0/0/0/1
Router(config-l2vpn-erp-port1)# monitor port1 interface 0/0/0/1.5
Router(config-l2vpn-erp-port1)# exit
```

3. Configure a set of VLAN IDs that should not be protected by ERPS.

Example

```
Router(config-l2vpn-erp)# exclusion-list vlan-ids a-d
```

4. Configure the ethernet ring G.8032 as an open ring.

Example

```
Router(config-l2vpn-erp)# open-ring
```

ERPS parameters are configured successfully. Verification output should show the ring instance and monitored ports in the desired state.

Configure TCN propagation

Configure topology change notification propagation for the ERPS instance.

Enable topology change notification (TCN) propagation for an ERPS instance to ensure network topology changes are propagated correctly between network rings.

TCN propagation ensures that changes detected within one ring are communicated to associated major rings or protocols, improving convergence and loop prevention in complex ring-based networks.

Before you begin

- Confirm that the ERPS instance is configured.
- Ensure the relevant bridge domain is configured.

Follow these steps to configure TCN propagation for an ERPS instance:

Procedure

Enable TCN propagation in L2VPN configuration mode.

Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# tcn-propagation
Router(config-l2vpn)# commit
```

TCN propagation is enabled for the ERPS instance. Verification commands display the updated state reflecting the configuration.

Configure CFM MEP

Configure Connectivity Fault Management (CFM) maintenance endpoints (MEPs) for G.8032 Ethernet ring monitoring and rapid fault detection.

Enable continuous monitoring and rapid rerouting of traffic in Ethernet rings using CFM MEPs in networks running G.8032 (ERPS).

CFM MEPs and G.8032 Ethernet Ring Protection Switching work together to monitor the Ethernet ring's health. If a ring link fails, Ethernet Fault Detection (EFD) automatically shuts down the affected port and reroutes traffic through an alternate path, maintaining network resilience.

Before you begin

Ensure all CFM domains, services, and ring interfaces are defined and planned.

Follow these steps to configure CFM MEP for G.8032 monitoring:

Procedure

1. Configure a CFM domain and service.

   Example

   ```
   Router# configure
   Router(config)# ethernet cfm
   Router(config-cfm)# domain dom23to24 level 6
   Router(config-cfm-domain)# service ser23to24 down-meps
   ```

2. Configure continuity checks to enable fault detection.

   Example

   ```
   Router(config-cfm-svc)# continuity-check interval 10s
   ```

3. Configure a MEP crosscheck on the main interface to detect failures and reroute traffic.

Example

```
Router(config-cfm-svc)# mep crosscheck
Router(config-cfm-svc-xcheck)# mep-id 3
Router(config-cfm-svc-xcheck)# exit
```

4. Configure Ethernet Fault Detection (EFD) for failure detection and rerouting.

Example

```
Router(config-cfm-svc)# efd
```

5. Configure CFM MEP on the sub-interface.

Example

```
Router# configure terminal
Router(config)# interface Gigabiteethernet0/0/0/0.5
Router(config-if)# ethernet cfm
Router(config-if-cfm)# mep domain dom23to24 service ser23to24 mep-id 4
```

The CFM MEP configuration enables continuous monitoring of the Ethernet ring. If a link failure occurs, the router shuts down the affected port and automatically reroutes traffic, ensuring service continuity. You can verify the configuration and operational state using verification commands relevant to your platform.

G.8032 Ethernet ring protection switching example

Provides source configuration details for G.8032 Ethernet Ring Protection Switching example.

This sample configuration illustrates the elements that a complete G.8032 configuration includes:

```
# Configure the ERP profile characteristics if ERPS instance behaviors are
non-default.
ethernet ring g8032 profile ERP-profile
  timer wtr 60
  timer guard 100
  timer hold-off 1
  non-revertive


# Configure the ERPS instance under L2VPN
l2vpn
  ethernet ring g8032 RingA
    port0 interface g0/0/0/0
    port1 interface g0/1/0/0
    instance 1
      description BD2-ring
      profile ERP-profile
      rpl port0 owner
      vlan-ids 10-100
      aps channel
        level 3
        port0 interface g0/0/0/0.1
        port1 interface g1/1/0/0.1

# Set up the bridge domains
bridge group ABC
```

```
    bridge-domain BD2
      interface Gig 0/0/0/0.2
      interface Gig 0/1/0/0.2
      interface Gig 0/2/0/0.2

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

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

interface Gig 1/1/0/0.1 l2transport
  encapsulation dot1q 5

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

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

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

G.8032 interconnection node example

Describes the configuration process and key components involved in setting up a G.8032 interconnection node to support open ring network topologies.

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

Summary

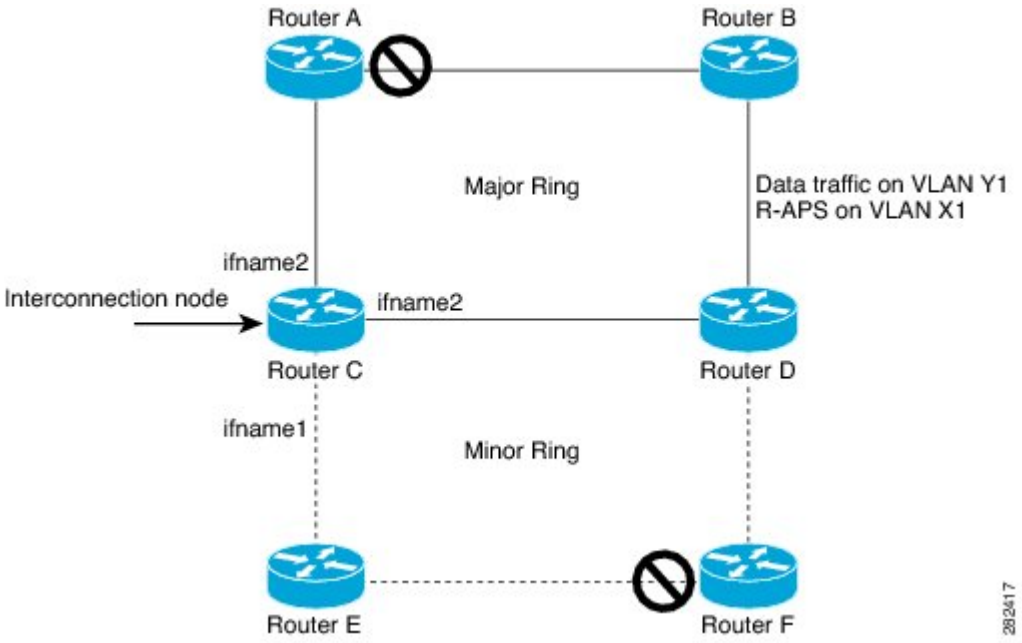
The key components involved in the process are:

- Interconnection node (Router C in this scenario): Serves as the connecting point between different segments of the Ethernet ring.
- Ethernet ring controller: Manages the G.8032 ring state and triggers protection switching when failures are detected.
- Network interfaces: Carry VLAN-tagged traffic segments and are configured with specific encapsulation settings to support the ring topology.

G.8032 interconnection nodes support network resiliency in Ethernet ring topologies by enabling open ring architectures, which help in rapid protection switching and fault recovery.

Workflow

Figure 24: Open ring scenario - interconnection node



The process involves these stages:

1. Prepare network interfaces: Identify and configure physical and subinterfaces on the interconnection node required for the ring.
2. Assign encapsulation to interfaces: Apply appropriate VLAN tags (e.g., dot1q X1, Y1) to distinguish different traffic segments across interfaces.
3. Define the Ethernet ring instance: Create the G.8032 Ethernet ring with required parameters, including ring name, interface assignments, and open-ring settings.
4. Set instance parameters: Include the VLAN ID inclusion list, enable the rapid protection switching channel (APS), and establish bridge domains for both APS control and data traffic.
5. Apply bridging configuration: Map the relevant interfaces into bridge domains associated with the APS channel and regular traffic domains.

Result

The interconnection node is active within an open G.8032 Ethernet ring, supporting fast protection switching and loop-free operation in the event of link or node failures.

The configuration steps referenced in the process correspond to this example:

```
interface <ifname1.1> l2transport
 encapsulation dot1q X1
interface <ifname1.10> l2transport
 encapsulation dot1q Y1
interface <ifname2.10> l2transport
 encapsulation dot1q Y1
interface <ifname3.10> l2transport
 encapsulation dot1q Y1
l2vpn
ethernet ring g8032 <ring-name>
      port0 interface <main port ifname1>
      port1 interface none #? This router is connected to an interconnection
node
      open-ring #? Mandatory when a router is part of an open-ring
      instance <1-2>
         inclusion-list vlan-ids  X1-Y1
         aps-channel
           Port0 interface <ifname1.1>
           Port1 none #? This router is connected to an interconnection node
```

```
 bridge group bg1
    bridge-domain bd-aps#? APS-channel has its own bridge domain
       <ifname1.1> #? There is only one APS-channel at the interconnection
node
    bridge-domain bd-traffic #? Data traffic has its own bridge domain
       <ifname1.10>
       <ifname2.10>
       <ifname3.10>
```

G.8032 node of an open ring example

Describes the process for configuring a G.8032 node as part of an open ring topology, outlining the key components and step-by-step configuration stages to ensure proper ring operation.

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

Summary

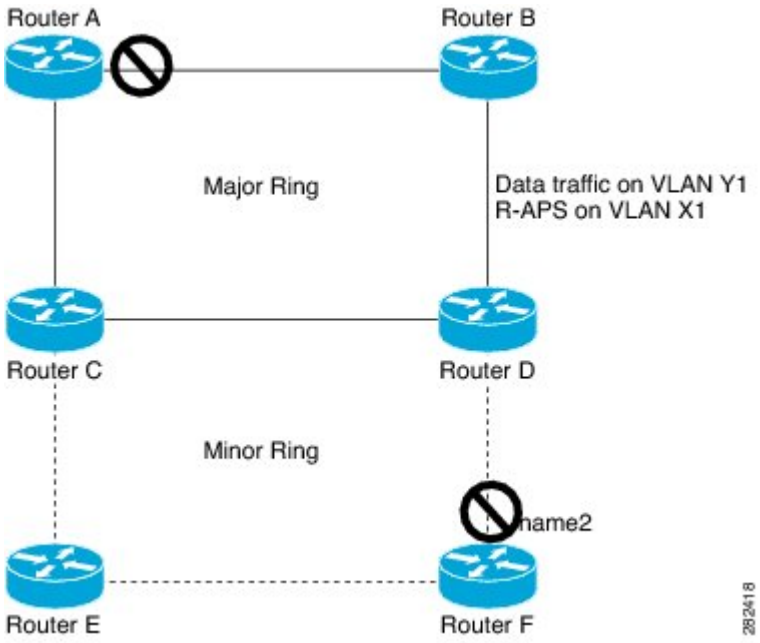
The key components involved in the process are:

- G.8032 node (router F): Acts as the part of the open ring, requiring specific configuration for participation in ERPS.
- Ethernet interfaces: Physical and logical interfaces used for ring and traffic, each assigned unique VLAN identifiers.
- Ring logic and protection features: Settings such as open ring mode, inclusion lists, RPL ownership, APS-channel, and bridge groups, which facilitate redundancy and protection.

Configuring a G.8032 node in an open ring topology enables redundancy and resiliency by implementing Ethernet Ring Protection Switching (ERPS). This process involves multiple interfaces, VLAN configurations, and logical ring parameters to ensure continuous service and rapid recovery from failures.

Workflow

Figure 25: Open ring scenario



The process involves these stages:

1. Interface configuration: Configure the relevant physical and logical interfaces required for ring traffic and APS.

2. Ethernet ring G.8032 setup: Enable open ring mode (mandatory for open ring topology), set the ring instance, inclusion list for VLANs, and assign the RPL owner.
3. APS-channel configuration: Define APS channel ports for monitoring and switching.
4. Bridge group and domain setup: Configure bridge groups with bridge-domain for APS channel interfaces and bridge-domain for regular ring data traffic interfaces.
5. Verification: Validate that the configuration matches the network topology for correct open ring and protection switching operation.

Result

Upon completing these configuration stages, the G.8032 node functions as an effective part of the open ring topology, enabling ERPS-based protection, redundancy, and seamless data traffic continuity in the network.

The configuration steps referenced in the process correspond to this example:

```
interface <ifname1.1> l2transport
 encapsulation dot1q X1
interface <ifname2.1> l2transport
 encapsulation dot1q X1
interface <ifname1.10> l2transport
 encapsulation dot1q Y1
interface <ifname2.10> l2transport
 encapsulation dot1q Y1
l2vpn
   ethernet ring g8032 <ring-name>
      port0 interface <main port ifname1>
      port1 interface <main port ifname2>
      open-ring #? Mandatory when a router is part of an open-ring
      instance <1-2>
         inclusion-list vlan-ids  X1-Y1
      rpl port1 owner #? This node is RPL owner and <main port ifname2> is
blocked
         aps-channel
           port0 interface <ifname1.1>
           port1 interface <ifname2.1>
 bridge group bg1
    bridge-domain bd-aps#? APS-channel has its own bridge domain
       <ifname1.1>
       <ifname2.1>
    bridge-domain bd-traffic #? Data traffic has its own bridge domain
       <ifname1.10>
       <ifname2.10>
```

VPLS preferred path over SR-TE policy

Outlines VPLS pseudowire path selection using SR-TE policies, integration with VPLS services, operational workflows, and provides configuration instructions for deploying VPLS preferred paths using Segment Routing Traffic Engineering.

VPLS preferred path over SR-TE policy is a traffic engineering feature that

- steers VPLS pseudowire traffic through explicit SR-TE policies
- overrides default IGP shortest-path behavior when a preferred path is configured, and
- uses LDP for signaling while SR-TE determines forwarding.

Feature history

The feature history table lists release support for this feature.

Table 40: Feature History Table

Feature Name	Release Information	Feature Description
VPLS preferred path over SR-TE policy	Release 25.4.1	Introduced in this release on: Fixed Systems (8100 [ASIC: Q200], 8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q200, P100]) You can now steer VPLS pseudowire traffic over a specific SR-TE policy, ensuring precise control over routing, bandwidth, or latency for Layer 2 VPN services. This feature allows you to bind a VPLS pseudowire to an SR-TE tunnel interface, overriding the default IGP shortest path. By doing so, you can enforce traffic engineering constraints and optimize network performance. This feature is only supported with static neighbors and is not supported when neighbor discovery is used.

SR-TE preferred-path behavior

- VPLS preferred paths over SR-TE policies steer VPLS pseudowire traffic over specific SR-TE policies.
- This behavior provides precise control over routing, bandwidth, or latency for Layer 2 VPN services.
- The feature binds a VPLS pseudowire to an SR-TE tunnel interface and overrides the default IGP shortest path.

SR-TE policy pseudowire path options

Provides information about ways SR-TE policy overrides default VPLS pseudowire path selection to achieve explicit control, resource guarantees, and predictable service traffic steering.

VPLS pseudowires traditionally select paths based on the shortest IGP route between Provider Edge (PE) routers. When you associate a VPLS pseudowire with a Segment Routing Traffic Engineering (SR-TE) policy, the default path selection is overridden. This association directs the pseudowire to use the SR-TE tunnel as its forwarding path, offering several traffic engineering advantages:

- Explicit path steering: Enables the use of a defined segment list for precise control of the traffic path.
- Resource reservation: Supports bandwidth or latency guarantees as specified by the SR-TE policy.
- Predictable and stable routing: Ensures the pseudowire uses the chosen path regardless of IGP recalculations, giving consistent service delivery.

By provisioning an SR-TE policy with a specific segment list, you can route VPLS service traffic through a preferred set of routers. Binding the VPLS pseudowire to this policy keeps service traffic on paths with guaranteed latency and bandwidth, independent of IGP changes or default routing decisions.

VPLS and SR-TE preferred path attributes

Provides key facts and details about how VPLS uses LDP for virtual LAN services and how SR-TE preferred path enables explicit routing across the network for service traffic.

VPLS and SR-TE preferred paths are network features that establish connectivity and optimize traffic steering in service provider networks. The following reference highlights their operational characteristics:

- Virtual Private LAN Service (VPLS)
  - Uses the Label Distribution Protocol (LDP) control plane to establish a full mesh of pseudowires (PWs) between Provider Edge (PE) routers.
  - Creates a virtual bridge that emulates LAN service, allowing transparent connection across multiple customer sites.
- SR-TE preferred path
  - Defines explicit, source-routed paths across the network using Segment Routing Traffic Engineering (SR-TE) policies.
  - SR-TE policies use a list of segment IDs (SIDs) to steer traffic.
  - The preferred-path feature enables existing services to direct traffic onto a particular SR-TE policy tunnel, overriding the network's default shortest path selection (through IGP or LDP).

How VPLS services use SR-TE policies for preferred path selection

Describes how VPLS, LDP signaling, SR-TE policy, and pseudowire forwarding work together for preferred path selection.

Describes how VPLS, LDP signaling, SR-TE policy, and pseudowire forwarding work together to ensure VPLS traffic follows an explicitly defined, preferred path across the MPLS core rather than the default IGP path.

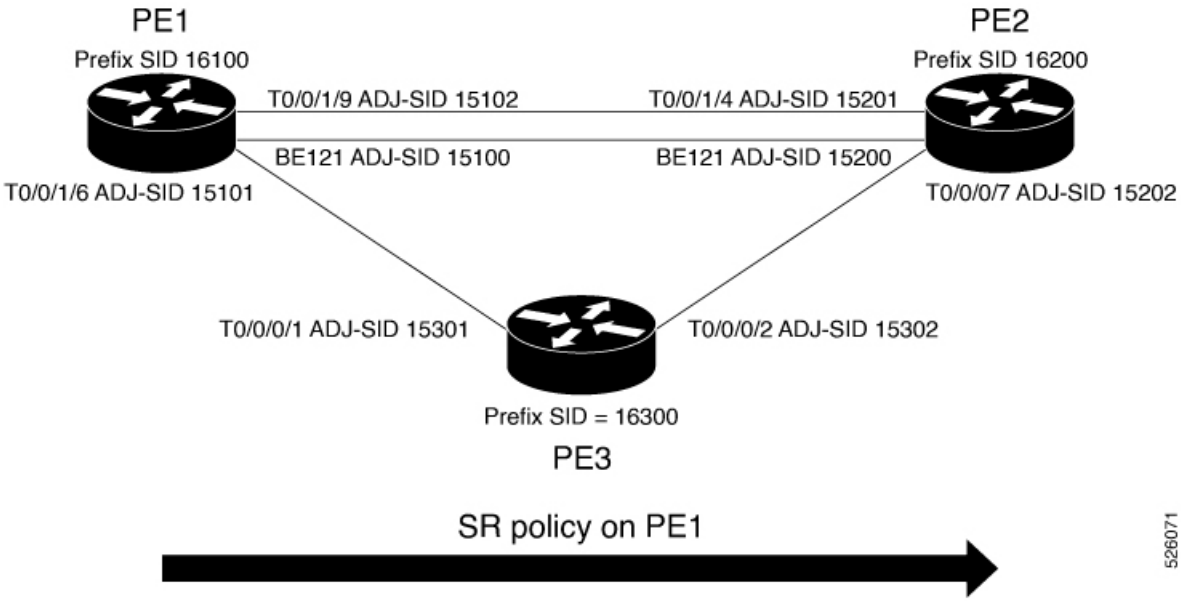
Summary

The key components involved in the process are:

- VPLS service: Signals pseudowires using LDP, negotiating parameters such as VC IDs, status, and MTU between PE routers.
- PE routers: Configure the VPLS service and steer pseudowire traffic using a specified SR-TE policy as the transport tunnel.
- SR-TE policy: Defines the explicit, preferred path through the MPLS core, with the destination set to the loopback address of the remote PE.
- MPLS label stack: Encapsulates VPLS Ethernet frames, where the outer labels correspond to the SR-TE path and the inner label is the VPLS service label negotiated by LDP.

This process enables service providers to control the forwarding path of L2VPN VPLS traffic within an MPLS core by leveraging SR-TE policies. This ensures traffic takes an explicitly defined path, overriding the default IGP shortest path routing.

Workflow



These stages describe how the VPLS preferred path over SR-TE policy works.

1. Control plane signaling: The VPLS service on the PE routers uses LDP to signal and establish pseudowires, negotiating necessary parameters such as VC IDs and MTU.
2. SR-TE policy configuration: The ingress PE router is configured with an SR-TE policy specifying the preferred explicit path through the core network, targeting the remote PE's loopback address.
3. Data plane steering: The ingress PE router directs pseudowire traffic into the SR-TE tunnel, ensuring packets follow the explicitly defined path provided by the policy.
4. Encapsulation: VPLS Ethernet frames are encapsulated using an MPLS label stack. The inner label identifies the VPLS service (negotiated by LDP) and the outer labels correspond to the SR-TE path.
5. Traffic forwarding: The ingress PE forwards customer packets through the VPLS instance, applies the service label, pushes the segment routing label stack, and the MPLS core forwards packets based on these labels.
6. Decapsulation and delivery: The egress PE router removes the segment routing labels, processes the VPLS service label, and delivers the original Ethernet frame to the customer-facing interface.

Result

VPLS traffic reliably follows the preferred SR-TE policy-defined path across the MPLS core, while the VPLS pseudowire remains established by LDP, ensuring predictable and controlled service delivery.

Configure VPLS preferred path over SR-TE policy

Configure VPLS traffic to use SR-TE policies as preferred pseudowire paths in an IS-IS routed network.

Ensure VPLS traffic uses Segment Routing Traffic Engineering (SR-TE) policies as preferred paths for pseudowires.

This task applies when you want to control the forwarding path of VPLS services by leveraging explicit SR-TE policies using prefix and adjacency SIDs. The procedure is intended for networks running IS-IS routing across PE routers.

Before you begin

Ensure IS-IS routing is operational on all PE routers and that you have administrative access to configure the routers.

Follow these steps to configure VPLS preferred path over SR-TE policy:

Procedure

1. Configure prefix-SID on PE1, PE2, and PE3.
   a) Configure prefix-SID on PE1.

      Example

```
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0031.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16100
Route(config-isis-af)# commit
```

   b) Configure prefix-SID on PE2.

      Example

```
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0021.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16200
Route(config-isis-af)# commit
```

   c) Configure prefix-SID on PE3.

      Example

```
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.3030.0030.0035.00
```

```
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16300
Route(config-isis-af)# commit
```

2. Configure adjacency-SID on IS-IS interfaces on PE1, PE2, and PE3.

a) Configure adjacency-SID on IS-IS interfaces on PE1.

Example

```
Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15100
Route(config-isis-if-af)# exit
Router# configure
Route(config)# router isis core
Route(config-isis)# interface HundredGigE 0/0/0/24
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15101
Route(config-isis-if-af)# exit
Router# configure
Route(config)# router isis core
Route(config-isis)# interface HundredGigE 0/0/0/23
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15102
Route(config-isis-if-af)# commit
```

b) Configure adjacency-SID on IS-IS interfaces on PE2.

Example

```
Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15200
Route(config-isis-if-af)# exit
```

```
Router# configure
Route(config)# router isis core
Route(config-isis)# interface HundredGigE 0/0/0/22
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15201
Route(config-isis-if-af)# exit
Router# configure
Route(config)# router isis core
Route(config-isis)# interface HundredGigE 0/0/0/21
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15202
Route(config-isis-if-af)# commit
```

c) Configure adjacency-SID on IS-IS interfaces on PE3.

Example

```
Router# configure
Route(config)# router isis core
Route(config-isis)# interface HundredGigE 0/0/0/20
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15301
Route(config-isis-if-af)# exit
Router# configure
Route(config)# router isis core
Route(config-isis)# interface HundredGigE 0/0/0/19
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15302
Route(config-isis-if-af)# commit
```

3. Configure segment-list on PE1, PE2, and PE3.

a) Configure segment-list on PE1.

Example

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE1-PE2
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment-list name PE1-PE3
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# exit
```

```
Router(config-sr-te)# segment-list segment-list name PE1-PE2-PE3
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# index 2 mpls label 16300
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# index 2 mpls label 16200
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment-list name PE1-PE2_BE121
Router(config-sr-te-sl)# index 1 mpls label 15100
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2_link
Router(config-sr-te-sl)# index 1 mpls label 15101
Router(config-sr-te-sl)# index 2 mpls label 15302
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2-t0016
Router(config-sr-te-sl)# index 1 mpls label 15101
Router(config-sr-te-sl)# index 2 mpls label 16200
Router(config-sr-te-sl)# commit
```

b) Configure segment-list on PE2.

Example

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE2-PE1
Router(config-sr-te-sl)# index 1 mpls label 16100
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment-list name PE2-PE3-PE1
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# index 2 mpls label 16100
Router(config-sr-te-sl)# commit
```

c) Configure segment-list on PE3.

Example

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE3-PE1
Router(config-sr-te-sl)# index 1 mpls label 16100
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment-list name PE3-PE2-PE1
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# index 2 mpls label 16100
Router(config-sr-te-sl)# commit
```

4. Configure SR-TE policies with candidate paths and preferences.

Example
On PE1

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 100
Router(config-sr-te-policy)# color 1 end-point ipv4 192.0.2.2
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 400
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy)# preference 500 <-----------------largest number
 takes the precedence
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 1013
Router(config-sr-te-policy)# color 1013 end-point ipv4 192.0.2.2
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 100
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2_BE121
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy)# preference 200
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2-t0016
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy)# preference 500
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy)# preference 600
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy)# preference 700
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2_link
Router(config-sr-te-pp-info)# commit
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 1300
Router(config-sr-te-policy)# color 1300 end-point ipv4 192.0.2.10
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 100
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3
Router(config-sr-te-pp-info)# commit
```

The highest preference number determines the active path.

5. Configure VPLS to use SR-TE policies as preferred path.
   a) Attach the auto-generated SR-TE policy name to the L2VPN PW class.

      Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name policy_name** command to display the auto-generated policy name.

Example

```
Router# show segment-routing traffic-eng policy candidate-path name 100

SR-TE policy database
---------------------
Color: 1, End-point: 192.0.2.2
 Name: srte_c_1_ep_2.2.2.2

Router# show segment-routing traffic-eng policy candidate-path name 1013

SR-TE policy database
---------------------
Color: 1013, End-point: 192.0.2.2
 Name: srte_c_1013_ep_2.2.2.2

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pw100
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy
srte_c_1_ep_2.2.2.2
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn)# pw-class pw1013
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy
srte_c_1013_ep_2.2.2.2 fallback disable
```

b) Configure fallback disable.

By default, fallback is enabled. If the SR-policy is down, then L2VPN VPLS will try to be UP using the regular IGP path, and not using the SR policy. If Fallback Disable is configured, the L2VPN PW will be down when the SR-policy is down. Preferred-path is the action of pinning down a PW to a SR TE policy.

Example

```
Router(config-l2vpn)# pw-class pw1013
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy
srte_c_1013_ep_2.2.2.2 fallback disable
```

c) Configure VPLS bridge groups and domains.

Example

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain vpls501
Router(config-l2vpn-bg-bd)# interface Bundle-Ether41.501
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/24.1
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# vfi vpls1
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.2 pw-id 501
Router(config-l2vpn-bg-bd-vfi-pw)# pw-class pw100
Router(config-l2vpn-bg-bd-vfi-pw)# exit
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.10 pw-id 501
Router(config-l2vpn-bg-bd-vfi-pw)# commit
```

6. Verify the VPLS preferred path over SR-TE policy configuration.

Example

```
PE1# show segment-routing traffic-eng ipv4 topology | inc Prefix
  Prefix SID:
    Prefix 192.0.2.11, label 16100 (regular)
  Prefix SID:
    Prefix 192.0.2.10, label 16300 (regular)
  Prefix SID:
    Prefix 192.0.2.2, label 16200 (regular)
```

Example

```
PE1# show segment-routing traffic-eng ipv4 topology | inc Adj SID
Adj SID: 61025 (unprotected) 15102 (unprotected)
    Adj SID: 61023 (unprotected) 15101 (unprotected)
    Adj SID: 65051 (unprotected) 15100 (unprotected)
    Adj SID: 41516 (unprotected) 15301 (unprotected)
    Adj SID: 41519 (unprotected) 15302 (unprotected)
    Adj SID: 46660 (unprotected) 15201 (unprotected)
    Adj SID: 24003 (unprotected) 15202 (unprotected)
    Adj SID: 46675 (unprotected) 15200 (unprotected)
```

Example

```
PE1# show segment-routing traffic-eng policy candidate-path name 100

SR-TE policy database
---------------------

Color: 100, End-point: 192.0.2.2
 Name: srte_c_1_ep_2.2.2.2
```

Example

```
PE1# show segment-routing traffic-eng policy name 100
SR-TE policy database
---------------------

Name: 100 (Color: 1, End-point: 192.0.2.2)
  Status:
    Admin: up  Operational: up for 05:44:25 (since Feb  1 17:32:34.434)
  Candidate-paths:
    Preference 500:
      Explicit: segment-list PE1-PE2 (active)
        Weight: 0, Metric Type: IGP
          16200 [Prefix-SID, 192.0.2.2]
    Preference 400:
      Explicit: segment-list PE1-PE3-PE2 (inactive)
      Inactive Reason: unresolved first label
        Weight: 0, Metric Type: IGP
  Attributes:
    Binding SID: 27498
      Allocation mode: dynamic
      State: Programmed
      Policy selected: yes
    Forward Class: 0
```

Example

```
PE1# show segment-routing traffic-eng forwarding policy name 100

Policy        Segment         Outgoing    Outgoing            Next Hop
  Bytes
Name          List            Label       Interface
  Switched
------------- --------------- ----------- ------------------- ---------------
 ------------
100           PE1-PE2         Pop         HundredGigE 0/0/0/23 192.0.2.12
    0
                              Pop          BE121               192.0.2.13
   0
```

The prefix-SID and adjacency-SID must be in the SR topology.

The VPLS service uses SR-TE policies as preferred pseudowire paths. The configuration completes successfully when verification commands show the intended state.

11 Integrated Routing and Bridging

Topics:

- Integrated routing and bridging

Explains integrated routing and bridging concepts, EVPN IRB architecture, packet forwarding mechanisms, planning guidelines, configuration procedures, and advanced distributed anycast gateway features for robust Layer 2/Layer 3 network deployment.

Integrated routing and bridging

Introduces integrated routing and bridging, covering Bridge-Group Virtual Interface in IRB, prerequisites and planning, packet forwarding behavior, IRB configuration steps, and EVPN IRB architecture with detailed components, operational environments, and benefits.

Integrated routing and bridging (IRB) is a network capability that

- exchanges traffic between bridging services and a routed interface by using a bridge-group virtual interface (BVI)
- provides Layer 2 bridging service for hosts in a Layer 2 domain, and
- provides routing service for hosts that are in different subnets within a Layer 3 VPN.

A bridge-group virtual interface (BVI) is a virtual interface that

- acts as the Layer 3 gateway for hosts within a bridge domain
- connects the bridging and routing domains on the device, and
- is configured with an IP address in the same subnet as the hosts it serves.

Bridge-group virtual interfaces

Explains how Bridge-Group Virtual Interfaces connect bridge domains to routing services in an IRB deployment.

A Bridge-Group Virtual Interface (BVI) is a virtual interface that

- acts as a normal routed interface and represents the link between the bridging and routing domains on a router
- uses a MAC address from the local chassis MAC address pool unless overridden, and
- uses an IPv4 or IPv6 address in the same subnet as the hosts on the bridged domain.

The BVI identifier is independent of the bridge domain identifier.

BVI interfaces support an identifier range from 1 to 4294967295. Although a BVI does not support bridging itself, it acts as the gateway for the corresponding bridge domain to a routed interface within the router.

BVI interface and line protocol states

The BVI interface is up when the BVI exists and the configured bridge domain has at least one available active bridge port.

- The bridge domain must be in the up state.
- The BVI IP address must not conflict with another IP address on an active interface, and
- The interface remains up if at least one bridge port is up, even when all Ethernet Flow Points (EFPs) are down.
- If all bridge ports associated with the bridge domain for that BVI are down, the BVI moves to the down state.

If you configure a BVI for a bridge domain, the BVI serves as the gateway for hosts on that domain, distributing IP addresses and routing traffic between the bridged and routed domains.

Best practice for IRB configuration

Outlines the best practices for planning IP addressing, MAC usage, and routing advertisement before you configure Integrated Routing and Bridging (IRB).

Follow these best practices before you configure IRB on a BVI:

- Know the IP addressing and all Layer 3 information you plan to configure on the BVI before you set up IRB.

- Plan MAC addressing if you intend to override the preferred MAC address for the BVI interface.

  You can replace the preferred MAC address for the BVI interface with the default MAC address allocated from the chassis pool. The MAC address is divided into:

  - 32 bits most significant bits called MAC prefix.

    The router has a limitation of four different MAC prefixes per system. You must not use more than four different MAC prefixes when choosing the MAC address for BVI and other L3 interfaces.

- 16 bits least significant called MAC host. You can choose any value for the MAC host.
- Advertise the BVI network by using static or dynamic routing on the BVI interface.

How packet forwarding works in IRB

Describes how IRB bridges traffic within a subnet and routes traffic between subnets by using the BVI.

When IRB is configured on a router, ARP requests are resolved between the hosts and the BVI as part of the bridge domain. This allows for seamless communication between hosts that are either within the same subnet or across routed networks.

Summary

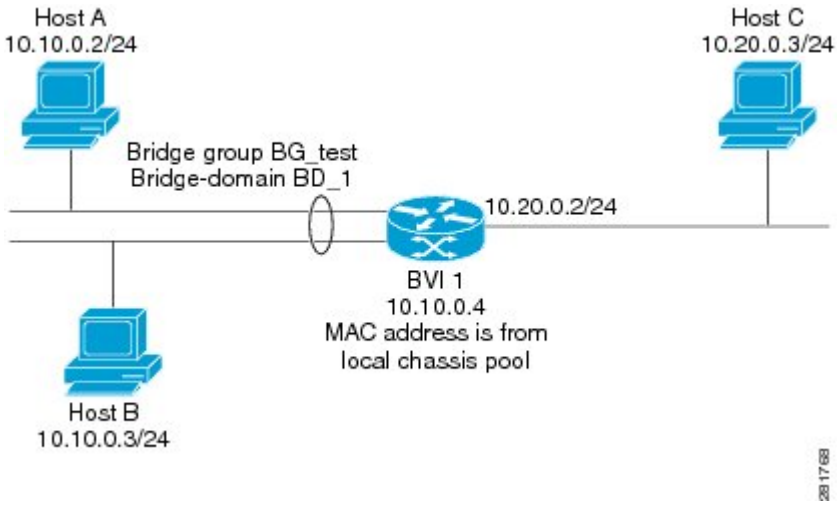
The key components involved in the process are:

- Host A: Sends traffic from the bridge domain, either to another host on the same subnet or to a host on a routed network.
- Host B: Receives bridged traffic within the same bridge domain.
- Host C: Communicates through a routed network connected to the router.
- BVI: Acts as the logical interface between the bridging domain and the routing domain.
- Bridging engine: Forwards frames within the bridge domain based on MAC address information.
- Routing engine: Routes packets between different IP networks.
- Bridged interfaces: Carry traffic for hosts that belong to the same bridge domain.
- Routed interface: Carries traffic between the router and external routed networks.
- ARP: Resolves addresses between hosts and the BVI within the bridge domain.
- Bridging table: Identifies the correct bridged interface for forwarding frames inside the bridge domain.

IRB enables a router to bridge traffic within a subnet while routing traffic between different subnets, using a logical interface known as the BVI. These flows describe how IRB handles local bridging and routed forwarding.

Workflow

Figure 26: IRB packet flows between hosts



These stages describe the packet-forwarding behavior:

1. Initial address resolution: ARP resolves address information between the hosts and the BVI within the bridge domain.
2. Bridged interface packet evaluation: A host on a bridged interface sends a packet. If the destination MAC address matches the BVI MAC address, the packet goes to the BVI. If it does not match, the bridging engine bridges the packet.
3. Bridging within the same subnet: When Host A sends data to Host B on the 10.10.0.0 network, the router does not route the traffic. The bridging engine forwards the packet between the appropriate bridged segment interfaces because both hosts are on the same subnet.
4. Sending traffic from the bridge domain to a routed network: When Host A sends data to Host C, Host A sends the packet to the BVI after ARP resolves the BVI address. The packet uses Host A as the source and the BVI as the destination at the MAC level.
5. Forwarding traffic from the BVI to the routing domain: Because Host C is on another network, the BVI forwards the packet to the routing engine. The router prepares the packet for delivery through the routed interface toward Host C.
6. Routing traffic to Host C: The routed interface at 10.20.0.2 receives the packet from the BVI side and routes it to Host C at 10.20.0.3.
7. Receiving traffic from a routed network: When Host C sends data to Host B, the packet enters the router through the routed interface. The packet includes Host C as the source, the ingress interface MAC address as the destination MAC, and Host B as the destination IP.
8. Routing decision toward the BVI: The routed interface checks the routing table and determines that the packet must go to the BVI at 10.10.0.4.
9. Forwarding traffic from the routing domain to the bridge domain: The routing engine captures the packet destined for the BVI and forwards it to the corresponding bridge domain.
10. Final bridging to Host B: The bridging engine checks the bridging table. If Host B's MAC address appears in the table, the router forwards the packet through the correct bridged interface. If the MAC address does not appear, the router floods the packet on all interfaces in the bridge group.

Result

The IRB process allows the router to bridge local traffic within a subnet and route traffic between different networks through the BVI. This process enables communication between hosts in the bridge domain and hosts on routed networks while preserving the correct forwarding behavior for each traffic type.

Configure IRB

Configure a BVI, assign Layer 2 attachment circuits to a bridge domain, and associate the BVI with that bridge domain.

Set up IRB to enable routing and bridging for local and tagged bridge ports.

IRB enables seamless integration of Layer 2 and Layer 3 functionality using a BVI, allowing traffic to be bridged within a domain and routed as needed.

Before you begin

Plan the BVI IP addressing, MAC addressing, and bridge-domain membership before you begin.

Procedure

1. Configure a BVI and assign IP addresses.

   Example

   ```
   Router# configure
   Router(config)# interface bvi 1
   Router(config-if)# ipv4 address 10.10.0.4 255.255.255.0
   Router(config-if)# ipv6 address 2001:100:1:1::1/96
   ```

   Optionally, configure a static MAC address on the BVI interface.

   ```
   Router(config-if)# mac-address 2001.100.2
   Router(config-if)# exit
   ```

2. Configure the Layer 2 AC interface and add it to the bridge domain.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/0/1 l2transport
   Router(config-if-l2)# exit
   Router(config)# l2vpn
   Router(config-l2vpn)# bridge group 10
   Router(config-l2vpn-bg)# bridge-domain 1
   Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/1
   Router(config-l2vpn-bg-bd-ac)# exit
   ```

3. Associate the BVI with the bridge domain.

   Example

   ```
   Router(config-l2vpn-bg-bd)# routed interface bvi 1
   Router(config-l2vpn-bg-bd-bvi)# commit
   ```

4. Configure IRB for tagged bridge ports using sub-interfaces within a bridge domain.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/2.1 l2transport
   Router(config-subif)# encapsulation dot1q 102
   Router(config-subif)# rewrite ingress tag pop 1 symmetric
   Router(config-subif)# exit
   Router(config)# interface bvi 2
   Router(config-if)# ipv4 address 56.78.100.1 255.255.255.0
   Router(config-if)# ipv6 address 56:78:100::1/64
   Router(config-if)# mac-address 2002.100.1
   ```

```
Router(config-if)# exit
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 10
Router(config-l2vpn-bg)# bridge-domain 2
Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/2.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# routed interface bvi 2
Router(config-l2vpn-bg-bd-bvi)# commit
```

Note

Double VLAN tagged sub-interface is not supported for IRB service.

5. Use the **show interfaces bvi 1 brief** command to verify the interface state and packet counters for the BVI.

Example

```
Router# show interfaces bvi 1 brief
BVI1 is up, line protocol is up
Interface state transitions: 701
Hardware is Bridge-Group Virtual Interface, address is 2001.0100.0001
Internet address is 10.10.0.4/24
MTU 1514 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
reliability 255/255, txload 0/255, rxload 1/255
Encapsulation ARPA, loopback not set,
Last link flapped 2d06h
ARP type ARPA, ARP timeout 04:00:00
Last input 00:00:00, output 00:00:13
Last clearing of "show interface" counters 3d18h
30 second input rate 43721000 bits/sec, 49684 packets/sec
30 second output rate 0 bits/sec, 0 packets/sec
15428019162 packets input, 1697081244790 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 0 broadcast packets, 0 multicast packets
6084259298 packets output, 669870073726 bytes, 0 total output drops
Output 0 broadcast packets, 0 multicast packets
```

The router bridges local traffic within the bridge domain and routes traffic through the BVI interface.

EVPN IRB

Details EVPN IRB functionality, including components, environments, route types, benefits, distributed anycast gateway features, multihoming modes, configuration procedures, and software MAC learning mechanisms, providing comprehensive guidance for EVPN IRB deployment.

EVPN IRB is a network technology that

- enables Layer 3 forwarding among hosts across different IP subnets
- preserves the multihoming capabilities of EVPN, and
- allows EVPN hosts or subnets to communicate with IP VPNs.

Starting from Release 26.1.1, you can configure EVPN IRB over a Segment Routing over IPv6 (SRv6) core.

Feature History Table
Table 41: Feature History Table

Feature Name	Release Information	Feature Description
EVPN IRB over SRv6 core	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
EVPN IRB over SRv6 core	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100]); Centralized Systems (8400 [ASIC: K100]); Modular Systems (8800 [LC ASIC: P100]) EVPN IRB enhances network flexibility by enabling seamless Layer 3 connectivity between hosts on different subnets over an SRv6 network. This feature allows Layer 3 forwarding among hosts across IP subnets, maintains EVPN multi-homing capabilities, and facilitates communication between EVPN hosts or subnets and IP VPNs. Leveraging SRv6 programmable and flexible transport, this solution streamlines the integration and management of modern, diverse network environments.

EVPN IRB components

Lists the key control-plane and forwarding components used by EVPN IRB to advertise, resolve, and forward Layer 2 and Layer 3 reachability.

These components support EVPN IRB traffic management and routing.

- BGP advertises subnet and host routes to the EVPN core by using route type 5 and route type 2 messages.
- EVPN manages Ethernet segment configuration, host route advertisement, and failover scenarios.
- L2RIB handles MAC or IP mobility, route resolution, and best-route computation.
- BVI manager manages IRB interfaces and advertises BVI subnet and MAC addresses.
- L2FIB forwards traffic by using MAC and IP information.

EVPN IRB environments

Lists the supported EVPN IRB deployment options and their distinguishing features.

- Single-homing interface: Customer edge (CE) devices connect directly to a single physical edge (PE) router.
- Multihoming interface: A CE device connects to multiple PE routers through dual links, a Link Aggregation Group (LAG), or a switch.
- Anycast gateway or BVI: BVI interfaces use the same IP and MAC addresses on all PE routers so devices can reach the same gateway address regardless of the designated forwarder.

From Release 25.1.1, you can implement EVPN IRB with distributed anycast gateway.

Benefits of EVPN IRB

Provides details on how EVPN IRB enables workload mobility and routed connectivity across EVPN fabrics, allowing flexible VM placement and efficient routing in data center environments.

- Hosts in the same IP subnet can be provisioned anywhere within the EVPN fabric, supporting flexible VM deployment.
- Virtual machines in a subnet can move behind different EVPN PE devices without requiring a subnet change.
- VMs do not need to be directly connected or located in the same physical complex, enabling greater scalability and mobility.
- EVPN PE devices route traffic efficiently through MPLS encapsulation to maintain connectivity and support seamless VM mobility.

EVPN IRB route types

Provides details about the route types used in EVPN IRB to distribute host and prefix reachability information.

EVPN IRB uses two main route types to carry Layer 2 and Layer 3 reachability information:

- Route type 2 (MAC/IP advertisement route): Advertises host IP and MAC addresses to peers within Network Layer Reachability Information (NLRI), reducing unknown unicast flooding.
- Route type 5 (IP prefix route): Advertises IP prefixes in the EVPN domain and integrates Layer 3 routing with the EVPN infrastructure.

Table 42: EVPN IRB route type formats

Route type	NLRI format	Net attributes	Path attributes
Route type 2	[Type][Len][RD][ESI][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr][MPLS Label1][MPLS Label2]	[Type][RD][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr]	[ESI], [MPLS Label1], [MPLS Label2]
Route type 5	[Type][Len][RD][ESI][ETag][IP Addr Len][IP Addr][GW IP Addr][Label]	[Type][RD][ETag][IP Addr Len][IP Addr]	[ESI], [GW IP Addr], [Label]

Figure 27: Route type 2 NLRI format

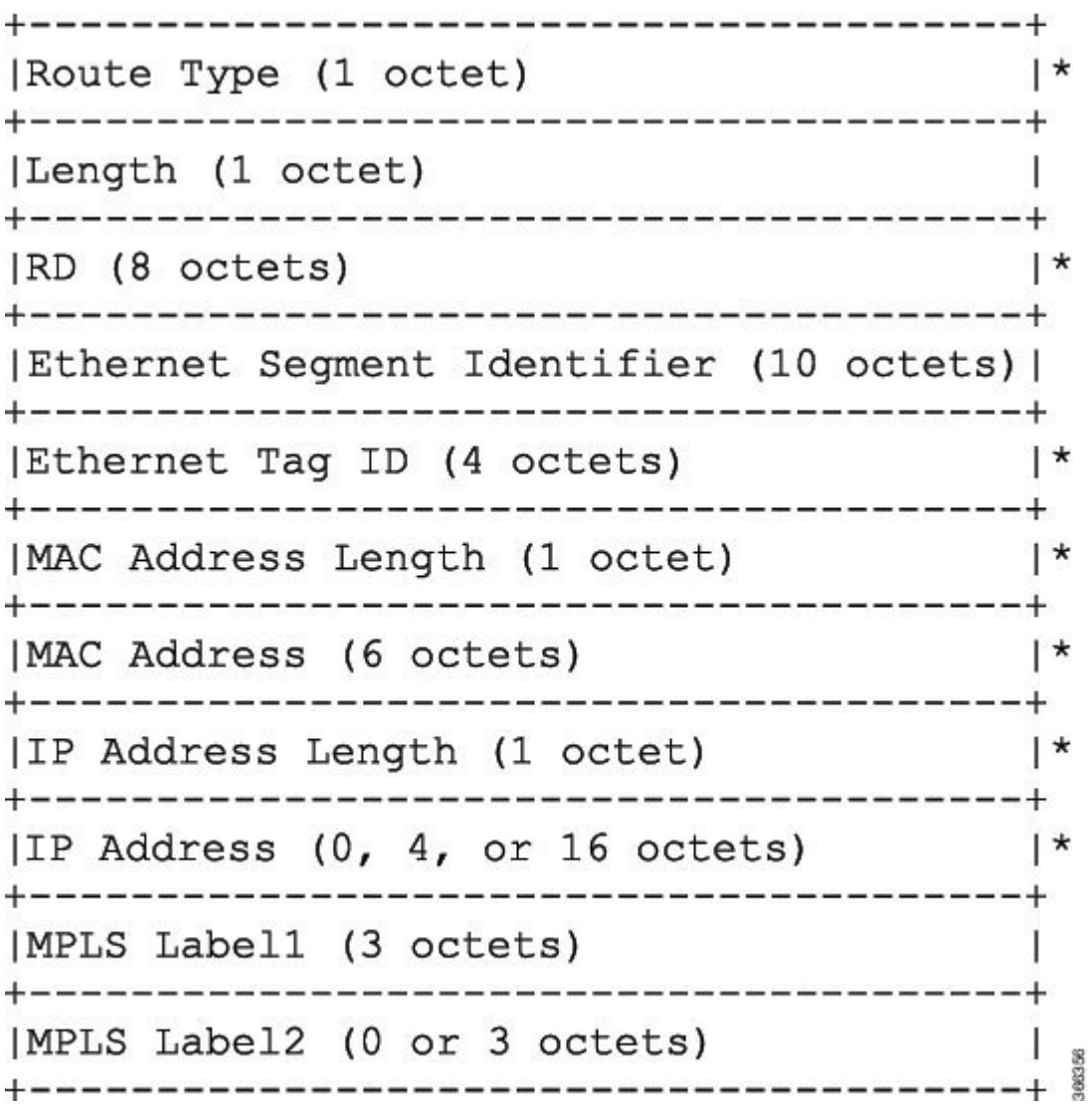
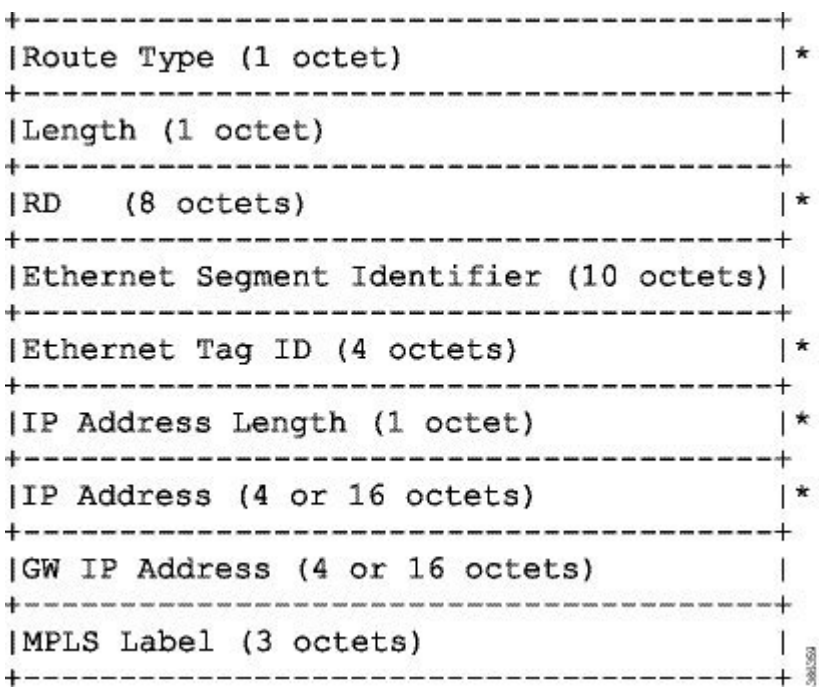


Figure 28: Route type 5 NLRI format



Example policy for route type 2:

```
route-policy evpn-policy
  if rd in (10.0.0.2:0) [and/or evpn-route-type is 2] [and/or esi in
(0000.0000.0000.0000.0000)] [and/or etag is 0] [and/or macaddress in
(0013.aabb.ccdd)] [and/or destination in (1.2.3.4/32)] then
    set ..
  endif
end-policy
```

Example policy for route type 5:

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

EVPN IRB with distributed anycast gateways

Explains how distributed anycast gateways allow multiple EVPN PE devices to share the same gateway IP and MAC addresses for a subnet, providing efficient routing and redundancy.

A distributed anycast gateway is an EVPN IRB feature that

- provides routing on the first hop
- enables load balancing and redundancy across PE nodes, and
- supports communication between EVPN hosts or subnets and IP VPNs.

Feature History Table
Table 43: Feature History Table

Feature name	Release information	Feature description
EVPN IRB with distributed anycast gateway	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
EVPN IRB with distributed anycast gateway	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
EVPN IRB with distributed anycast gateway	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8404-SYS-D router.
EVPN IRB with distributed anycast gateway	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) * This feature is supported on: • 8712-MOD-M • 8711-48Z-M
EVPN IRB with distributed anycast gateway	Release 25.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100]),8700 [ASIC: P100])(select variants only*); Centralized Systems (8800 [ASIC: P100])(select variants only*) * This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH

Feature name	Release information	Feature description
EVPN IRB with distributed anycast gateway	Release 25.1.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100])(select variants only*) EVPN Integrated Routing and Bridging (IRB) facilitates efficient Layer 3 communication across subnets, leveraging PE routers for connectivity over MPLS or IP networks. It supports single and multi-homing, processes packets using VRF table lookups, and enables seamless EVPN to IP VPN communication without route stitching or re-origination. A distributed anycast gateway enhances routing by sharing IP/MAC addresses for load balancing and redundancy, ensuring optimal performance and reduced latency. * This feature is supported on: • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

EVPN IRB capabilities and distributed gateway features

Lists the key capabilities and architectural features of EVPN IRB, including gateway behavior, control plane roles, and overlay routing.

EVPN IRB provides both Layer 2 and Layer 3 VPN overlay services, enabling communication within and across subnets in the same VPN. Its core capabilities include:

EVPN IRB includes these capabilities:

- Single-homing: Supports connection of a device to a single PE node.
- Multihoming: Offers all-active, single-active, and port-active multihoming options for redundancy and load balancing.
- MAC/IP advertisement route: Advertises MAC/IP address bindings within the EVPN network.
- IP prefix route: Propagates IP prefixes for advanced routing scenarios.
- MAC aging and MAC freezing: Allows for management of MAC table entries based on activity.
- Symmetric IRB forwarding: Enables routing of Layer 3 traffic on both source and destination PE nodes for optimal traffic flow and VM mobility.
- RT-stitching of subnet route (RT2) into VPNv4/VPNv6: Integrates subnet routes into broader VPN infrastructures.
- Subnet route (RT5) interconnect to VPNv4/VPNv6: Provides interconnection between subnet routes and VPN instances.
- Distributed anycast L3 gateway: All active PE nodes share the same gateway IP/MAC for redundancy and efficient routing.

EVPN IRB architecture and forwarding model

The EVPN IRB architecture incorporates several important components to enable scalable, resilient data center connectivity:

- Distributed anycast L3 gateway: All PE devices use the same gateway IP and MAC address, ensuring first-hop routing, load balancing, reduced latency, and improved redundancy. Each PE is an active Layer 3 gateway without standby roles.
- Anycast address in the distributed gateway: For each subnet, all PE nodes use the same gateway IP and MAC address (the anycast pair), supporting redundancy and transparent failover.

- EVPN PE connectivity: EVPN provider edge (PE) devices connect via a spine-leaf topology and maintain IP reachability to each other's loopbacks using an MPLS-based underlay fabric.
- EVPN control plane: An MPLS control plane distributes Layer 2 MAC and Layer 3 IP reachability. This capability enables hosts on the same subnet and Layer 2 domain to communicate across the fabric as if on a single network.
- Routing service: EVPN supports subnet stretching and seamless inter-subnet Layer 3 VPN services. PEs provide Layer 2 bridging, Layer 3 VPN services, and inter-subnet routing within the VPN.
- Symmetric IRB forwarding: Layer 3 traffic between hosts is routed both at the source and destination PE, supporting a single anycast gateway address and maintaining VM mobility. The forwarding model ensures optimal traffic flow.

How EVPN IRB distributed anycast gateway works

Describes how EVPN IRB distributed anycast gateways enable seamless VM mobility and uninterrupted connectivity in data center networks.

This process is essential in data centers where workloads frequently migrate, ensuring uninterrupted service and simplified network management during VM mobility.

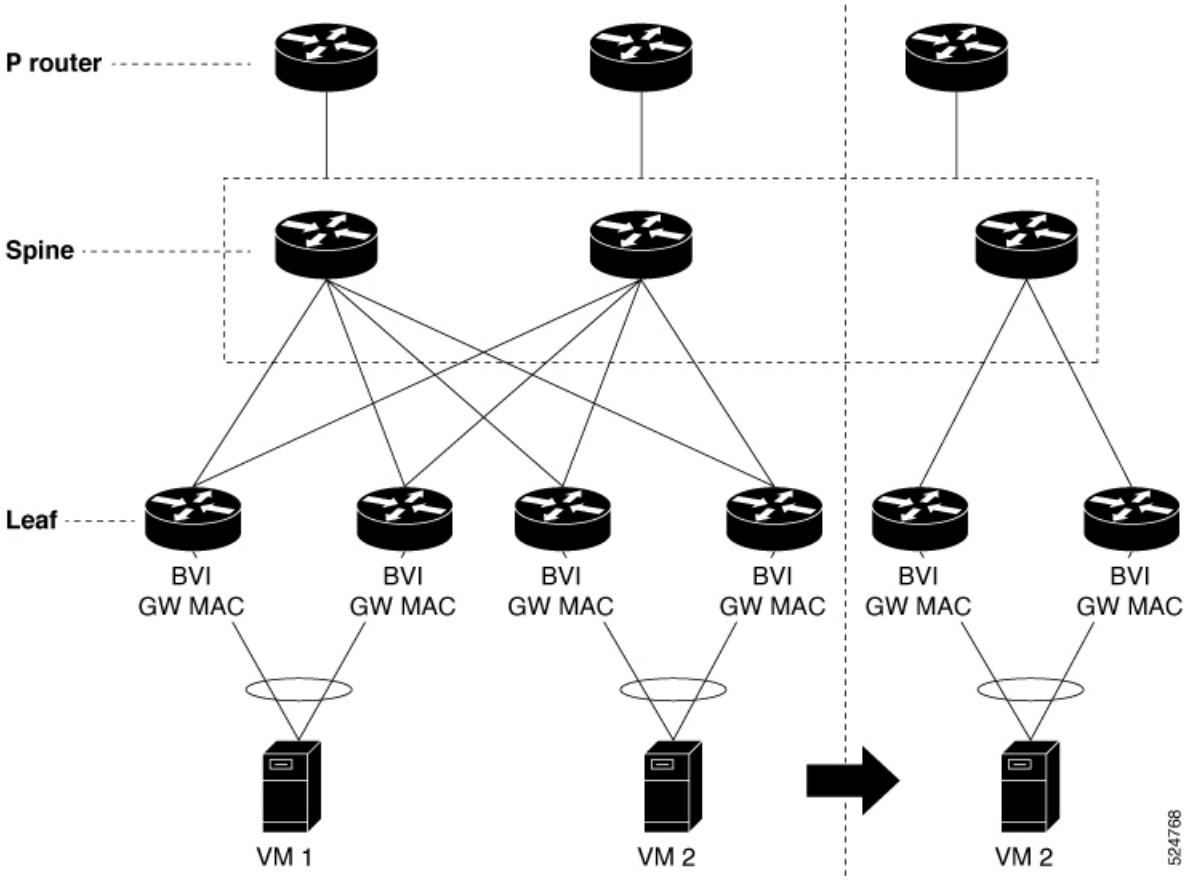
Summary

The key components involved in the process are:

- EVPN PE: Acts as the gateway, handles Layer 2 and Layer 3 forwarding, and supports the anycast gateway functionality across devices.
- Virtual machines (VMs): Endpoints that retain connectivity and their assigned IP/MAC addresses even when they move to different hosts or switches.
- MPLS tunnels (fabric): Carry traffic between PEs, enabling layer-2 extension and distributed gateway services.

The EVPN IRB distributed anycast gateway process enables virtual machines to maintain seamless connectivity and consistent addressing as they move within the data center, by leveraging distributed gateway configuration and optimized traffic forwarding.

Workflow



These stages describe the EVPN IRB distributed anycast gateway process:

1. Gateway configuration and distribution: Each EVPN PE in the fabric is configured with the same anycast gateway MAC and IP addresses for the relevant subnet. This ensures every PE acts as a gateway for local VMs.
2. VM connectivity and mobility: When a VM connects to a switch, it uses the local PE's anycast gateway for default routing (Layer 3), regardless of location. If the VM migrates to another host or PE, it continues to use the same MAC/IP gateway address, eliminating the need to update its network configuration.
3. Traffic forwarding across the Ffabric: Traffic between VMs in the same subnet but on different PEs is carried via MPLS tunnels. The distributed anycast gateway ensures efficient east-west traffic flow by routing packets as close to the destination as possible, reducing tromboning.
4. Gratuitous ARP/ND and MAC learning: When a VM migrates, the local PE immediately broadcasts a gratuitous ARP or Neighbor Discovery message so that the new location is learned by other devices, allowing swift redirection of traffic and avoiding traffic loss.

Key characteristics of the MAC and IP unicast control plane for distributed anycast gateway

Provides reference information on prefix routing, host routing, ARP and MAC synchronization, route re-origination, and key characteristics in an EVPN fabric.

The MAC and IP unicast control plane in a distributed anycast gateway supports several routing models and behaviors in an EVPN fabric, including prefix routing, host routing, ARP and MAC synchronization, and route re-origination.

Prefix routing (no subnet stretch)

- Establishes IP reachability using subnet prefix routes advertised via EVPN Route Type 5 with VPN labels and VRF route targets.
- Synchronizes host ARP and MAC entries across multi-homing EVPN PEs using MAC+IP Route Type 2 with a shared Ethernet Segment Identifier (ESI), supporting BVI all-active multihoming and load balancing.

Host routing (stretched subnet)

- When a host is discovered via ARP, it advertises MAC and IP Route Type 2 with MAC VRF route targets, IP VRF route targets, and VPN labels for both MAC-VRF and IP-VRF.
- Associates VRF route targets and Layer 3 VPN labels with Route Type 2 to enable PE-to-PE IP routing consistent with traditional L3VPN behavior.
- Allows remote EVPN PEs to install IP/32 host entries directly in the Layer 3 VRF table using the advertising EVPN PE next hop with Layer 3 VPN label encapsulation.
- Provides a scale advantage by enabling multiple IP hosts in a stretched subnet to share forwarding rewrites or load-balancing resources.

ARP and MAC sync

- Hosts connecting through a LAG to multiple EVPN PEs are learned locally in the data plane.
- The router synchronizes these ARP and MAC entries across multihoming EVPN PEs using MAC and IP Route Type 2 with a shared ESI.
- When a router receives a MAC and IP Route Type 2 with a local ESI, it installs a synchronized MAC entry pointing to the local AC port and a synchronized ARP entry on the local BVI interface.

MAC and IP route re-origination

- Routers may re-originate MAC and IP Route Type 2 received with local ESI for MAC and ARP synchronization if the host is not locally learned.
- Re-origination is required to establish overlay IP ECMP paths on remote EVPN PEs and to minimize traffic impact during local AC link failures, which can trigger MAC and IP route withdrawal.

Table 44: Key characteristics

Area	Reference information
IP reachability	The fabric uses Route Type 5 for subnet prefix routing and Route Type 2 for host-specific routing and synchronization.
Multihoming support	The design uses a shared ESI to synchronize MAC and ARP information across multihoming EVPN PEs.
Stretched subnet behavior	Remote PEs install IP/32 routes directly in the Layer 3 VRF by using the advertising PE next hop and Layer 3 VPN label.
Scale benefit	The design reduces per-host forwarding state by allowing shared forwarding rewrites or load-balance resources.
Resiliency	Route re-origination helps maintain overlay ECMP and reduces traffic disruption during local access link failures.

EVPN single-homing access gateway

Explains how EVPN single-homing connects each customer edge device to only one provider edge device while preserving host and subnet reachability through EVPN.

A single-homing method in EVPN is an access-gateway architecture that

- connects each customer edge (CE) device to only one provider edge (PE) device
- preserves EVPN host and subnet reachability, and

- enables PE devices to learn and advertise MAC and IP addresses for optimal connectivity.

All the PE nodes add host routes to the IP-VRF table, and the EVPN PE nodes add MAC routes to the MAC-VRF table. The PE device attaches to the Ethernet Segment through bundle interfaces or physical interfaces, using a Null Ethernet Segment Identifier (ESI) for single-homing.

Feature History Table
Table 45: Feature History Table

Feature Name	Release Information	Feature Description
EVPN single-homing access EVPN gateway over SRv6 core	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100]); Centralized Systems (8400 [ASIC: K100]); Modular Systems (8800 [LC ASIC: P100]) You can deploy an EVPN single-homing access EVPN gateway to provide Layer 2 and Layer 3 VPN services using EVPN technology over an SRv6 core. In this architecture, each customer edge (CE) device is connected to only one provider edge (PE) device, enabling a single-homed configuration that simplifies connectivity between CE devices and the service provider programmable, flexible SRv6 network.

How EVPN single-homing access gateway works

Describes how an EVPN single-homing access gateway connects a customer edge (CE) device to the network using a single provider edge (PE) device.

EVPN single-homing is used when a customer edge device connects to only one provider edge device, simplifying deployment and eliminating redundancy.

Summary

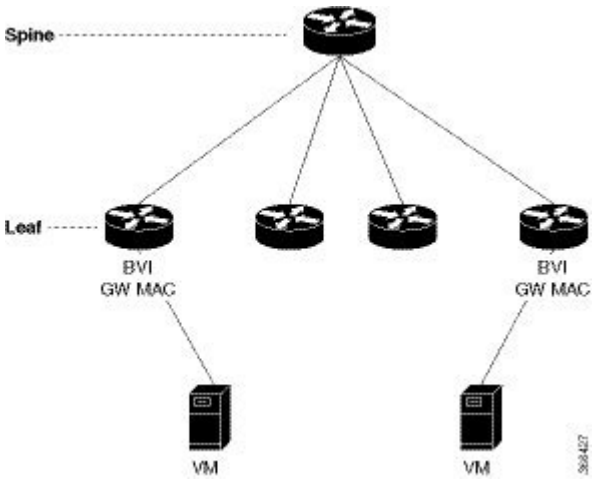
The key components involved in the process are:

- Customer Edge (CE) device: Connects to the network through the access gateway and interfaces with customer resources.
- Provider Edge (PE) device: Serves as the single point of connection for the CE device and attaches to the Ethernet segment.
- Ethernet Segment Identifier (ESI): Identifies the Ethernet segment. For single-homing, a null ESI is used.

The EVPN single-homing access gateway provides network connectivity for a single-homed customer by linking a CE device to the provider's backbone through a single PE device and an Ethernet segment.

Workflow

Figure 29: How EVPN single-homing access gateway works



These are the stages of EVPN single-homing access gateway:

1. Device connection: The CE device connects to the PE device using a physical or bundle interface.
2. Null ESI configuration: The PE device is configured with a null ESI to specify that only a single PE serves a given Ethernet segment.
3. EVPN route advertisement: The PE device advertises the CE connection into the EVPN network using BGP.
4. Traffic forwarding: The PE forwards traffic between the CE device and the core network, ensuring only one active path and no multi-homing redundancy.

EVPN all-active multihoming access gateway

Explains how EVPN all-active multihoming access gateways enable a customer edge (CE) device to connect redundantly to multiple provider edge (PE) devices for uninterrupted network access.

An EVPN all-active multihoming access gateway is a network access-gateway model that

- allows a customer edge (CE) device to connect to more than one provider edge (PE) device for redundancy
- prevents disruptions to network connectivity by enabling multihoming through various Ethernet links, and
- uses the Multi-chassis Link Aggregation Group (MC-LAG) bundle as an Ethernet segment to aggregate these connections.

An Ethernet segment refers to the group of Ethernet links that connect a CE device to multiple PE devices in an all-active multihoming deployment. The MC-LAG bundle allows all connected links to participate actively, providing efficient load balancing and rapid failover should any single link or device fail.

EVPN IRB behavior in all-active multihoming without subnet stretch or host-routing across the fabric

Lists the behaviors and route/state-handling characteristics for EVPN IRB in all-active multihoming deployments that do not use subnet stretch or host routing across the fabric.

EVPN IRB with all-active multihoming without subnet stretch or host-routing across the fabric provides a distributed anycast gateway for subnets that remain local to a set of multi-homing EVPN PEs.

For these local subnets, the solution advertises subnet routes by using EVPN Route Type 5 to remote leaf nodes that host the VRF. It does not require advertising /32 host routes within the subnet across the fabric. However, host MAC and ARP entries must remain synchronized across the EVPN PEs to which the servers are multi-homed.

Characteristics

- Uses all-active Ethernet LAG on the access side.
- Uses Layer 3 ECMP across the fabric for dual-homed hosts based on subnet routes.

- Does not stretch the Layer 2 subnet from remote PEs to the local EVPN IRB multi-homing PE.
- Supports Layer 2 stretch only within the redundancy group of leaf nodes that contain orphan ports.

Route and state handling across multihoming EVPN PEs

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

Route handling across remote EVPN PEs

- Dual-homed MAC+IP EVPN Route Type 2 routes are exchanged with the ESI, EVI label, and Layer 2 route type.
- If there is no subnet stretch or host-routing, these Route Type 2 routes are not imported across the fabric.
- Subnet IP EVPN Route Type 5 routes are exchanged with the VRF label and Layer 3 route type.
- Layer 3 route types are imported only for VRFs that are present locally.
- Layer 2 route types for locally present bridge domains (BDs) are imported.
- If a BD is not stretched, the Layer 2 route type is imported only from the leaf in the same redundancy group.

Summary

This model uses prefix routing for inter-subnet traffic from remote PEs to the EVPN IRB multi-homing PE. It keeps host synchronization where needed for dual-homed access while avoiding subnet stretch and host-route distribution across the broader fabric.

To configure the BVI for EVPN IRB with all-active multihoming without subnet stretch or host-routing across the fabric, see the **Configure IRB** section.

EVPN IRB behavior in all-active multihoming with subnet stretch or host-routing across the fabric

Lists the characteristics, routing behaviors, and limitations of EVPN IRB in all-active multihoming scenarios with subnet stretch or host-routing across the fabric.

EVPN IRB with all-active multihoming with subnet stretch or host-routing across the fabric distributes both /32 host routes and MAC routes in the EVPN overlay control plane. This model enables both Layer 2 and Layer 3 traffic forwarding to endpoints that belong to a stretched subnet across remote EVPN PEs.

Characteristics

- Uses Layer 2 or Layer 3 ECMP across the fabric for dual-homed hosts based on Route Type 1 and Route Type 2.
- Uses Layer 3 unipath across the fabric for single-homed hosts based on Route Type 2.
- Supports Layer 2 subnet stretch across the fabric.
- Supports Layer 2 stretch within the redundancy group of leaf nodes that contain orphan ports.

Traffic and route distribution

- Distributes /32 host routes across the EVPN overlay control plane.

- Distributes MAC routes across the EVPN overlay control plane.
- Enables Layer 2 connectivity to endpoints in stretched subnets.
- Enables Layer 3 connectivity to endpoints in stretched subnets.

Scope and limitation

- The subnet stretch feature with EVPN IRB is available only within VRF instances.
- The subnet stretch feature does not apply to the global VRF.

To configure the BVI for EVPN IRB with all-active multi-homing with subnet stretch or host-routing across the fabric, see the **Configure EVPN IRB with distributed anycast gateway** section.

EVPN IRB port-active multihoming

Explains EVPN IRB port-active multihoming mode as a network model that provides single-active redundancy and fast convergence at the interface level.

EVPN IRB port-active multihoming mode is a network model that

- sends traffic to a specific interface, avoiding per-flow load balancing across multiple PE routers
- can be enabled only on bundle interfaces, and
- supports both Layer 2 and Layer 3 port-active functionality on the same bundle while using designated forwarder (DF) election to determine the active and standby PE.

You can use either the modulo algorithm or the Highest Random Weight (HRW) algorithm for per-port DF election. By default, the modulo algorithm is used. If the interface runs LACP, the standby side sets LACP to Out-of-Service instead of bringing the interface down.

EVPN IRB port-active multihoming behavior and benefits

Provides an overview of EVPN IRB port-active multihoming, including its advantages, supported topologies, and a comparison with ICCP-based MC-LAG solutions

EVPN IRB port-active multihoming improves convergence during a link failure by allowing only one physical port to remain active at a time. This model simplifies protocol operation because traffic uses a single active port while the other port remains in standby. This functionality is available only on bundle interfaces.

Key benefits

- Provides faster convergence during link failure.
- Simplifies protocol operation because only one physical port is active at a given time.
- Acts as an alternative to multi-chassis link aggregation group (MC-LAG) with ICCP.
- Supports deployments where certain QoS features must operate correctly.

Comparison with ICCP-based MC-LAG

- EVPN port-active simplifies the protocol model compared to Inter-Chassis Communication Protocol (ICCP).
- I

  CCP runs on top of Label Distribution Protocol (LDP).
- EVPN port-active can replace MC-LAG with ICCP in suitable deployments.

Port activity model

- One PE operates in active mode at the port level.
- Another PE operates in standby mode at the port level.
- Only the PE in active mode sends and receives traffic.
- The PE in standby mode does not forward traffic while it remains in standby.

Designated forwarder election

- The PEs use the Designated Forwarder (DF) election mechanism to determine active and standby roles.
- The DF election selects which PE must operate in active mode and which PE must operate in standby mode.
- The per-port DF election supports the modulo algorithm.
- The per-port DF election also supports the Highest Random Weight (HRW) algorithm.
- The modulo algorithm is the default selection.

How EVPN IRB port-active multihoming works

Describes the process of EVPN IRB port-active multihoming, including key components, workflow stages, and how the approach enables redundancy and rapid role transitions.

This process supports EVPN IRB multihoming deployments where the customer edge device connects redundantly to two provider edge devices. Only one interface forwards traffic at a time, with fast switch-over between standby and active modes to maintain seamless service.

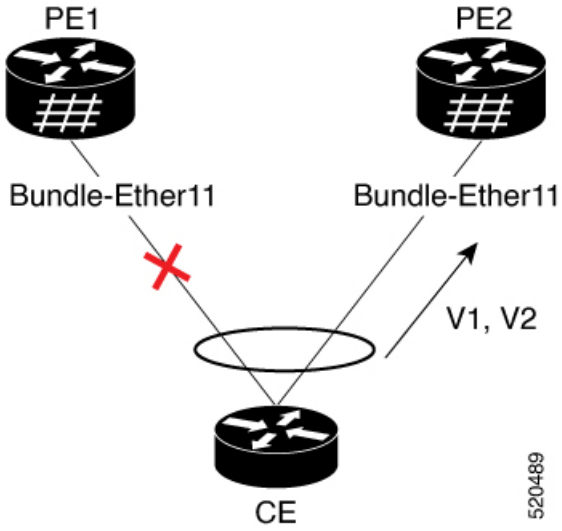
Summary

The key components involved in the process are:

- Customer edge device (CE): Connects to both provider edge devices through a multihomed topology and uses single link aggregation.
- Provider edge device 1 (PE1): Operates in standby mode and keeps its services in standby when it is not the active forwarding device.
- Provider edge device 2 (PE2): Operates in active mode and carries traffic from the customer edge device.
- Active interface: Remains in the forwarding state and handles traffic and service operation.
- Standby interface: Remains in the standby state and does not forward traffic unless an active-to-standby transition occurs.
- LACP: Sets the standby interface state to Out-of-Service (OOS) instead of bringing the interface down, which improves convergence during transition.
- L2 and L3 subinterfaces: Participate in EVPN IRB and allow both Layer 2 and Layer 3 port-active functionality to coexist on the same bundle.

EVPN IRB port-active multihoming enables a customer edge device to connect redundantly to two provider edge devices, ensuring only one interface actively forwards traffic at a time. This approach supports rapid convergence and maintains efficient traffic handling and service operation.

Workflow



These stages describe how EVPN IRB port-active multihoming works:

1. Topology establishment: The customer edge device connects to PE1 and PE2 in a multihomed topology and uses single link aggregation
2. Role assignment: The system places one interface in the forwarding state and places the other interface in the standby state.
3. Active forwarding: PE2 operates in active mode and carries traffic from the customer edge device.
4. Standby service state: PE1 operates in standby mode, and all services on its interface remain in standby.
5. LACP standby handling: If the interface runs LACP, the standby device sets the LACP state to Out-of-Service instead of setting the interface state down.
6. Convergence improvement: The Out-of-Service state helps the system achieve better convergence when the standby interface transitions to active mode.
7. Port-active reconfiguration: If an operator removes the port-active configuration from both PE1 and PE2 and then adds it back on both devices, the system selects PE2 as the active interface again.
8. Mixed service support: The bundle supports both Layer 2 and Layer 3 port-active functionality at the same time.
9. EVPN IRB participation: The same bundle includes a mix of L3 subinterfaces and L2 subinterfaces that participate in EVPN IRB.

EVPN single-active multihoming for anycast gateway IRB

Explains how single-active multihoming operates in an EVPN service instance for anycast gateway IRB deployments.

A single-active multihoming for anycast gateway IRB is an EVPN forwarding model that

- uses one Provider Edge (PE) device to forward traffic to and from an Ethernet Segment within each EVPN service instance
- enables local PE nodes connected to the Ethernet segment to load-balance traffic based on the Ethernet Virtual Instance (EVI), and
- performs designated forwarder (DF) election between PEs to ensure that only the DF forwards traffic while the non-DF blocks traffic in both directions.

Feature History Table
Table 46: Feature History Table

Feature Name	Release Information	Feature Description
EVPN single-active multihoming for anycast gateway over SRv6 core	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100]); Centralized Systems (8400 [ASIC: K100]); Modular Systems (8800 [LC ASIC: P100]) You enable EVPN single-active multihoming for anycast gateway IRB over an SRv6 core to provide single-active redundancy, where only one PE forwards traffic for an Ethernet Segment within each EVPN service instance. This feature supports intersubnet scenarios and balances traffic based on the EVI, leveraging a programmable, flexible SRv6 network.

How EVPN single-active multihoming for anycast gateway IRB works

Describes the processes by which EVPN single-active multihoming enables seamless failover, prevents duplicate forwarding, and ensures reliable gateway behavior for anycast gateway IRBs.

This process supports EVPN multihoming in single-active mode. This mode ensures that only one peering PE forwards traffic for a shared EVI at a time, while the other PE remains blocked. This behavior prevents duplicate forwarding and provides failover when the active path fails.

Summary

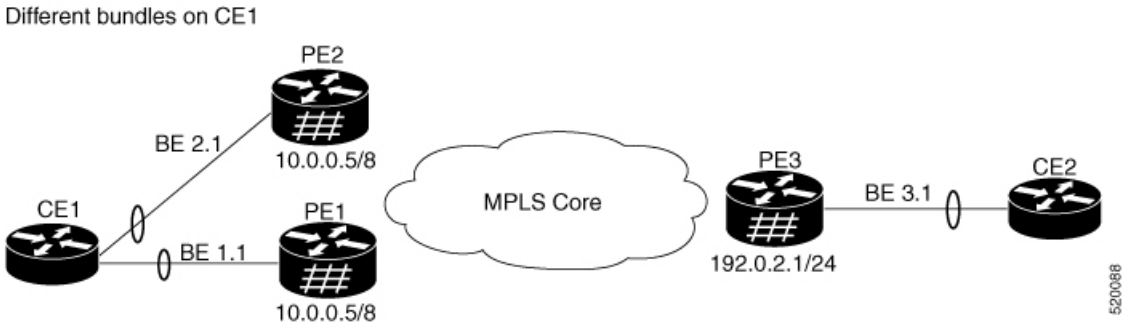
The key components involved in the process are:

- CE1 and CE2: A multihomed customer edge device that connects to both PE1 and PE2 through bundle Ethernet interfaces in the same switching domain.

  CE2: A customer edge device that receives traffic from PE3.
- PE1: A peering provider edge device that participates in host routing, uses the anycast gateway IP address, advertises Type 4 routes, and can become the designated forwarder.
- PE2: A peering provider edge device that mirrors the PE1 configuration, participates in designated forwarder election, and remains in standby or blocked state when it is the non-DF.
- PE3: A remote provider edge device that connects to CE2 through an Ethernet interface bundle and exchanges reachability information with PE1 and PE2 through the MPLS core.
- Bundle Ethernet interfaces BE 1.1 and BE 2.1: The multihomed access interfaces on CE1 that must belong to the same switching domain.
- MPLS core: The transport network that connects PE1 and PE2 to PE3.
- Anycast gateway IP address: The shared gateway address configured on both peering PEs to provide consistent gateway behavior.
- Type 4 EVPN routes: The routes PE1 and PE2 advertise to signal multihoming information and support designated forwarder election.
- Designated forwarder election: The mechanism that selects the active forwarding PE for the shared Ethernet VPN instance in single-active mode.

This process describes how EVPN single-active multihoming forwards traffic between a multihomed customer edge device and a remote customer edge device by electing a designated forwarder and blocking the standby path.

Workflow



These stages describe how EVPN single-active multihoming for anycast gateway IRB works:

1. Configure the multihomed access: CE1 connects to PE1 and PE2 through bundle Ethernet interfaces BE 1.1 and BE 2.1, and the ingress interface belongs to the same switching domain on CE1.
2. Enable EVPN gateway and reachability functions: PE1 and PE2 enable host routing, configure the same anycast gateway IP address, connect to PE3 through the MPLS core, and maintain reachability with PE3 subnets. CE2 connects to PE3 through an Ethernet interface bundle.
3. Advertise multihoming information: PE1 and PE2 advertise EVPN Type 4 routes for the shared EVI.
4. Elect the designated forwarder: PE1 and PE2 perform designated forwarder election. In single-active mode, one PE becomes the DF and the other PE becomes the non-DF.
5. Block the standby path: The non-DF blocks traffic in both directions, so only the DF forwards traffic for the shared EVI.
6. Process the ARP request from CE1: CE1 sends an ARP broadcast request to both PE1 and PE2. If PE1 is the DF, PE1 replies to the ARP request and PE2 drops the traffic from CE1.
7. Forward unicast traffic through the active PE: After ARP resolution completes, CE1 sends unicast traffic only through the DF. The non-DF remains in standby or blocked state and does not forward traffic.
8. Exchange remote MAC and data traffic: The active PE advertises the MAC to PE3, and PE3 always sends and receives traffic through that active PE. PE3 forwards the traffic to CE2 over the Ethernet interface bundle.
9. Fail over to the standby PE when the active bundle fails: If BE1 fails, PE2 becomes the designated forwarder and traffic starts to flow through PE2.

Configure EVPN IRB with distributed anycast gateway

Configure the bridge domain, BVI, EVPN signaling, and access connectivity to enable distributed anycast gateway integrated routing and bridging (IRB).

Set up distributed anycast gateway IRB using EVPN, enabling seamless bridging and routing between Layer 2 and Layer 3 domains with enhanced redundancy.

Use this task when you need to implement distributed anycast gateway functionality for IRB in your EVPN deployment, providing efficient and redundant Layer 3 gateway services across your network.

Procedure

1. Configure EVPN distributed anycast IRB in the bridge domain.

Example

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 10
Router(config-l2vpn-bg)# bridge-domain 2
Router(config-l2vpn-bg-bd)# routed interface BVI10
Router(config-l2vpn-bg-bd-bvi)# split-horizon group core
Router(config-l2vpn-bg-bd-bvi)# exit
Router(config-l2vpn-bg-bd)# evi 100
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# commit
```

2. Configure the BVI interface for the EVPN routing model that you need.

Example

```
Router# configure
Router(config)# interface BVI10
Router(config-if)# host-routing
Router(config-if)# vrf 30
Router(config-if)# ipv4 address 10.0.0.5 255.0.0.0
Router(config-if)# local-proxy-arp
Router(config-if)# mac-address 1.1.1
Router(config-if)# commit
```

To configure the BVI without subnet stretch, omit **host-routing** and **local-proxy-arp**.

```
Router# configure
Router(config)# interface BVI10
Router(config-if)# vrf 30
Router(config-if)# ipv4 address 10.0.0.5 255.0.0.0
Router(config-if)# mac-address 1.1.1
Router(config-if)# commit
```

3. Configure EVPN BGP signaling and add the attachment circuit to the bridge domain.

Example

```
Router# configure
Router(config)# router bgp 200
Router(config-bgp)# bgp router-id 10.10.10.1
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# neighbor 10.10.10.10
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# exit
Router(config-bgp)# exit
Router(config-evpn)# evi 100
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# exit
Router(config-evpn)# exit
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 10
Router(config-l2vpn-bg)# bridge-domain 2
Router(config-l2vpn-bg-bd)# interface FourHundredGigE0/0/0/0.2
Router(config-l2vpn-bg-bd)# exit
```

```
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# exit
Router(config)# exit
```

4. Configure the access pseudowire in the bridge domain.

Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 10
Router(config-l2vpn-bg)# bridge-domain 2
Router(config-l2vpn-bg-bd)# neighbor 192.0.2.1 pw-id 100
Router(config-l2vpn-bg-bd)# commit
```

The bridge domain is now enabled for distributed anycast gateway IRB with EVPN signaling and access connectivity. The configuration provides high-availability and seamless integrated routing and bridging across the EVPN fabric.

EVPN IRB software MAC learning

Explains how EVPN IRB software MAC learning mechanisms discover, distribute, and synchronize MAC addresses across VLANs for consistent network forwarding.

EVPN IRB software MAC learning is a network capability that

- learns the MAC addresses of devices available in a VLAN
- distributes MAC addresses learned on one device to other devices in the VLAN, and
- uses BGP to learn MAC addresses from remote devices.

- MAC address: A hardware address that identifies a device on a network.
- VLAN: A logical network segment that groups devices into the same broadcast domain.
- BGP: A routing protocol that can distribute reachability information, including MAC address information in EVPN environments.

MAC learning identifies which MAC addresses belong to devices on a VLAN. In EVPN IRB software MAC learning, this information does not stay only on the local device. Instead, the learned MAC addresses propagate to other connected devices so that the network maintains consistent forwarding information across the VLAN.

- A switch learns a host MAC address on a local VLAN and shares that information with other devices in the same EVPN network.
- A remote device advertises MAC address information through BGP, and another device installs that information for forwarding decisions.

How software MAC learning works in EVPN IRB

Describes how software MAC learning distributes learned MAC addresses across EVPN devices in an integrated routing and bridging (IRB) environment using BGP.

MAC learning is the method of learning the MAC addresses of all devices available in a VLAN.

The MAC addresses learned on one device must also be learned or distributed to the other devices in the VLAN. Software MAC learning enables the distribution of the MAC addresses learned on one device to the other devices connected to the network, and the remote devices learn those MAC addresses through BGP.

Summary

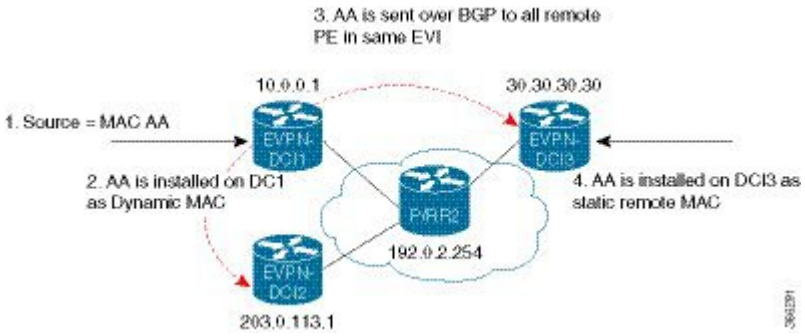
The key components involved in the process are:

- Ingress device: Learns the source MAC address as traffic enters its port in the bridge domain.
- BGP protocol: Converts learned MAC addresses into type-2 BGP routes for distribution across the EVPN.
- Remote PE devices: Receive and update MAC addresses as static remote entries to maintain synchronized address tables.

Software MAC learning enables the distribution of MAC addresses learned on one device to other devices within the EVPN IRB network, ensuring consistent MAC address knowledge across network nodes.

Workflow

Figure 30: Software MAC learning



These stages describe the software MAC learning process.

1. Traffic enters one port in the bridge domain.
2. The ingress device learns the source MAC address and stores it as a dynamic MAC entry.
3. The MAC address is converted into a type-2 BGP route and sent over BGP to all remote PEs in the same EVI.
4. Each remote device updates the MAC address as a static remote MAC entry.

Result

At the end of the process, all EVPN devices in the IRB network maintain synchronized MAC address tables, enabling seamless communication and optimized forwarding across the VLAN.

MAC freezing in EVPN IRB

Explains how MAC freeze mechanisms detect duplicate IP addresses and block MAC-IP routes to maintain network security and integrity.

MAC freeze mechanism is a duplicate IP address detection feature that

- identifies hosts assigned duplicate IP addresses, either unintentionally or by malicious intent
- automatically blocks all MAC-IP routes associated with the duplicate IP address, and
- temporarily or permanently freezes routes when duplicate-detection thresholds are exceeded.

The router manages mobility of EVPN hosts by tracking MAC and IP addresses as they move across hosts. When two hosts are assigned the same IP address, IOS XR continually learns and re-learns MAC-IP routes from both hosts. Each MAC-IP route learning event is counted as a "move" because the newly learned route replaces the previous route from the other host.

These parameters control duplicate detection:

- **move-interval**: The period within which a MAC or IP address has to move certain number of times between different hosts to be considered as duplicate and frozen temporarily. This number is specified in the **move-count** parameter.
- **move-count**: The number of times a MAC or IP address has to move within the interval specified for the **move-interval** parameter between different hosts to be considered a duplicate.
- **freeze-time**: The length of time a MAC or IP address is locked after it has been detected as a duplicate. After this period, the IP address is unlocked and it is allowed to learn again.
- **retry-count**: The number of times a MAC or IP address is unlocked after it has been detected as a duplicate before it is frozen permanently.

When an IP address moves the specified number of times (**move-count**) within a specified interval (**move-interval**), the IP address is flagged as duplicate, and associated MAC-IP routes are frozen for the duration defined by **freeze-time**. After freeze-time elapses, MAC-IP routes are unfrozen, **move-count** resets to zero, local MAC-IP routes trigger ARP probe and flush, and remote MAC-IP routes re-enter probe mode to restart duplicate detection.

Manual unfreeze commands

The router maintains the number of times a particular IP address has been frozen and unfrozen. If an IP address is marked as duplicate again after it has been unfrozen retry-count times, it is frozen permanently until you manually unfreeze it.

The router records the number of times a particular IP address has been frozen and unfrozen. If an IP address is marked as duplicate more times than allowed by **retry-count**, it is frozen permanently until manually unfrozen. To unfreeze, use one of these commands:

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

A syslog notifies the user when an IP address is frozen. While an IP address is frozen, any new MAC-IP routes or updates to existing MAC-IP routes with that frozen IP address are ignored.

Configure MAC freezing

Configure duplicate IP address detection and MAC freezing parameters to ensure proper operation in the network.

Configure MAC freezing parameters to detect and prevent duplicate IP addresses on your network, reducing operational issues and improving security.

MAC freezing helps identify devices with duplicate IP addresses and temporarily freezes their MAC to prevent conflicts and maintain proper routing.

Procedure

1. Configure duplicate detection for IPv4 and IPv6 addresses.

Example

```
Router# configure
Router(config)# evpn
Router(config-evpn)# host ipv4-address duplicate-detection
Router(config-evpn-host-ipv4-addr)# move-count 2
Router(config-evpn-host-ipv4-addr)# freeze-time 10
Router(config-evpn-host-ipv4-addr)# retry-count 2
Router(config-evpn-host-ipv4-addr)# commit
Router(config-evpn-host-ipv4-addr)# end
```

```
Router# configure
Router(config)# evpn
Router(config-evpn)# host ipv6-address duplicate-detection
Router(config-evpn-host-ipv6-addr)# move-count 2
Router(config-evpn-host-ipv6-addr)# freeze-time 10
Router(config-evpn-host-ipv6-addr)# retry-count 2
Router(config-evpn-host-ipv6-addr)# commit
```

2. Check the running configuration.

Example

```
Router# show running-config
evpn
 host ipv4-address duplicate-detection
  move-count 2
  freeze-time 10
  retry-count 2
 !
!
evpn
 host ipv6-address duplicate-detection
  move-count 2
  freeze-time 10
  retry-count 2
 !
```

3. Use the **show l2route evpn mac-ip all detail** to verify duplicate IP address detection and recovery parameters.

Example

```
Router# show l2route evpn mac-ip all detail

Flags:  (Stt)=Static; (L)=Local; (R)=Remote; (F)=Flood;
        (N)=No Redistribution; (Rtr)=RP/0/RSP0/CPU0:router MAC; (B)=Best
Route;
        (S)=Peer Sync; (Spl)=Split; (Rcv)=Recd;
        (D)=Duplicate MAC; (Z)=Frozen MAC;
Topo ID    Mac Address     IP Address  Prod   Next Hop(s)        Seq No  Flags
         Opaque Data Type    Opaque Data Len   Opaque Data Value
-------    -----------     ----------  ----   ----------         ------  -----
         ----------------    ---------------   -----------------
33          0022.6730.0001  10.130.0.2  L2VPN  Bundle-Ether6.1300  0      SB
0 12      0x06000000
```

12 Multiple Spanning Tree Protocol

Topics:

- Multiple spanning tree protocol
- MSTP supported features
- Restrictions for MSTP
- Configure MSTP base services
- MSTP interface parameters
- Per-VLAN Rapid Spanning Tree
- Information about Multiple Spanning Tree Protocol

Introduces Multiple Spanning Tree Protocol concepts, supported features, configuration tasks, restrictions, interface parameters, protocols, and operational behaviors to guide deployment and optimize network efficiency across diverse environments.

Multiple spanning tree protocol

Provides an overview of Multiple Spanning Tree Protocol functionality and outlines MSTP instance mapping and load balancing techniques for efficient network segmentation and traffic optimization.

A multiple spanning tree protocol (MSTP) is a Spanning Tree Protocol (STP) variant that

- allows you to configure each spanning tree instance independently
- lets you select different root bridges or loop-free paths for each spanning tree, and
- enables blocking or unblocking physical interfaces for specific spanning tree instances.

You can configure the parameters for each spanning tree separately. This means you can choose different network devices as the root bridge or select distinct paths to form the loop-free topology. Therefore, you can block a given physical interface for some spanning trees and unblock it for others.

Feature History Table

Feature Name	Release Information	Feature Description
Multiple Spanning Tree Protocol	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Multiple Spanning Tree Protocol	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The MSTP functionality is now extended to the Cisco 8712-MOD-M routers.
Multiple Spanning Tree Protocol	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The MSTP functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Multiple Spanning Tree Protocol	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The Multiple Spanning Tree Protocol (MSTP) enhances network efficiency by allowing the creation of multiple, independent spanning trees over the same physical network. This flexibility enables customized configuration of parameters for each spanning tree, selection of different root bridges or paths, and the ability to block or unblock specific physical interfaces for different trees. *This functionality is extended to routers with the 88-LC1-36EH line cards.

MSTP instance mapping and load balancing

Provides details on how VLAN-to-instance mapping in MSTP distributes traffic across redundant links for load balancing.

MSTP allows you to partition VLANs among multiple spanning tree instances. By assigning different VLAN ranges to separate spanning tree instances, you can distribute data traffic across available redundant links to achieve load balancing.

A typical VLAN-to-instance mapping might include:

- VLANs 1-100 assigned to spanning tree instance 1
- VLANs 101-200 assigned to spanning tree instance 2
- VLANs 201-300 assigned to spanning tree instance 3

Since each spanning tree instance has its own active topology and uses different links, data traffic is divided among the available link paths based on VLAN assignment. This approach balances network traffic and optimizes link utilization.

MSTP supported features

Details MSTP supported features, including BPDU Guard, flush containment mechanisms with operational workflows, and bringup delay characteristics essential for robust network protection and loop avoidance.

The Cisco 8000 Series Routers support MSTP, as defined in IEEE 802.1Q-2005, with these features:

Supported interface types

- Physical Ethernet interfaces
- Ethernet bundle interface

Supported operating modes

- Standard 802.1Q mode
- Provider edge (802.1ad) mode (uses a different MAC address for BPDUs; transparently forwards BPDUs received with the 802.1Q MAC address)

Layer 2 loop prevention features

- Port Fast
- Bridge Protocol Data Unit (BPDU) Guard

Legacy-BPDU behavior

If the **allow-legacy-bpdu** command is not configured on the MST default instance and a bridge port receives a legacy BPDU, the port enters the **error-disable** state.

BPDU Guard

Provides details on how BPDU Guard protects edge ports in Multiple Spanning Tree Protocol (MSTP) environments by preventing misconfigurations that can compromise network stability.

The BPDU Guard feature protects against misconfigured edge ports within MSTP by ensuring that interfaces intended for edge use are not allowed to participate in the spanning tree if an MSTP Bridge Protocol Data Unit (BPDU) is received. BPDU Guard is an enhancement to the MSTP Port Fast feature. When Port Fast is configured on an interface, MSTP designates the interface as an edge port and excludes it from spanning tree calculations. With BPDU Guard enabled,

MSTP will automatically shut down that interface using error-disable if an MSTP BPDU is detected, thereby preventing accidental introduction of loops or changes to the spanning tree topology.

Flush containment

Provides details about the flush containment feature and its default behavior in MSTP.

Flush containment is a Cisco feature for MSTP that helps prevent unnecessary MAC address table flushes caused by unrelated topology changes in other parts of a network.

Key points about flush containment in MSTP:

- Prevents topology change notifications from being sent on interfaces where no VLANs are configured for the relevant Multiple Spanning Tree Instance (MSTI).
- Is enabled by default to avoid unnecessary MAC flushes.
- Can be disabled by configuration, which restores standard IEEE 802.1Q behavior (all appropriate interfaces will again send topology change notifications).
- Helps improve network stability and limits the scope of disruption during topology changes.

Default behavior

- Flush containment is enabled on Cisco MSTP by default.
- Disabling flush containment restores the standard IEEE 802.1Q behavior, which may be necessary for interoperability.

How flush containment works

Describes how flush containment confines topology-change MAC flushes to only the affected part of the network.

In a network where multiple VLANs exist (for example, VLAN 1 used only on device D, and VLAN 2 spanning devices A, B, and C), both VLANs can share the same spanning tree instance (MSTI) but not share any links. Traditionally, a topology change notification could trigger MAC address table flushes for all VLANs in the instance—even those not affected—leading to unnecessary disruption. Flush containment changes this behavior.

Summary

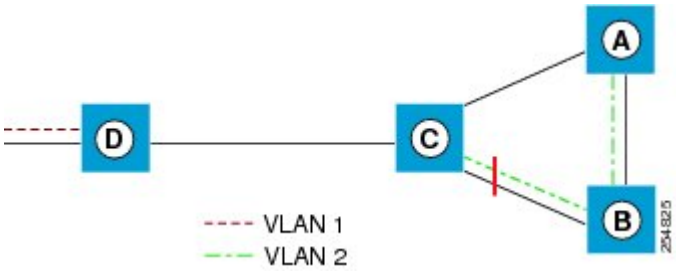
The key components involved in the process are:

- Spanning Tree Protocol (STP) devices: Switches that participate in topology change detection and notification.
- Multiple VLANs and MSTIs: Logical subdivisions of the network that may or may not be directly affected by a change.
- Topology change notification (TCN) mechanism: The signaling process for informing devices of topology events that may cause MAC table flushes.

Flush containment is a feature that limits topology-change-induced MAC address flushes to only those parts of a network directly affected by the change, increasing stability and reducing unnecessary disruptions.

Workflow

Figure 31: Flush Containment



These stages describe how flush containment affects topology change handling.

1. Topology change event: When a link fails (for example, link AB goes down), device C responds by activating its previously blocked port.
2. Traditional notification: In standard operation, device C sends a topology change notification (TCN) on all its other interfaces, including toward device D. This action causes MAC address flushes even for VLANs that are not impacted by the change (for example, VLAN 1 on device D).
3. Flush containment operation: With flush containment active, device C suppresses TCN messages on interfaces that do not carry VLANs belonging to the affected MSTI. In this scenario, no notification is sent from C to D, so MAC flushes do not occur for VLAN 1 on D.
4. Resulting limitation of flush scope: Only those sections of the network carrying the affected MSTI (the right-hand side with A, B, and C in the example) process the MAC address flush.

Bringup delay for MSTP interfaces

Provides an overview of the bringup delay feature for MSTP interfaces and how it prevents premature inclusion in spanning tree calculations.

Bringup delay is a Cisco feature that prevents MSTP (Multiple Spanning Tree Protocol) from considering an interface in the spanning tree calculation until the interface is fully ready to forward traffic. This feature is especially useful when a line card initially boots up, as the system may declare its interfaces as "up" before the data plane is actually able to forward traffic. According to the MSTP standard, interfaces are included in the calculation as soon as they are declared up, which can result in other interfaces being moved into the blocking state if the newly started interfaces are erroneously selected.

The bringup delay addresses this issue by introducing a configurable delay period when MSTP-configured interfaces first come into existence, such as after a card reload. During this delay, the interfaces remain in the blocking state and are excluded from spanning tree calculations until they are truly ready to forward traffic. It is important to note that bringup delay only applies when new MSTP-configured interfaces are created; if MSTP is later configured on an existing interface, no delay is applied.

Restrictions for MSTP

Outlines operational restrictions and guidelines for implementing MSTP to ensure network stability and compliance with supported features.

MSTP restrictions

Apply these restrictions when deploying MSTP on Cisco 8000 Series Router interfaces and bridge domains:

- Configure MSTP only on the main Layer 3 or Layer 2 interface.
- Map subinterfaces to MSTI instances by the outermost VLAN tag ID, regardless of whether encapsulation uses QinQ or a single VLAN tag.
- Treat split-horizon group alignment and MSTI grouping by VLAN ID as independent requirements.
- Use the same MSTI for all subinterfaces in a bridge domain when MSTP runs on the main interface.

- Do not create an untagged subinterface when MSTP runs on the main interface.
- Ensure the bridge-domain design meet the supported interface and VLAN-mapping rules before deploying MSTP.
- Redesign the interface and bridge-domain layout if your design requires a different encapsulation or subinterface model.

Configure MSTP base services

Guides configuration of MSTP base services, enabling essential MSTP functions for foundational network layer protection.

Set up the essential service objects required for MSTP operation, including VLAN interfaces, Layer 2 VPN bridge domains, and MSTP parameters.

By default, STP is disabled on all interfaces. MSTP must be enabled on each physical or Ethernet Bundle interface to support Layer 2 redundancy and loop prevention. Once enabled on a parent interface, all its subinterfaces are MSTP-enabled automatically.

Perform these tasks to configure MSTP:

- Configure VLAN interfaces.
- Configure Layer 2 VPN bridge domains.
- Configure MSTP parameters.

Before you begin

Make sure that the target interfaces and VLAN IDs are available in your router configuration.

Procedure

1. Configure the VLAN interfaces.

   Example

```
Router# configure
Router(config)# interface HundredGigE0/0/0/2.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface HundredGigE0/0/0/3.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface HundredGigE0/0/0/14.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface HundredGigE0/0/0/2.1021 l2transport
Router(config-subif)# encapsulation dot1q 1021
Router(config)# interface HundredGigE0/0/0/3.1021 l2transport
Router(config-subif)# encapsulation dot1q 1021
Router(config)# interface HundredGigE0/0/0/14.1021 l2transport
Router(config-subif)# encapsulation dot1q 1021
Router(config-subif)# commit
```

2. Configure the Layer 2 VPN bridge domains.

   Example

```
Router# configure
Router(config)# l2vpn bridge group mstp
```

```
Router(config-l2vpn-bg)# bridge-domain mstp1001
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/2.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/3.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/14.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn-bg)# bridge-domain mstp1021
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/2.1021
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/3.1021
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/14.1021
Router(config-l2vpn-bg-bd-ac)# commit
```

3. Configure the MSTP parameters for the service.

Example

```
Router# configure
Router(config)# spanning-tree mst a
Router(config-mstp)# interface HundredGigE0/0/0/2
Router(config-mstp-if)# portfast bpduguard
Router(config-mstp-if)# commit
```

MSTP interface parameters

Describes MSTP interface parameter configuration, global and region settings, instance and cost parameters, protection features, enablement procedures, and includes verification tasks along with practical configuration examples.

The per-interface parameters are:

- External port path cost
- Hello Time
- Link Type
- Port Fast and BPDU Guard
- Root Guard and Topology Change Guard
- Port priority (per spanning-tree instance)
- Internal port path cost (per spanning-tree instance)

Per-interface configuration takes place in an interface submode within the MST configuration submode.

MSTP enablement

Lists interface types and MSTP configuration requirements.

MSTP does not operate on interfaces by default. These requirements apply:

- STP is disabled by default on all interfaces.

- MSTP must be explicitly enabled through configuration on each physical or Ethernet bundle interface.
- When MSTP is configured on a parent interface, all its subinterfaces are automatically MSTP-enabled.

Configurable MSTP global parameters

Lists the configurable MSTP global parameters that control region identity, timing behaviors, bridge modes, and instance-level VLAN mapping.

The MSTP standard defines a number of configurable global parameters.

- Region name and revision: Determines the MST region identity and tracks configuration changes.
- Bringup delay: Sets the delay before the protocol becomes active after configuration changes.
- Forward delay: Controls the delay before transitioning ports between learning and forwarding states.
- Max age or hops: Specifies the maximum time (or hop count) before removing spanning tree information.
- Transmit hold count: Limits the number of BPDUs transmitted in a given interval.
- Provider bridge mode: Adjusts protocol operation when running in provider bridge environments.
- Flush containment: Manages how topology changes are propagated and how MAC address flushing is contained.
- VLAN IDs (per spanning-tree instance): Maps VLANs to specific MST instances.
- Bridge Priority (per spanning-tree instance): Sets priority values for individual MST instances.

In most cases, the default values for these parameters are sufficient. Related tasks describe specific configuration steps for adjusting each parameter as needed.

MSTP interface parameters

Describes MSTP interface parameter configuration, global and region settings, instance and cost parameters, protection features, enablement procedures, and includes verification tasks along with practical configuration examples.

The per-interface parameters are:

- External port path cost
- Hello Time
- Link Type
- Port Fast and BPDU Guard
- Root Guard and Topology Change Guard
- Port priority (per spanning-tree instance)
- Internal port path cost (per spanning-tree instance)

Per-interface configuration takes place in an interface submode within the MST configuration submode.

Configure MSTP region parameters

Configure the MSTP region name, timers, and global protocol settings.

Use this task to enter MSTP configuration mode and set the global region and timer parameters.

Configuring MSTP region parameters ensures your network uses the desired region identity, revision number, and timer values for optimal spanning tree operation in environments requiring multiple spanning tree instances.

Before you begin

Make sure that you know the MST instance identifier, region name, revision, and timer values that you want to use.

Procedure

1. Enter MSTP configuration mode for the protocol instance.

   Example

   ```
   Router(config)# spanning-tree mst a
   ```

2. Set the bringup delay interval.

   Example

   ```
   Router(config-mstp)# bringup delay for 10 seconds
   ```

3. Disable flush containment if you want IEEE 802.1Q behavior.

   Example

   ```
   Router(config-mstp)# flush containment disable
   ```

4. Set the MST region name.

   Example

   ```
   Router(config-mstp)# name customer1
   ```

5. Set the MST region revision number.

   Example

   ```
   Router(config-mstp)# revision 1
   ```

6. Set the forward-delay timer.

   Example

   ```
   Router(config-mstp)# forward-delay 15
   ```

   Allowed values are 4 through 30.

7. Set the maximum age or hop count.

   Example

   ```
   Router(config-mstp)# maximum age 20
   ```

8. Set the transmit hold count.

Example

```
Router(config-mstp)# transmit hold-count 6
```

Configure MSTP instance parameters

Configure provider bridge mode and the MST instance priority and VLAN mapping.

Configure MSTP instance parameters, including provider bridge mode, MST instance priority, and VLAN range assignments.

Use this task when you need to set up Provider Bridge mode and assign MSTI-specific parameters for an MSTP deployment.

Procedure

1. Enable provider bridge.

   Example

   ```
   Router(config-mstp)# provider-bridge
   ```

2. Enter the MST instance submode.

   Example

   ```
   Router(config-mstp)# instance 101
   ```

   Allowed values are 0 through 4094.

3. Set the bridge priority for the instance.

   Example

   ```
   Router(config-mstp-inst)# priority 28672
   ```

4. Assign VLAN ranges to the instance.

   Example

   ```
   Router(config-mstp-inst)# vlan-id 1001-1005
   ```

Configure MSTP interface cost parameters

Configure the MSTP interface submode, per-instance port priority, path cost, external cost, and link type.

Enable control over MSTP instance behavior and path selection by adjusting port priority, internal and external costs, and interface type.

Configuring MSTP interface cost parameters allows you to influence Spanning Tree Protocol (STP) decisions for Layer 2 network topology, balancing redundancy and efficiency on MSTP-enabled routers.

Procedure

1. Enable MSTP-enabled interface.

   Example

   ```
   Router(config-mstp)# interface FastEthernet 0/0/0/1
   ```

   Enter the MSTP interface configuration submode and enable STP for the specified port.

2. Set the MSTI port priority.

   Example

   ```
   Router(config-mstp-if)# instance 101 port-priority 160
   ```

   Allowed priority values are 0 through 240 in multiples of 16.

3. Set the internal path cost for the instance.

   Example

   ```
   Router(config-mstp-if)# instance 101 cost 10000
   ```

   Repeat this step for each MSTI on each interface.

4. Set the external path cost on the port.

   Example

   ```
   Router(config-mstp-if)# external-cost 10000
   ```

5. Set the link type for the port.

   Example

   ```
   Router(config-mstp-if)# link-type point-to-point
   ```

Configure MSTP interface protection parameters

Configure the MSTP interface hello timer, Port Fast, BPDU Guard, Root Guard, Topology Change Guard, and commit behavior to enhance interface protection and ensure network stability.

Use this task to configure interface protection features and manage commit behavior for MSTP interfaces.

MSTP interface protection parameters help avoid network disruptions and unauthorized changes by enforcing timers and guards on each interface.

Procedure

1. Set the port hello timer.

Example

```
Router(config-mstp-if)# hello-time 1
```

Allowed values are 1 and 2.

2. Enable port fast, and optionally enable BPDU Guard.

Example

```
Router(config-mstp-if)# portfast
Router(config-mstp-if)# portfast bpduguard
```

3. Enable root guard on the port.

Example

```
Router(config-mstp-if)# guard root
```

4. Enable topology change guard on the port.

Example

```
Router(config-mstp-if)# guard topology-change
```

Repeat the interface-parameter tasks for each interface.

5. Save the configuration changes.

commit saves the configuration changes and remains within the configuration session. **end** prompts you to save, discard, or cancel before you leave configuration mode.

Verify MSTP

Verify MSTP configuration and operational status using show commands.

Verify MSTP configuration, interface state, and protocol events after setup.

Use these steps after configuring MSTP to ensure that the protocol is operating as expected. These verifications help you confirm the network topology, check for errors, and validate MSTP stability on your device.

Procedure

1. Use the **show spanning-tree mst** *mst-name* command to view the overall status and configuration for the specified MST instance.
2. Use the **show spanning-tree mst** *mst-name* **interface** *interface-name* command to check the spanning-tree state for a particular interface.
3. Use the **show spanning-tree mst** *mst-name* **errors** command to display any protocol errors associated with the MST instance.
4. Use the **show spanning-tree mst** *mst-name* **configuration** command to review the detailed MST configuration. .
5. Use the **show spanning-tree mst** *mst-name* **bpdu interface** *interface-name* command to inspect BPDU information for a specific interface in the MST instance. .

6. Use the **show spanning-tree mst** *mst-name* **topology-change flushes** command to display records of topology change notifications and MAC address flush events.

You can view the MSTP protocol state, interface status, configuration, BPDU information, errors, and records of topology changes. This helps confirm that MSTP is operating as intended and to identify any issues requiring attention.

MSTP configuration examples

Review the MSTP configuration and operational output examples for a single spanning-tree instance.

To review configuration and topology change output to confirm proper operation.

MSTP allows support for multiple spanning-tree instances within a network, helping optimize VLAN distribution and network reliability. This task walks you through configuring MSTP for a specific instance, viewing operational output, and checking topology changes.

Procedure

1. Review the MSTP configuration example for a single spanning-tree instance with MSTP enabled on a single interface.

   Example

```
config
spanning-tree mst customer1
name customer1
revision 1
instance 0
  priority 28672
!
instance 1
  vlan-ids 1001
  priority 28672
!
interface bundle-ether8171
!
interface bundle-ether861

interface Bundle-Ether861.10000 l2transport
encapsulation dot1q 1001
!
interface Bundle-Ether861.10001 l2transport
encapsulation dot1q 1002

interface Bundle-Ether8171.10000 l2transport
encapsulation dot1q 1001
!
interface Bundle-Ether8171.10001 l2transport
encapsulation dot1q 1002
```

2. Use the **show spanning-tree mst customer1** command to review the output.

   Example

```
show spanning-tree mst customer1
Role:  ROOT=Root, DSGN=Designated, ALT=Alternate, BKP=Backup, MSTR=Master
State: FWD=Forwarding, LRN=Learning, BLK=Blocked, DLY=Bringup Delayed
```

```
Operating in dot1q mode

MSTI 0 (CIST):

  VLANS Mapped: 1-1000,1006-4094

  CIST Root  Priority    24576
             Address     b026.80da.e800
             Ext Cost    0

  Root ID    Priority    24576
             Address     b026.80da.e800
             Int Cost    10000
             Max Age 20 sec, Forward Delay 15 sec

  Bridge ID  Priority    28672 (priority 28672 sys-id-ext 0)
             Address     00bc.6025.64d8
             Max Age 20 sec, Forward Delay 15 sec
             Max Hops 20, Transmit Hold count  6

Interface    Port ID           Role State Designated           Port ID
             Pri.Nbr Cost                 Bridge ID            Pri.Nbr
------------ ------- --------- ---- ----- -------------------- -------
BE8171       128.2   10000     ALT  BLK   28672 14a2.a05c.6600 128.1
BE861        128.1   10000     ROOT FWD   24576 b026.80da.e800 128.1

MSTI 1:

  VLANS Mapped: 1001-1005

  Root ID    Priority    24576
             Address     14a2.a05c.6600
             Int Cost    10000
             Max Age 20 sec, Forward Delay 15 sec

  Bridge ID  Priority    28672 (priority 28672 sys-id-ext 0)
             Address     00bc.6025.64d8
             Max Age 20 sec, Forward Delay 15 sec
             Max Hops 20, Transmit Hold count  6

Interface    Port ID           Role State Designated           Port ID
             Pri.Nbr Cost                 Bridge ID            Pri.Nbr
------------ ------- --------- ---- ----- -------------------- -------
BE8171       128.2   10000     ROOT FWD   24576 14a2.a05c.6600 128.1
BE861        128.1   10000     ALT  BLK   24576 b026.80da.e800 128.1
```

In this output, the first line indicates whether MSTP is operating in dot1q mode or in Provider Bridge mode, and this line is followed by details for each MSTI.

For each MSTI, the output shows the VLAN list for the instance, the root-bridge identity and path-cost information, the timer values that are received from the root bridge, and the local bridge identity and timer values.

The interface table shows the MSTP-enabled interfaces together with the interface name, port priority, port ID, port cost, port role, and port state.

The port role values are DSGN for designated, ROOT for root, ALT for alternate, BKP for backup, and MSTR for master. The port state values are BLK for blocked, LRN for learning, FWD for forwarding, and DLY for bringup delayed.

If the port is a boundary port and is not designated, only BOUNDARY PORT is displayed and the remaining information is not displayed. If the port is not up, or if the bringup delay timer is running, the remaining fields are not displayed.

3. Use the **spanning-tree mst customer1 topology-change flushes** to teview the topology-change flush output.

Example

```
spanning-tree mst customer1 topology-change flushes
STI 0 (CIST):
Interface     Last TC               Reason                           Count
------------  --------------------  -------------------------------- -----
BE8171        23:05:36 Jun  6 2023  Role change: ROOT to ALT             1
BE861         --------------------  No flushes                           0

MSTI 1:
Interface     Last TC               Reason                           Count
------------  --------------------  -------------------------------- -----
BE8171        --------------------  No flushes                           0
BE861         --------------------  Flush Containment active         -----
```

Per-VLAN Rapid Spanning Tree

Introduces Per-VLAN Rapid Spanning Tree concepts and explains PVRST behavior and characteristics for optimizing per-VLAN network convergence and redundancy.

A Per-VLAN rapid span tree (PVRST) is a spanning tree protocol implementation that

- runs a separate instance of the rapid spanning tree protocol (IEEE 802.1w) for each configured VLAN
- assigns a single root switch to each VLAN instance, and
- requires the feature to be configured on all VLANs for a particular port.

A single instance of spanning tree protocol (STP) operates on each configured VLAN unless manually disabled. Each Rapid PVST+ instance for a VLAN has its own root switch. When Rapid PVST+ is enabled, the feature must be configured for all VLANs present on a specific port.

Feature History Table

Feature Name	Release Information	Feature Description
Per-VLAN Rapid Spanning Tree	Release 26.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*); *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Per-VLAN Rapid Spanning Tree	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The PVRST functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
Per-VLAN Rapid Spanning Tree	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100]) (select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) (select variants only*) *The PVRST functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Per-VLAN Rapid Spanning Tree	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) Per-VLAN Rapid Spanning Tree (PVRST) allows faster convergence of the spanning tree protocol by running a separate instance of STP on each configured VLAN. By configuring PVRST on all VLANs for a specific port, you can take advantage of its rapid convergence and improve network performance and resiliency. *The PVRST functionality is now extended to routers with the 88-LC1-36EH line cards.

Characteristics of PVRST operation

Lists the defining characteristics, operational behaviors, and attribute limits of PVRST implementations for Ethernet networks.

- PVRST uses point-to-point wiring to provide rapid spanning tree convergence, achieving reconfiguration times of less than 1 second—compared to 50 seconds with default IEEE 802.1D STP.
- PVRST supports a single STP instance for each VLAN. PVRST supports up to 64 VLANs and 512 EFP instances per router.
- Designated or root ports send a Bridge Protocol Data Unit (BPDU) every two seconds by default. If three consecutive BPDUs are missed or the maximum age timer expires, the port immediately flushes all protocol information, facilitating rapid failure detection.
- PVRST transitions ports to the forwarding state rapidly only on edge ports and point-to-point links. Link types are automatically derived from the duplex mode: full-duplex implies point-to-point, half-duplex implies shared.
- Forward Delay and Max Age timers are configured globally, not per VLAN. The Hello timer can be set per port, but not per VLAN; this per-port setting applies to all VLANs on the port.
- The cost of a spanning tree bundle port is fixed at 10,000, regardless of the number or speed of bundle members, operational status, or changes in bundle membership.
- BPDU guard, when enabled on an interface, error-disables both the physical port and any associated Layer-2 or Layer-3 sub-interfaces upon receiving a BPDU.
- PVRST protection is available only for Ethernet Flow-points (EFPs) that are untagged or have a single VLAN tag.
- If any EFP in a bridge domain is protected by PVRST, all EFPs within that domain must belong to the same VLAN. If any EFP on a port is protected by PVRST, all EFPs on that port must be protected.

Information about Multiple Spanning Tree Protocol

Outlines fundamental MSTP information, including Spanning Tree Protocol overview, protocol operation, variants, MSTP region details and interaction, as well as MSTP Port Fast, Root Guard, and Topology Change Guard features.

To configure Ethernet services access lists, you must understand these concepts.

Spanning Tree Protocol loop prevention and redundancy

Provides details on how Spanning Tree Protocol (STP) prevents loops while preserving redundant Layer 2 paths in Ethernet networks.

Ethernet is no longer just a link-layer technology used to interconnect network devices and hosts. Its low cost, broad bandwidth options, and simple plug-and-play provisioning have transformed Ethernet into a robust solution for building networks, especially in access and aggregation regions within service provider environments.

Ethernet networks that lack a Time To Live (TTL) field in the Layer 2 header and promote multicast traffic can be vulnerable to broadcast storms if loops are present. However, network loops are often desirable because they provide redundant paths, enabling continued connectivity in the event of link failures.

Spanning Tree Protocol (STP) offers a loop-free topology within Ethernet networks while still supporting redundancy. Within networks that may contain loops, STP disables certain interfaces as needed to ensure there is a single path between any two devices. If a fault occurs on an active link, STP recalculates the spanning tree so all devices remain reachable. STP operates transparently to end stations, ensuring seamless network connectivity while preventing broadcast storms.

There are several variants of STP used in modern networks, each designed to address specific redundancy and loop prevention requirements.

STP protocol operation

This topic describes how STP exchanges BPDUs, selects the root bridge, chooses active paths, and blocks redundant ports.

All variants of STP operate in a similar fashion. STP frames, known as bridge protocol data units (BPDUs), are exchanged at regular intervals over Layer 2 LAN segments between network devices that participate in STP. These devices do not forward the frames, but use the information to construct a loop-free spanning tree.

The spanning tree is constructed by first selecting a device that becomes the root of the spanning tree, known as the root bridge, and then by determining a loop-free path from the root bridge to every other device in the network. Redundant paths are disabled by setting the appropriate ports into a blocked state, where STP frames can still be exchanged but data traffic is not forwarded. If a network segment fails and a redundant path exists, STP recalculates the spanning tree topology and activates the redundant path by unblocking the appropriate ports.

The root bridge is selected by the lowest Bridge ID, which is a combination of the configured bridge priority and the embedded MAC address of each device. The device with the lowest priority, or with equal lowest priority but the lowest MAC address, becomes the root bridge.

The root port is selected based on the lowest root path cost to the root bridge. If there is a tie in root path cost, the local switch selects the port that receives the BPDU with the lowest sender bridge ID as the root port.

The designated port is selected as the least-cost port on the local switch toward the root bridge. If there is a tie, the local switch selects the lowest-numbered port as the designated port.

The active path among redundant paths is determined primarily by the port path cost. The port path cost represents the cost of transiting between that port and the root bridge. If two paths from a LAN segment have the same cost, the selection is further determined by the lowest bridge ID of the attached devices and, when needed, by the configured port priority and port ID of the neighboring attached ports.

Once the active paths have been selected, any ports that do not form part of the active topology are moved to the blocking state.

STP variants

This topic lists the supported spanning tree protocol variants.

The supported variants of the Spanning Tree Protocol are:

- Legacy STP (STP): The original STP protocol was defined in IEEE 802.1D-1998. It creates a single spanning tree that is used for all VLANs and most convergence is timer-based.
- Multiple STP (MSTP): A further enhancement was defined in IEEE 802.1Q-2005. It allows multiple spanning tree instances to be created over the same physical topology. By assigning different VLANs to different spanning tree instances, data traffic can be load-balanced over different physical links. Multiple VLANs can be assigned to the same spanning tree instance, and the BPDUs that exchange MSTP information are always sent untagged because the VLAN and spanning tree instance data is encoded inside the BPDU.

Attributes of MSTP regions

Lists the configuration attributes that define MSTP regions and explains how regions are identified in MSTP.

MSTP introduces the concept of regions to support multiple spanning trees within a single network. An MSTP region is a group of devices under the same administrative control that share identical configurations.

The configuration attributes that define and identify an MSTP region are:

- Region name: The unique identifier assigned to the MSTP region.
- Revision number: The configuration version for the region.
- VLAN-to-instance mapping: The assignment of VLANs to specific spanning tree instances, which must be the same on all devices within the region.

A digest (hash) of these configuration attributes is included in every MSTP BPDU (Bridge Protocol Data Unit) sent by devices. Other devices in the network use this digest to verify region membership—if the digest matches, devices recognize each other as belonging to the same MSTP region.

How MSTP region communication works

Describes how an MSTP region uses the internal spanning tree when it communicates with non-MST bridges or devices in a different MSTP region.

For this example, switches SW1, SW2, SW3, and SW4 are MSTP-capable, while switches SW5 and SW6 are not. MSTP must present a unified logical topology at boundaries with non-MSTP devices and with other MSTP regions.

Summary

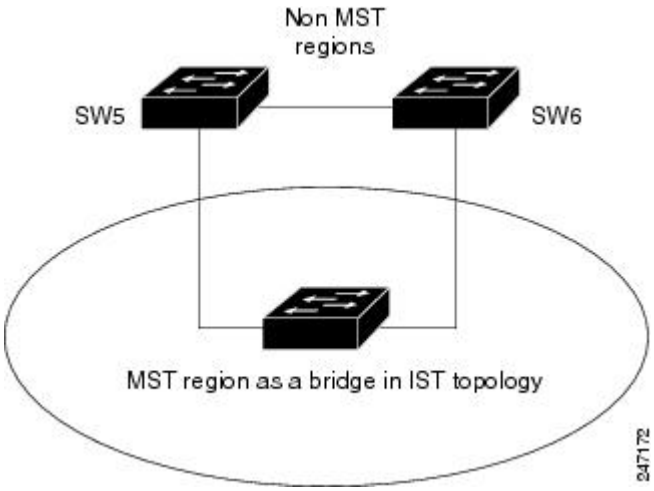
The key components involved in the process are:

- Internal spanning tree: Always instance 0, represents the region externally.
- MSTP region: Composed of MSTP-capable switches (e.g., SW1, SW2, SW3, SW4).
- Non-MSTP-aware devices: Devices not supporting MSTP, such as SW5 and SW6.

MSTP uses the internal spanning tree to represent the region externally, ensuring consistent topology and communication when interacting across region boundaries.

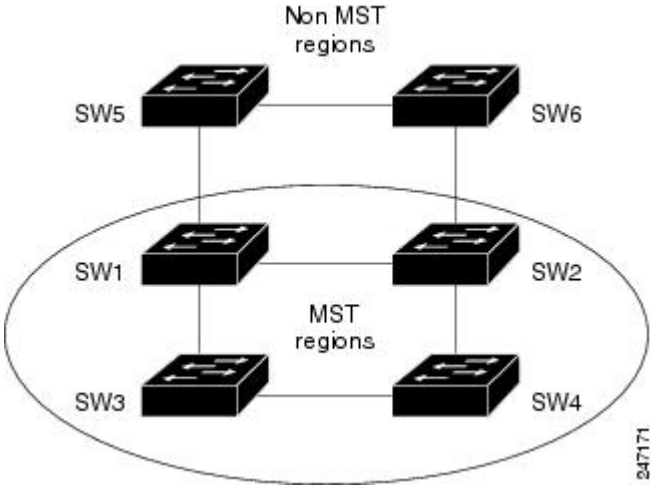
Workflow

Figure 32: MST interaction with non-MST regions



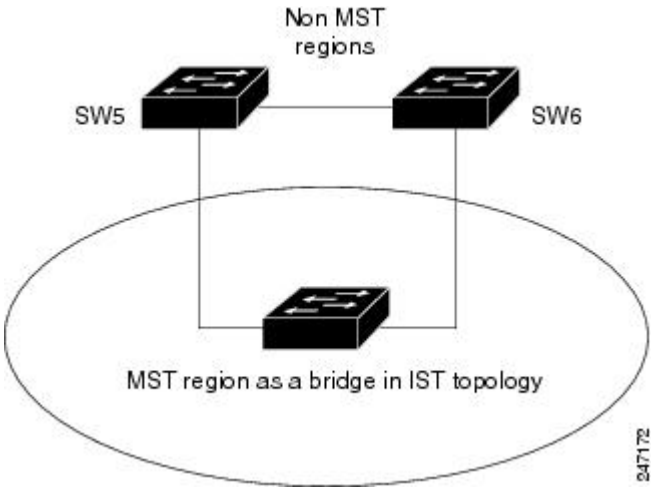
These stages describe how MSTP handles region-boundary communication.

1. When bridges running MSTP are connected to bridges running legacy STP or RSTP, the MSTP region must present a consistent topology to neighboring devices. MSTP uses the internal spanning tree (instance 0) for this purpose.



2. When communicating with non-MSTP-aware devices, the entire MSTP region is represented externally as a single switch. The logical topology is based on the internal spanning tree view.

Figure 33: Logical topology in MST region interacting with non-MST bridges



The logical topology is therefore based on the internal spanning tree view of the region.

3. The same mechanism applies when the neighboring devices belong to a different MSTP region. For example, SW5 represents MSTP devices located in a different region than SW1-SW4.

Result

At the region boundary, the MSTP region appears as a single logical switch, and communication between regions—whether to non-MSTP devices or other MSTP regions—relies on the internal spanning tree model, ensuring a stable and consistent topology.

MSTP port fast

Provides the edge-port behavior and configuration notes for MSTP port fast.

MSTP port fast handles ports at the edge of a switched Ethernet network. The following reference information summarizes its behavior and configuration notes:

- For devices with a single network link (typically host devices), MSTP does not need to run, as only one path exists and topology changes are unnecessary.
- Port fast prevents unnecessary MSTP participation and topology changes, avoiding MAC flushes when a single link fails or is restored.
- By default, MSTP monitors ports where no BPDUs are received; after a timeout, it places them into edge mode so they do not participate in MSTP.
- Explicitly configuring edge ports as port fast speeds up the process and improves convergence.
- Configuration changes require you to disable and then re-enable the port using the commands **shutdown** and **no shutdown** in interface configuration mode.
- Port fast is a Cisco-proprietary extension for legacy STP, but in MSTP standards it is known as Edge Port.

MSTP Root Guard

Provides details on how MSTP Root Guard enforces control over root bridge selection and blocks conflicting information.

MSTP Root Guard is a mechanism that allows administrators to enforce and secure the location of the root bridge in a spanning tree network. This feature is particularly useful in networks with shared administrative control, where it is essential to maintain an optimal network topology by positioning the root bridge at a specific location, often the center of the network.

Key aspects of MSTP Root Guard include:

- Preventing designated interfaces from becoming root ports if superior BPDU information is received, instead placing those interfaces in a blocking state to avoid undesired topology changes.
- Supporting administrator efforts to set the root bridge priority to the lowest possible value while providing additional protection when competing devices may also use this strategy.
- Ensuring that a switch with Root Guard on all interfaces will become the root bridge for the spanning tree, as any conflicting information arriving on those ports causes them to be blocked.

Additional information

- Root Guard originally appeared as a Cisco-proprietary extension in legacy STP implementations but is now standardized in MSTP as the "Restricted Role."
- The root bridge itself has no root ports; Root Guard helps maintain this by ensuring a device cannot be displaced as root due to unexpected superior BPDUs.

MSTP topology change guard

Provides the key features and behaviors of MSTP topology change guard.

MSTP topology change guard prevents topology changes that originate at or are received on a specific port from being propagated through the rest of the network.

Key features and behaviors:

- Helps prevent MAC address table flushing in the network core by blocking external or unauthorized topology changes.
- Is useful when the network is not under single administrative control, to prevent devices outside the network core from affecting core stability.
- Can be enabled by configuring topology change guard on individual ports.
- Is referred to as "Restricted TCN" in the MSTP standard.

13 Ethernet Service Activation Testing with Y.1564

Topics:

- Y.1564 Ethernet service activation test
- Bandwidth parameters for Y.1564 service activation test
- Service activation test targets for Y.1564 support
- Start or stop a service activation test on an interface

Outlines the use of Y.1564 for Ethernet service activation testing, including methodology concepts, SLA assurance, key performance indicators, supported bandwidth and encapsulation, test targets, operational guidelines, and procedures for test execution on interfaces.

Y.1564 Ethernet service activation test

Introduces Y.1564 Ethernet service activation testing, covering SLA assurance methods, key performance indicators, test objectives, supported encapsulation modes, testing restrictions, and guidance for effective SAT and SADT usage.

A Y.1564 Ethernet service activation test is a standardized methodology that

- validates Service Level Agreements (SLAs) for Ethernet-based services
- measures key performance metrics such as delay, jitter, loss, and throughput, and
- ensures consistent service quality and network performance in both User Network Interface (UNI) and Network-to-Network Interface (NNI) deployments.

Feature history
Table 47: Feature History Table

Feature Name	Release Information	Feature Description
Y.1564 Ethernet service activation test	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8400 [ASIC: K100]). Use the Y.1564 Ethernet Service Activation Test (SAT) to check SLA settings. Ensure all services meet SLA objectives at their maximum committed rate under maximum load. Conduct medium-term and long-term soak tests under stress. Use this procedure to turn up service, install, and troubleshoot Ethernet-based services. You can use this test to collect Key Performance Indicator (KPI) metrics, including Frame Transfer Delay (latency or Round Trip Time (RTT)) and Frame Loss Ratio (packet loss). Use the test to measure the non-drop rate and the maximum rate under stress conditions.

Ethernet service activation tests for SLA assurance

Provides standardized test methods (ITU-T Y.1564) for assessing SLA compliance of Ethernet services in provider networks.

The ITU-T Y.1564 Ethernet service activation tests deliver a common approach for carriers and service providers to evaluate Ethernet-based services. As Ethernet deployment expands across provider networks, these tests:

- allow for SLA verification even under stress conditions and at full committed data rates,
- support effective network troubleshooting and service activation.

Key performance indicators for Y.1564

Lists the key performance indicators used by Y.1564 tests—Frame Transfer Delay and Frame Loss Ratio—to verify that Ethernet provider SLAs are met, even under stress and full committed rates.

SLA assurance for provider Ethernet networks using the Y.1564 standardized test is based on a common method that assesses Ethernet-based services for carriers and service providers. As Ethernet usage increases across provider networks, standardized testing such as Y.1564 ensures services fully satisfy SLAs, including under high load.

Y.1564 tests collect these KPIs to verify that configured SLAs are met:

- Frame Transfer Delay (FTD) or latency: Measures the round-trip time taken by a test frame across the network.
- Frame Loss Ratio (FLR): Measures the number of lost test packets relative to packets sent, providing a direct indication of network reliability.

> **Note**
>
> Jitter is not directly measured in Cisco Y.1564 implementations.

SLA validation objectives of Y.1564 tests

Lists the primary objectives Y.1564 tests address for SLA validation, committed rate behavior, and service quality under network load.

Y.1564 tests are used to:

- Validate network service performance against SLA standards within a controlled test time.
- Confirm that all services meet committed rate objectives and perform reliably at maximum committed rates.
- Provide medium-term and long-term testing to evaluate service quality and stability under sustained network load and stress conditions.

Additional information

The primary use case for Service Activation and Data Traffic (SADT) testing with Y.1564 focuses on identifying the "non-drop rate" (the packet transmission rate below which no loss occurs) and the maximum achieved rate under stress. Traditionally, packet count has not been a decisive factor in these tests.

Supported encapsulation modes for Y.1564 service activation and deactivation tests

Lists the encapsulation modes supported by Y.1564 SAT and SADT for service activation testing, including dot1q, dot1ad, priority-tagged, untagged, and default modes.

The Y.1564 Service Activation and Deactivation Test (SAT and SADT) feature supports multiple encapsulation modes used to establish a baseline for Ethernet service behavior during service activation testing. The supported encapsulation modes are:

- dot1q
- dot1q + second dot1q
- dot1ad
- dot1ad + second dot1q
- priority tagged
- untagged
- default

In two-way statistics collection mode, you send test traffic and the remote node loops it back, enabling local measurement and statistics collection.

To learn more about default encapsulation in Layer 2 VPNs, refer to *Virtual LANs in Layer 2 VPNs* in the *L2VPN Configuration Guide for Cisco 8000 Series Routers*.

Restrictions for Y.1564 service activation test

Outlines the platform and interoperability restrictions you must follow before running a Y.1564 service activation test (SAT) or service activation and diagnostics test (SADT) session.

You must follow these restrictions for Y.1564 SAT and SADT sessions to ensure proper functionality and compatibility:

- Run SAT and SADT only within the supported interface, packet, and interoperability limits.
- Do not use color aware mode because it is not supported.
- Do not rely on SAT egress Ethernet ACL classification by VID, DEI, and CoS fields to drop packets based on VLAN and class-of-service matches.
- Do not run SAT tests on interfaces with Layer 2 ACLs, and do not configure Layer 2 ACLs on interfaces where SAT tests have already run.
- Restart SAT sessions after Route Processor failover because SAT ACL sessions fail and the sessions do not persist across switchover.
- Do not use a multicast destination MAC address. Do not use interfaces that rely on VLAN rewrite.
- Configure the Layer 2 interface under Layer 2 services before you run SAT.
- Use a largest packet size of 4096 bytes when you run SAT or Ethernet mix tests.
- Expect up to one percent frame loss for 64-byte and 128-byte packets on K100-based Silicon One ASICs.

Usage guidelines for SAT and SADT testing

Outlines best practices to ensure SAT traffic matches the intended subinterface by sequencing tagged and default sessions when using default encapsulation.

When running SAT or SADT tests with default, untagged, and tagged encapsulation combinations, follow these best practices:

- Use default encapsulation only when the interface design and session sequencing keep SAT traffic on the intended subinterface.
- Use default encapsulation only without a Class of Service value because CoS is not supported with default encapsulation.
- Initiate only one session at a time with default encapsulation because the SAT engine uses only the CoS value to differentiate sessions and default encapsulation packets do not include VLAN priority.
- If your device has both untagged and default subinterfaces, run SAT sessions on the untagged subinterface. Packets sent using default encapsulation are processed by the untagged subinterface and the session will not function.
- If both tagged and default subinterfaces exist, initiate the session over the tagged interface before you start the default session so both sessions function correctly.
- Start the session on the tagged interface before you begin the default session to prevent frame loss for tagged packets that can be matched by the default PMF entry because of a missing VLAN priority.

Bandwidth parameters for Y.1564 service activation test

Details the range of bandwidth parameters supported for Ethernet service activation testing with Y.1564, illustrating limits and configuration considerations for verifying service performance.

This table summarizes the bandwidth parameters supported for the Y.1564 service activation test:

Table 48: Bandwidth parameters

Bandwidth parameters	Internal direction	External direction
Committed information rate	Y	Y
Exceeded information rate	Y	Y

Service activation test targets for Y.1564 support

Explains the service activation test target matrix, specifying interface and service combinations that can be validated using Y.1564 testing for comprehensive coverage.

Use this table to determine whether a specific interface or service target supports Y.1564 service activation tests in the internal and external directions.

Table 49: Service activation test target matrix

Target	Internal direction	External direction
L2 Interface over physical main/sub interfaces	Y	Y
L2 Interface over bundle main/sub interfaces	N	N
L2 PW VPWS over physical main/sub interfaces	Y	Y
L2 PW VPWS over bundle main/sub interfaces	N	N
L2 EVPN/XConnect over physical main/sub interfaces	Y	Y
L2 EVPN/XConnect over bundle main/sub interfaces	N	N
L2 VPLS PW	N	N
L2 EVPN Bridge-Domain	N	N
L3 Interfaces	N	N

Start or stop a service activation test on an interface

Provides instructions to initiate or terminate a service activation test on an interface, guiding users through Y.1564 test execution steps for operational management.

Use this task to start, stop, and verify Y.1564 service activation tests on network interfaces.

Service activation tests use the Y.1564 methodology to validate SLA compliance and performance parameters, such as throughput and packet loss, on network interfaces. These tests help ensure that new or existing services meet operational requirements before deployment or after configuration changes. You can initiate tests in internal or external directions and stop them individually or collectively as needed.

Procedure

1. Start a service activation test on an interface with external direction.

   Example

   ```
   Router# ethernet service-activation-test start interface HundredGigE0/0/0/0.100
    profile SAT-8K destination 11:22:33:44:55:66 direction external
   ```

2. Start a service activation test on an interface with internal direction.

   Example

   ```
   Router# ethernet service-activation-test start interface HundredGigE0/0/0/0.100
    profile SAT-8K destination 11:22:33:44:55:66 direction internal
   ```

3. Stop a service activation test on an interface.

   Example

   ```
   Router# ethernet service-activation-test stop interface HundredGigE0/0/0/0.100
    profile SAT-8K destination 11:22:33:44:55:66 direction external
   ```

4. Stop all service activation tests.

   Example

   ```
   Router# ethernet service-activation-test stop all
   ```

5. Use the **show ethernet service-activation-test interface** command in privileged EXEC mode to verify Y.1564 service activation test interfaces.

   Example

   ```
   Router# show ethernet service-activation-test interface
               HundredGigE0/0/0/0.100
               Interface HundredGigE0/0/0/0.100
               Service activation tests permitted
               Test completed:
               Duration 10 minute(s)
               Information rate 750 Mbps
               Color-blind
               Internal, Two-way, Destination 00:01:00:02:00:03
               Packet size EMIX, Sequence 'abceg', Pattern hex 0x00
               Outer CoS 5
               Results:
               Step 1, Information Rate 750 Mbps
               CIR packets:
               Tx packets: 94286350, bytes: 0
               Rx packets: 94286350, bytes: 38476876568
   ```

```
            FL: 0, FLR: 0%
            FD: Not supported
            IFDV: Not supported
            Out of order packets: 0 (0%)
            Error packets: 0 (0%)
            EIR packets:
            Tx packets: 0, bytes: 0
            Rx packets: 0, bytes: 0
            FL: 0, FLR: 0%
            FD: Not Supported
            IFDV: Not Supported
            Out of order packets: 0 (0%)
            Error packets: 0 (0%)
```

The command output shows which interfaces have Y.1564 enabled and your permission status. You can view test status and detailed results including packet statistics and error counts.