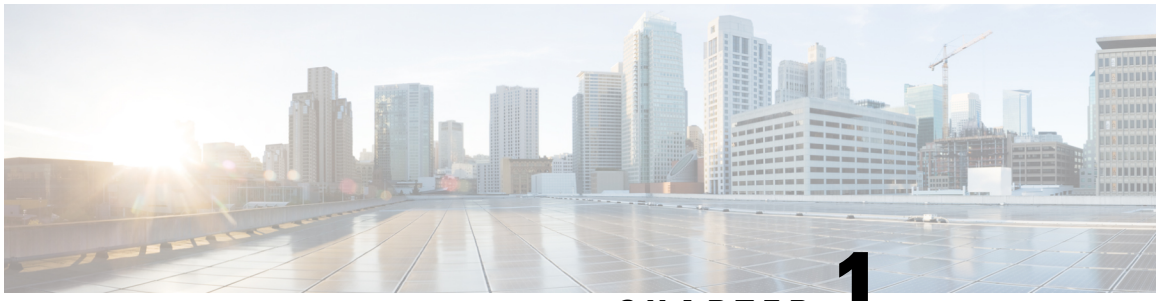
EVPN Configuration Guide for Cisco 8000 Series Routers, Cisco IOS XR Releases

First Published: 2025-10-31

Last Modified: 2026-06-10

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

CONTENTS



CHAPTER 1

YANG Data Models for EVPN Features

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

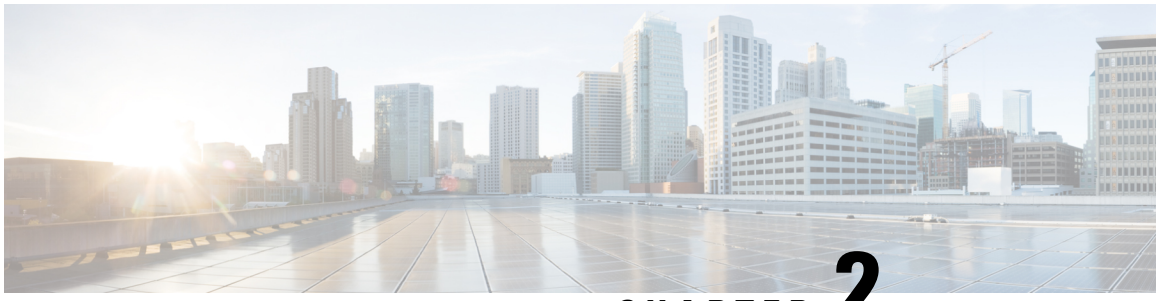
Using YANG Data Models

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*.



CHAPTER 2

Getting Started with EVPN MPLS

EVPN MPLS

An Ethernet VPN (EVPN) is a networking technology that

- offers scalable and simplified Layer 2 and Layer 3 VPN services
- enables control-plane-based MAC learning using the MP-BGP protocol, and
- supports flexible data-plane encapsulations while maintaining a unified control plane.

Multiprotocol Border Gateway Protocol (MP-BGP) is a routing protocol designed to exchange routing and reachability information across multiple network layer protocols, enabling versatile and scalable network communication.

Ethernet architecture distinctly separates the control plane from the data plane, which significantly improves network scalability and simplifies operational management. EVPN supports efficient traffic load balancing and facilitates various service models, including E-LAN, E-LINE, and E-TREE, to meet diverse networking requirements.

Need for EVPN

Today, our networks have different protocols serving different purposes, which makes daily operations more complex than they need to be. This hinders the ability to deliver end-to-end services with speed and agility. As you deploy multiple geographically disparate data centers, they're looking for scalable and simplified network solutions to extend virtualization and cluster domains between multiple data centers.

The evolution of EVPN started due to the need for a scalable solution to bridge various layer 2 domains and overcome the limitations faced by VPLS, such as scalability, multihoming, and per-flow load balancing.

EVPN addresses the shortcomings of VPLS, including scalability, multihoming, and per-flow load balancing. EVPN provides secure and private connectivity for multiple sites within an organization spread across different geographical locations. EVPN uses MAC addresses as routable addresses and distributes them to all participating PEs through the MP-BGP EVPN control plane.

Note Starting with the Cisco IOS XR Release 24.3.1, line cards and routers equipped with the Q100, Q200, and P100-based Silicon One ASICs support EVPN MPLS.

To know more about EVPN, see https://e-vpn.io.

Benefits of EVPN

These are the key benefits of EVPN:

- Per flow-based load balancing—enables more granular load balancing of traffic across available paths.
- Scalability—offers a more scalable solution compared to previous L2VPN technologies.
- Reduced operational complexity—simplifies network operations by providing a unified approach for L2 and L3 VPN services.
- Improved network efficiency by eliminating flooding and learning—reduces or eliminates the need for flooding and learning in the data plane.
- Provides fast reroute, resiliency, fast reconvergence during link failure—offers mechanisms for fast reroute and reconvergence in case of link failures.
- Integrates L2 and L3 VPN services—integrates both Layer 2 and Layer 3 VPN services.

Key concepts of EVPN

Here are the key concepts related to EVPN fundamentals, including Ethernet Segments, EVPN instances, and associated mechanisms for traffic management and redundancy:

- Ethernet segment (ES)—is a set of Ethernet links that connects a multihomed device or network to two or more PEs, and uses Ethernet segment route (route type 4) for designated forwarder (DF) election for BUM traffic.
- Ethernet segment identifiers (ESI)—are unique non-zero identifiers that are assigned to Ethernet segments, and represent each Ethernet segment uniquely across the network.
- EVPN instances (EVI)—is a virtual network identifier (VNI) on a PE router that acts like a VPN Routing and Forwarding (VRF), uses Route Targets (RTs) for import and export policies, maps traffic from ports or VLANs to a Bridge Domain (BD), and forwards it to the MPLS core.
- Ethernet auto discovery routes per ES (EAD-ES)—is also referred to as route type 1 that is used to converge traffic faster during access failure scenarios, and has an Ethernet Tag of 0xFFFFFFFF.
- Ethernet auto discovery routes per EVI (EAD-EVI)— is also referred to as route type 1 that is used for aliasing and load balancing when traffic only hashes to one of the switches, and cannot have an Ethernet tag value of 0xFFFFFFFF to differentiate it from the EAD-ES route.
- Aliasing—is a process that is used for load balancing traffic to all connected switches for a given Ethernet segment using the route type 1 EAD-EVI route, and is performed irrespective of the switch where the hosts are actually learned.

- Mass withdrawal—is a process that is used for fast convergence during access failure scenarios using the route type 1 EAD-ES route.
- DF election—is a process that is used to prevent forwarding loops, and allows only a single router to decapsulate and forward traffic for a given Ethernet Segment.
- VPN membership—is the process where PEs exchange EVPN routes at startup to discover remote PE members of an EVI, which enables building flood lists for multicast ingress replication and exchanging BUM and unicast labels for MAC address learning.
- Ethernet segment reachability—is a process where, in multihoming scenarios, the PE auto-discovers remote PE and the corresponding redundancy mode (all-active or single-active), which allows PEs to withdraw the routes used at this stage in case of segment failures in order to trigger fast convergence by signaling a MAC mass withdrawal on remote PEs.
- Redundancy group membership—is a process where PEs connected to the same Ethernet segment (multihoming) automatically discover each other and elect a Designated Forwarder (DF) that is responsible for forwarding Broadcast, Unknown unicast, and Multicast (BUM) traffic for a given EVI.

EVPN route types

EVPN supports EVPN route types that define several route types that serve different purposes in delivering Layer 2 and Layer 3 services over an IP-MPLS network. Each route type carries specific information.

Route Type	Description	Usage
Route Type 1: Ethernet Auto-Discovery (AD) Route	The Ethernet Auto-Discovery (AD) routes are advertised on per EVI and per ESI basis. These routes are sent per ES. They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed. This route type is used for mass withdrawal of MAC addresses and aliasing for load balancing.	Few routes are sent per ES, carries the list of EVIs that belong to ES.
Route Type 2: MAC-IP Advertisement Route	These routes are per-VLAN routes, so only PEs that are part of a VNI require these routes. The host's IP and MAC addresses are advertised to the peers within NRLI. The control plane learning of MAC addresses reduces unknown unicast flooding.	Advertises MAC address reachability and IP-MAC binding.
Route Type 3: Inclusive Multicast Ethernet Tag Route	This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.	Discovers multicast tunnel end point.

Route Type	Description	Usage
Route Type 4: Ethernet Segment Route	Ethernet segment routes enable to connect a CE device to two or PE devices. ES route enables the discovery of connected PE devices that are connected to the same Ethernet Segment.	Discovers redundancy group and DF election.
Route Type 5: IP Prefix Route	The IP prefixes are advertised independently of the MAC-advertised routes. With EVPN IRB, host route /32 is advertised using RT-2 and subnet /24 is advertised using RT-5.	Advertises IP prefixes.

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

EVPN modes

EVPN modes address modern network needs like scalability, multi-tenancy, redundancy, and workload mobility, providing a unified solution for Layer 2 and Layer 3 services. These modes provide an efficient solution for Layer 2 and Layer 3 services, ensuring operational simplicity, high performance, and flexibility for data centers, service providers, and cloud environments.

These EVPN modes are supported:

- Single-homing—enables you to connect a customer edge (CE) device to one provider edge (PE) device.
- Multihoming—enables you to connect a customer edge (CE) device to more than one provider edge (PE) device. Multihoming ensures redundant connectivity, so that there is no traffic disruption when there is a network failure. These multihoming modes are supported:
  - All-active
  - Single-active
  - Single-flow active
  - Port-active

How EVPN works

Summary

The key components involved in the process are:

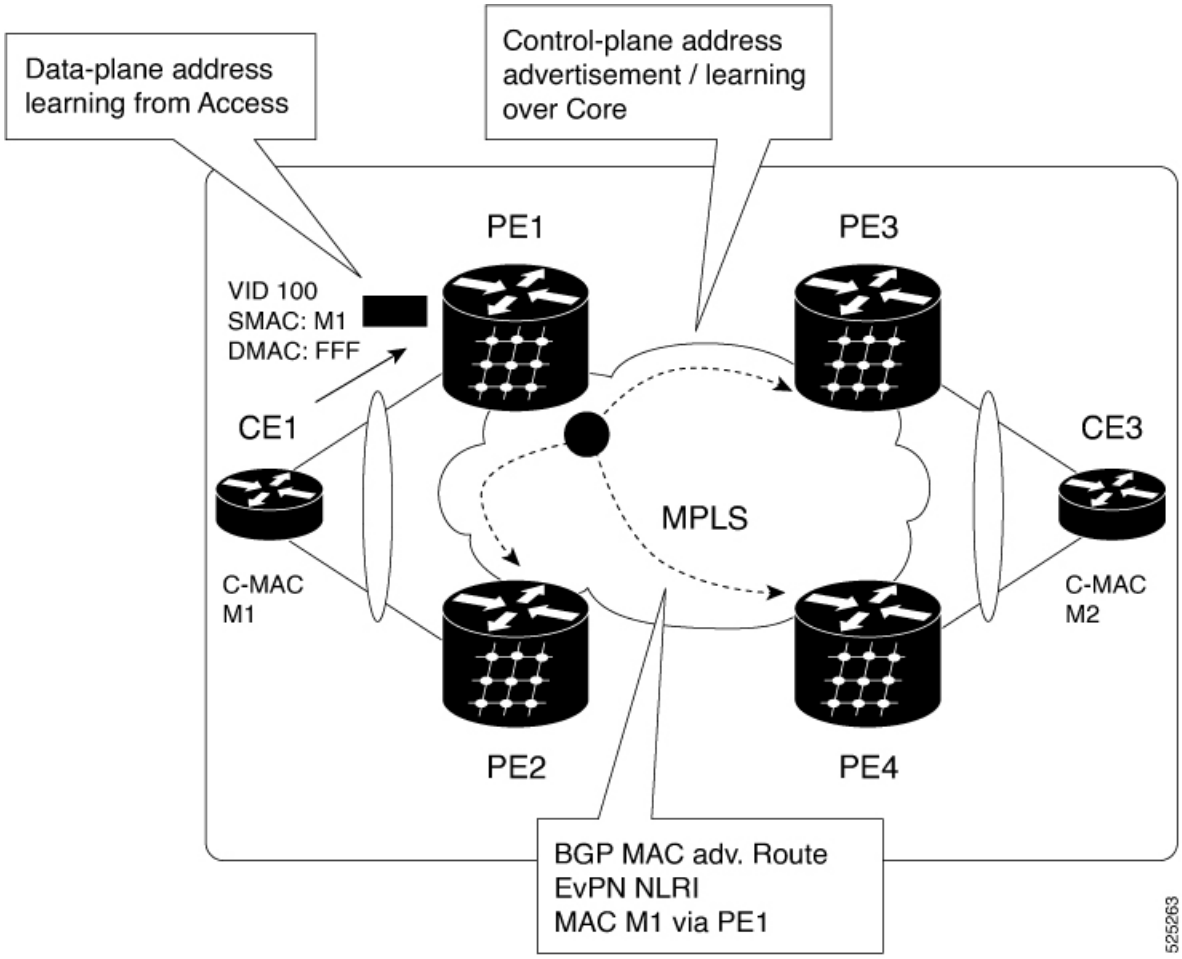
- VPN membership—PEs exchange EVPN routes to discover remote PE members of an EVI, build flood lists for multicast ingress replication, and exchange BUM and unicast labels during MAC learning.

- Ethernet segment reachability—PEs perform auto-discovery of remote PEs and determine redundancy modes in multihoming scenarios. During segment failures, routes are withdrawn to trigger fast convergence via MAC mass withdrawal.
- Redundancy group membership—PEs discover and elect a Designated Forwarder (DF) for forwarding BUM traffic in multihoming scenarios.

Note Starting with the Cisco IOS XR Release 24.2.1, the unknown unicast suppression (**unknown-unicast-suppression**) operation under a given EVI is not supported on the Cisco 8000 series routers.

Workflow

Figure 1: How EVPN works



The EVPN operation involves these stages:

1. Startup—PEs exchange EVPN routes for VPN membership, segment reachability, and redundancy group discovery.

2. MAC Advertisement—unknown unicast or BUM MAC addresses are advertised as route type 2.
3. Multihoming—route types 1, 3, and 4 are exchanged to discover redundancy modes (single-active or all-active) and elect a DF.

EVPN can be set up to operate in either single-homing or multihoming modes, providing various levels of redundancy and connectivity in network configurations.

Single-homing mode stages:

The EVPN single-homing mode involves these stages:

1. Route type 3 advertisement—PEs advertise route type 3 upon enabling EVPN.
2. Discovery—PEs discover all other member PEs within the EVPN instance.
3. BUM MAC handling—unknown unicast or BUM MACs are advertised as route type 2.
4. MAC route distribution—route type 2 advertises MAC routes to other PEs.

Multihoming mode stages:

The EVPN multihoming mode involves these stages:

1. Route types 1, 3, and 4 advertisement—PEs advertise these routes to discover redundancy modes and elect a DF
   a. Route type 1 usage—automatically discovers PEs hosting the same CE and enables fast rerouting during link failures.
   b. Route type 4 usage—facilitates DF election for BUM traffic.
2. Customer traffic handling—distributes MAC reachability information through route type 2.
3. Traffic delivery—remote PEs use MPLS labels to deliver traffic to the specified MAC address.

This process ensures efficient and reliable EVPN operation in both single-homing and multihoming scenarios, enabling robust connectivity, redundancy, and seamless traffic handling for customer networks.

Behavior change due to ESI label assignment

To adhere to RFC 7432 recommendations, the encoding or decoding of MPLS label is modified for extended community. Earlier, the lower 20 bits of extended community were used to encode the split-horizon group (SHG) label. The SHG label encoding now uses a higher 20 bits of extended community.

According to this change, routers running old and new software release versions in the same ethernet segment decodeextended communities differently. This change causes inconsistent SHG labels on peering EVPN PE routers. Almost always, the router drops BUM packets with incorrect SHG labels. However, remote PE may accept such packets and forward them to CE in certain conditions, potentially causing a loop. One such instance is when the label incorrectly reads NULL.

To overcome this problem, Cisco recommends you to:

- Minimize the time both PEs are running different software release versions.
- Before upgrading to a new release, isolate the upgraded node and shut down the corresponding AC bundle.

- After upgrading both the PEs to the same release, you can bring both into service.

Similar recommendations apply to peering PEs with different vendors with SHG label assignment that does not adhere to RFC 7432.

EVPN timers

EVPN timers are parameters within EVPN that control the timing of route propagation and convergence events.

EVPN timers ensure efficient and reliable exchange of MAC and IP information by controlling route propagation and convergence. Their significance lies in optimizing network stability, reducing downtime, and improving responsiveness during topology changes.

This table shows various EVPN timers:

Table 1: EVPN timers

Timer	Range	Default Value	Trigger	Applicability	Action	Sequence
startup-cost-in	30-86400	disabled	node recovered*	Single-Homed, All-Active, Single-Active	Postpone EVPN startup procedure and Hold AC link(s) down to prevent CE to PE forwarding. Startup-cost-in timer allows PE to set core protocols first.	1
recovery	20-3600s	30s	node recovered, interface recovered **	Single-Homed***, Single-Active	Postpone EVPN Startup procedure. Recovery timer allows PE to set access protocols (STP) before reachability towards EVPN core is advertised.	2

Timer	Range	Default Value	Trigger	Applicability	Action	Sequence
peering	0-3600s	3s	node recovered, interface recovered	All-Active, Single-Active	Starts after sending EVPN RT4 to postpone rest of EVPN startup procedure. Peering timer allows remote PE (multihoming AC with same ESI) to process RT4 before DF election will happen.	3

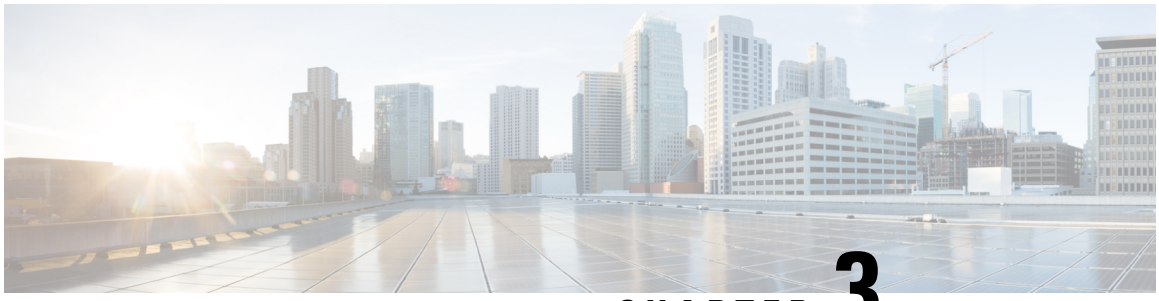
Note

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

* indicates all required software components are loaded.

** indicates link status is up.

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



CHAPTER 3

EVPN MPLS Single-Homing

EVPN MPLS single-homing mode

EVPN MPLS single-homing is a network connectivity mode that

- enables a single connection between a customer edge (CE) device and a provider edge (PE) device using MPLS as the transport protocol
- simplifies network design by eliminating the need for redundant connections at the CE level, and
- provides efficient Layer 2 and Layer 3 connectivity over an MPLS backbone.

EVPN MPLS single-homing is a cost-effective and simplified network solution designed for scenarios prioritizing cost efficiency and simplicity over redundancy. This mode is commonly used in environments where the customer edge (CE) device does not require a backup connection to the provider edge (PE) device, and is ideal for non-critical applications, such as small enterprises connecting branch offices to an MPLS network, as it reduces costs, simplifies deployment, and minimizes operational complexity.

EVPN control plane using BGP distributes MAC address information learned from the single-homed CE to other PEs, enabling Layer 2 bridging across the MPLS network.

EVPN MPLS single-homing mode services

The EVPN MPLS single-homing mode supports various service types to provide diverse network connectivity options. These services are designed to replace legacy technologies, offering enhanced flexibility, scalability, and fault tolerance. Both services utilize the EVPN control plane for scalability, redundancy, and efficient MAC address learning.

The EVPN MPLS single-homing mode supports these services:

- EVPN E-LAN—EVPN E-LAN in EVPN MPLS single-homing offers multipoint-to-multipoint Layer 2 connectivity, enabling seamless communication among multiple sites as if they were on the same LAN. This solution is particularly suitable for scenarios requiring interconnectivity among more than two locations, such as data center interconnects or corporate WANs. For example, a company with offices in New York, London, and Tokyo can utilize EVPN E-LAN to interconnect these offices, allowing all sites to communicate as part of the same Ethernet network.

- EVPN E-Line—EVPN E-Line in EVPN MPLS single-homing provides point-to-point Layer 2 connectivity between two endpoints, such as customer sites or data centers, making it an ideal solution for scenarios where direct communication is required over a service provider's MPLS or IP backbone. For instance, a business with offices in two cities can leverage EVPN E-Line to establish a direct, transparent Ethernet connection between the sites, ensuring seamless and efficient communication.

This table compares the key features and use cases of EVPN E-LAN and EVPN E-Line services.

Table 2: EVPN E-LAN and EVPN E-Line services

Feature	EVPN E-LAN	EVPN E-Line
Connectivity type	Multipoint-to-Multipoint	Point-to-Point
Typical use cases	Multi-site connectivity, interconnecting multiple locations	Data center interconnects, connecting two sites
MAC address learning	Across all endpoints in the E-LAN	Limited to two endpoints

EVPN E-LAN single-homing

A EVPN E-LAN single-homing is a network connectivity model that

- simplifies network infrastructure by connecting a Layer 2 gateway device to a single access network
- reduces costs by eliminating the need for additional infrastructure, and
- supports seamless communication between customer sites over an MPLS network.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-LAN L2 Gateway Single-Homing	Release 26.2.1	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 E-LAN L2 Gateway Single-Homing	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN E-LAN L2 Gateway Single-Homing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
EVPN E-LAN L2 Gateway Single-Homing	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The EVPN E-LAN single-homing functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN E-LAN L2 Gateway Single-Homing	Release 24.3.1	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*) *The EVPN E-LAN single-homing 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
EVPN E-LAN L2 Gateway Single-Homing	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The EVPN E-LAN single-homing functionality is now extended to the Cisco 88-LC1-36EH routers.
EVPN E-LAN L2 Gateway Single-Homing	Release 7.11.1	We now offer a cost-effective and simplified solution for seamless communication between various customer sites connected to the same service provider network using Ethernet Virtual Private Network (EVPN) single-homing mode. EVPN LAN (E-LAN) is a service to bridge Ethernet data traffic among different sites across the MPLS network connecting a Layer 2 gateway device to a single access network. In the single-homing mode, a device is connected to one router in the MPLS core through physical ports or bundle ports, and in the event of a failure on those links, the traffic over the links is not protected by links to another router on the core. This feature is supported only on Q200-based line cards. The feature introduces the **evpn** commands.

EVPN E-LAN single-homing for scalable Layer 2 connectivity

The EVPN E-LAN single-homing delivers seamless multipoint-to-multipoint Layer 2 connectivity, enabling efficient communication between multiple endpoints such as customer sites or data centers. Designed for scenarios requiring transparent Ethernet connectivity over a service provider's MPLS or IP backbone, this solution ensures reliable and scalable data exchange, making it an ideal choice for businesses aiming to interconnect multiple locations with ease.

Scenarios for deploying EVPN E-LAN single-homing

These scenarios outline the ideal use cases for deploying EVPN E-LAN single-homing:

- Small branch offices—cost-effective solution for small remote sites or branch offices with minimal traffic or redundancy requirements.
- Initial deployments—suitable for organizations starting their EVPN journey, with the flexibility to scale to multi-homing later.

Benefits of EVPN E-LAN single-homing

The EVPN E-LAN single-homing offers these benefits:

- Simplified design—reduces network complexity by using a single CE-to-PE connection, eliminating the need for redundancy protocols like MC-LAG.
- Cost-effectiveness—minimizes physical links and hardware requirements, making it ideal for small sites or budget-conscious scenarios.
- Efficient resource utilization—consumes less bandwidth and hardware compared to multi-homing, making it suitable for low-traffic or non-critical environments.
- Ease of management—simplifies troubleshooting and operations with single PE-CE connections, removing the need for configuring load balancing or redundancy mechanisms.
- Multipoint connectivity— enables full multipoint-to-multipoint connectivity across all sites, making it perfect for collaborative workspaces or shared data centers.
- Interoperability—ensures compatibility across multi-vendor environments through open standards, providing a flexible foundation for scaling to multi-homing in the future.
- Centralized control—utilizes BGP signaling for efficient MAC address distribution and control, while distributed forwarding ensures effective data handling.
- Scalability—supports large-scale deployments for small or remote sites.

How EVPN E-LAN single-homing transports traffic

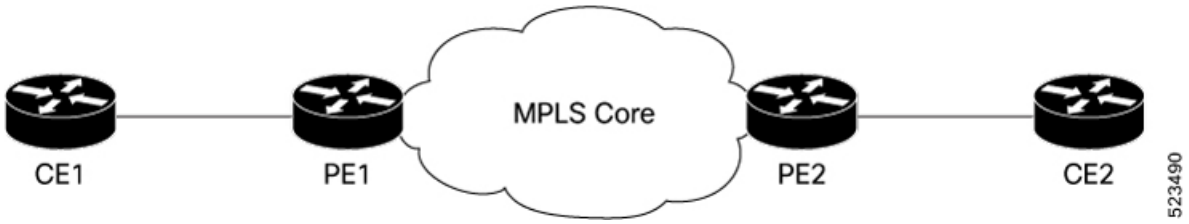
Summary

The EVPN E-LAN single-homing process facilitates the transport of Layer 2 traffic between customer sites by utilizing EVPN control plane protocols. This process ensures efficient traffic forwarding and seamless MAC address distribution across the network.

The key components involved in the process are:

- CE1: Sends Layer 2 traffic to PE1.
- PE1: Receives and processes Layer 2 frames from CE1, determines the destination, and distributes MAC address information.
- PE2: Forwards the Layer 2 frames to the appropriate attachment circuit connected to CE2.
- CE2: Receives the Layer 2 traffic from PE2.

Workflow



The process involves these stages:

1. Traffic transmission from CE1 to PE1
   - CE1 sends Layer 2 traffic to PE1 over a single bundle or physical link.
   - The traffic is encapsulated in Layer 2 frames.
2. Frame processing at PE1
   - PE1 receives the Layer 2 frames on its ingress interface.
   - PE1 checks the destination MAC address of the frame and determines the appropriate attachment circuit for forwarding.
3. MAC address distribution through EVPN control plane
   - PE1 uses EVPN control plane protocols to distribute the MAC address information learned from CE1 to PE2.
4. Traffic forwarding by PE2
   - PE2, having obtained the destination MAC address from the EVPN control plane, forwards the Layer 2 frames to the appropriate attachment circuit connected to CE2.

The process ensures seamless and efficient transport of Layer 2 traffic between customer sites, leveraging EVPN protocols for MAC address distribution and traffic.

Configure EVPN E-LAN single-homing

Enable EVPN E-LAN single-homing on PE1 to advertise MAC addresses and distribute MAC address information from CE1 to PE2.

This task involves configuring Ethernet VPN Identifier (EVI) under the bridge domain and ensuring PE1 advertises MAC addresses. The configuration disables EVPN multi-homing on the bundle interface and sets up BGP for L2VPN and EVPN.

Follow these steps to configure EVPN E-LAN single-homing:

Before you begin

- Ensure PE1 is connected to CE1 and PE2.
- Verify that the required software version supports EVPN E-LAN single-homing.

Procedure

Step 1 Disable EVPN multi-homing.

By default, the bundle interface is in EVPN multi-homing mode. To disable EVPN multi-homing, configure bundle-Ether AC with ESI value (identifier type) set to zero.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.00.00.00.00.00.00.00.00
```

Alternatively, disable EVPN multi-homing globally by:

```
Router(config)# evpn
Router(config-evpn)# ethernet-segment type 1 auto-generation-disable
```

Step 2 Set up BGP for L2VPN and EVPN.

```
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)# neighbor 10.10.10.10
Router#(config-bgp-nbr)# remote-as 200
Router#(config-bgp-nbr)# update-source Loopback 0
Router#(config-bgp-nbr)# address-family l2vpn evpn
```

Step 3 Configure the bridge domain.

```
Router(config)# l2vpn
Router (config-l2vpn)# bridge group BG1
Router (config-l2vpn-bg)# bridge-domain BD1
Router (config-l2vpn-bg-bd)# interface Bundle-Ether1.2001
Router (config-l2vpn-bg-bd-ac)# evi 2001
```

Step 4 Configure MAC advertisement.

```
Router(config)# evpn
Router(config-evpn)# evi 2001
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit
```

Step 5 Running configuration of EVPN E-LAN single-homing mode.

```
router bgp 200
 bgp router-id 10.10.10.1
 address-family l2vpn evpn
 neighbor 10.10.10.10
  remote-as 200 description MPLS-FACING-PEER
  update-source Loopback0
  address-family l2vpn evpn
 !

l2vpn
 bridge group BG1
  bridge-domain BD1
   interface BundleEther1.2001
    evi 2001
   !
```

```
evpn
 interface Bundle-Ether 1
    ethernet-segment
    identifier type 0 00.00.00.00.00.00.00.00.00
   !
evi 2001
 advertise-mac
!
```

Step 6 Use the **show evpn ethernet-segment interface Bundle-Ether 1 detail** command to verify that EVPN E-LAN single-homing mode is configured.

Example:

In this example, the operational mode is SH or single-homing, which indicates that CE1 is connected to PE1 through a single link.

```
Router# show evpn ethernet-segment interface Bundle-Ether 1 detail

Ethernet Segment Id     Interface           Nexthops
----------------------  ----------          ----------
N/A                     Bundle-Ether 1      10.0.0.2
……
Topology :
Operational : SH
```

After the configuration is complete, PE1 operates in single-homing mode (SH) and advertise MAC addresses learned from CE1 to PE2.

EVPN E-Line single-homing

An EVPN E-Line single-homing is a network connectivity mode that

- provides point-to-point Ethernet connectivity between two customer endpoints
- operates over a single physical connection to the PE router, and
- ensures simplicity and cost-effectiveness for scenarios where redundancy is not required.

EVPN E-Line operates over both IP and MPLS cores: the IP core supports BGP functionality, while the MPLS core enables efficient packet switching between endpoints.

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-Line Single-Homing	Release 26.2.1	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
EVPN E-Line Single-Homing	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN E-Line Single-Homing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN E-Line Single-Homing	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The EVPN E-Line single-homing is now extended to the Cisco 8712-MOD-M routers.
EVPN E-Line Single-Homing	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The EVPN E-Line single-homing is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN E-Line Single-Homing	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: Q200, P100]) (select variants only*) *The EVPN E-Line single-homing is now extended to routers with the Q200 and 88-LC1-36EH line cards.
EVPN E-Line Single-Homing	Release 7.8.1	The EVPN E-Line single-homing is a BGP control plane solution for point-to-point services. It implements the signaling and encapsulation techniques for establishing an EVPN instance between a pair of PEs. It provides the service of forwarding L2 Ethernet traffic between network devices without inspecting the MAC header in the Ethernet frame. The use of EVPN for VPWS eliminates the need for signaling single-segment and multi-segment pseudowire (PW) for point-to-point Ethernet services.

EVPN E-Line single-homing for oint-to-point Ethernet connectivity

The EVPN E-Line single-homing is a network service mode that provides point-to-point Ethernet connectivity between two customer endpoints using a single physical connection to the PE router. It is designed for simplicity and cost-effectiveness, making it ideal for small-scale deployments or non-critical applications where

redundancy is not required. This service leverages EVPN technology to deliver reliable Layer 2 VPN services over an IP-MPLS network, adhering to the E-Line service type defined by the Metro Ethernet Forum (MEF).

Benefits EVPN E-Line single-homing

The EVPN E-Line single-homing mode offers these benefits:

- Scalability—achieves scalability without the need for signaling pseudowires, simplifying network operations.
- Ease of provisioning—simplifies the setup process, reducing the time and effort required for deployment.
- Elimination of pseudowires—operates without pseudowires (PWs), streamlining the network architecture.
- Optimal forwarding—leverages BGP's best path selection mechanism to ensure efficient and optimal data forwarding.

Prerequisites for E-Line single-homing

To configure EVPN E-Line, ensure these prerequisites are met:

- BGP configuration: Verify that BGP is configured to support EVPN Subsequent Address Family Identifier (SAFI).
- BGP session: Establish a BGP session between PEs with the `address-family l2vpn evpn` configuration to enable the exchange of EVPN routes.

Restrictions for EVPN E-Line single-homing

These restrictions ensure compatibility, interoperability, and adherence to system requirements and are critical for ensuring proper configuration and avoiding operational issues in EVPN E-Line single-homing deployments.

- VPN ID uniqueness—The VPN ID must be unique per router.
- Route target uniqueness—When specifying a list of route targets, they must be unique per PE (per BGP address-family).
- MTU signaling and enforcement:
  - On versions earlier than IOS XR Release 7.0.x—MTU is not signaled, and MTU mismatches are ignored without interoperability issues.
  - On versions later than IOS XR Release 7.0.x—L3 MTU is advertised by default, and MTU mismatches are enforced by default. However, interoperability issues arise with IOS XR Release 7.3.2 if **transmit-l2-mtu** is configured, as L3 and L2 MTUs do not match. To avoid this, configure the **transmit-mtu-zero** and **ignore-mtu-mismatch** commands.
  - On versions later than IOS XR Release 7.3.2—MTU of 0 is advertised by default, and MTU mismatches are ignored by default. L2 MTU can be advertised using the **transmit-l2-mtu** command, and MTU mismatches can be enforced with the **enforce-mtu-mismatch** command.
- Pseudowire Headend (PWHE) configuration—EVPN E-Line does not support PWHE configuration.
- On versions prior to IOS XR Release 24.1.1:

- EVPN E-line is supported only on single-homing and is not supported on dual homing. This is applicable for both the local and remote sides of the network.
- EVPN validates if the route is for a single home next hop, otherwise it issues an error message. An Ethernet Segment Identifier (ESI) is an attribute that is used to enable EVPN multi-homing. EVPN relies on the ESI value being zero to determine if this is a single home or not. If the AC is a Bundle-Ether interface running LACP, then you need to manually configure the ESI value to zero to overwrite the auto-sense ESI, as EVPN-VPWS multi-homing is not supported.

  To disable EVPN dual homing, configure bundle-Ether AC with ESI value (**identifier type**) set to zero.

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether12
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.00.00.00.00.00.00.00.00
```

  As an alternative, you can disable EVPN dual homing globally.

```
Router(config)# evpn
Router(config-evpn)# ethernet-segment type 1 auto-generation-disable
```

How EVPN-Line single-homing works

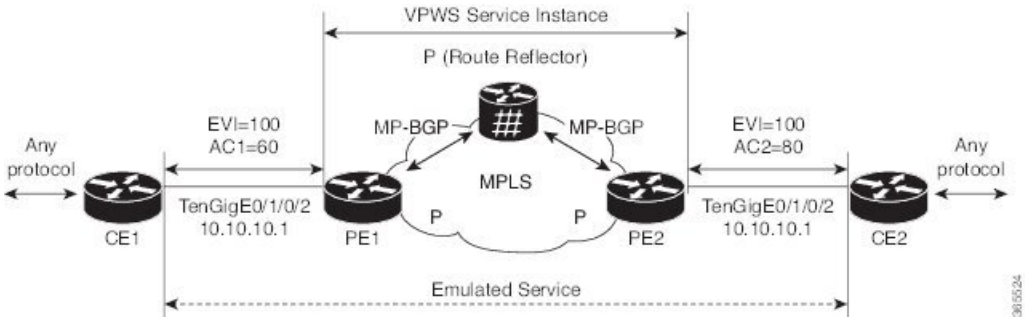
Summary

The EVPN-Line single-homing process enables EVPN services for point-to-point connections using a single PE device. This process ensures efficient communication between local and remote endpoints by leveraging multi-protocol BGP and MPLS labels.

The key components involved in the process are:

- PE devices: PE devices that run multi-protocol BGP to manage EVPN routes and connect to CE routers.
- Ethernet Virtual Instance (EVI): Represents the VPN ID for the service.
- Attachment Circuits (ACs): Local (AC1) and remote (AC2) identifiers for the emulated service endpoints.
- MPLS labels: Allocated per local AC for reachability.
- CE devices: Customer Edge routers (CE1 and CE2) that connect to the service provider network.
- Route Reflector (P): Facilitates MP-BGP signaling within the MPLS network.
- MPLS Network: Provides the transport for the emulated service.

Workflow



The EVPN E-line single-homing process involves these stages:

1. Configuration of PE1 and PE2.
    - PE1:
        - Specify the VPN ID (EVI).
        - Define local and remote AC identifiers (AC1 and AC2).
        - Allocate an MPLS label for the local AC.
    - PE2:
        - Mirror the configuration of PE1 to ensure symmetry.
        - Allocate an MPLS label for the local AC.
2. Route advertisement.
    - PE1 advertises a single EVPN per EVI Ethernet Auto Discovery (AD) route for each local endpoint (AC) to remote PEs, including the associated MPLS label.
    - PE2 performs the same task.
3. Route reception and database update
    - Upon receiving the EVPN per EVI AD route from PE2, PE1 updates its local EVPN database with the path list to reach AC2 (for example., the next hop is PE2's IP address and MPLS label for AC2).
    - PE2 performs the same task for routes received from PE1.
4. Traffic encapsulation and forwarding
    - CE1 and CE2 communicate using any protocol to send traffic to PE1 and PE2, respectively.
    - PE1 and PE2 encapsulate the traffic and forward it across the MPLS network.
    - MP-BGP is used between PE1, PE2, and the RR (P) to exchange VPWS service instance information.
    - The RR (P) reflects the MP-BGP updates between PE1 and PE2.
    - PE1 and PE2 decapsulate the traffic and forward it to CE1 and CE2, respectively.

The EVPN E-line single-homing process enables efficient and reliable communication between customer edge devices over an MPLS network, ensuring seamless data transfer and network stability.

Configure EVPN E-Line single-homing

Configure EVPN E-line single-homing to establish connectivity between PE devices.

Before you begin

- Ensure that the PE devices are connected and reachable.
- Verify that the required interfaces are operational.
- Confirm that the BGP configuration is enabled on both PE devices.

Procedure

Step 1 Configure PE1.

Example:

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

Step 2 Configure PE2.

Example:

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

```
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# commit
```

If the source and target AC IDs are the same, use the **neighbor evpn evi 100 service 10** command to configure the neighbor EVPN.

Step 3 Running configuration of EVPN E-Line single-homing mode.

Example:

```
/* On PE1 */
configure
router bgp 100
 address-family l2vpn evpn
 neighbor 10.10.10.1
  address-family l2vpn evpn
 !

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


/* On PE2 */
configure
router bgp 100
 address-family l2vpn evpn
 neighbor 10.10.10.1
  address-family l2vpn evpn
 !

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

Supported EVPN E-Line single-homing features

EVPN E-Line single-homing supports these features:

- Flow label support for EVPN E-Line
- Private line emulation over EVPN E-Line Single-Homing

Flow label support for EVPN E-Line

A flow label support for EVPN E-Line is a feature that

- enables provider (P) routers to use flow-based load balancing to forward traffic between PE devices

- utilizes Flow-Aware Transport (FAT) of pseudowires (PW) over an MPLS packet-switched network for load balancing traffic across BGP-based signaled pseudowires for EVPN E-Line, and
- improves traffic distribution across equal cost multipaths (ECMP) or link-bundled paths in the core.

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
Flow label support for EVPN E-Line	Release 26.2.1	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.
Flow label support for EVPN E-Line	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Flow label support for EVPN E-Line	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
Flow label support for EVPN E-Line	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 Flow Label Support for EVPN functionality is now extended to: • 8212-48FH-M • 8712-MOD-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E • 88-LC1-36EH
Flow label support for EVPN E-Line	Release 7.3.2	The Flow Label support improves traffic distribution by enabling flow-based load balancing between provider edge devices. It uses Flow-Aware Transport (FAT) pseudowires over an MPLS network to efficiently manage traffic across BGP-signaled pseudowires in an Ethernet VPN. Flow labels, based on packet flows, help optimize traffic across equal cost multipaths or link-bundled paths.

Optimize traffic distribution using flow label

A service provider deploying EVPN E-Line can use this feature to optimize traffic distribution across multiple paths in their MPLS core network. A network with high traffic volumes can benefit from reduced congestion and improved performance by leveraging flow-based load balancing.

Flow-based load balancing with FAT pseudowires

FAT pseudowires (PWs) in EVPN E-Line provides the capability to identify individual flows within a PW, allowing routers to load-balance traffic effectively. When ECMP is used, FAT PWs ensure better traffic distribution by creating a flow label based on indivisible packet flows entering an imposition PE. This flow label is inserted as the lowermost label in the packet. P routers then use the flow label for load balancing.

A flow is identified by either:

- The source and destination IP address of the traffic, or
- The source and destination MAC address of the traffic.

How FAT PWs distribute flows over ECMPs and bundle links

FAT PWs are designed to improve traffic distribution across ECMPs and link bundles by introducing greater entropy through flow labels. This ensures efficient utilization of network resources and enhances performance.

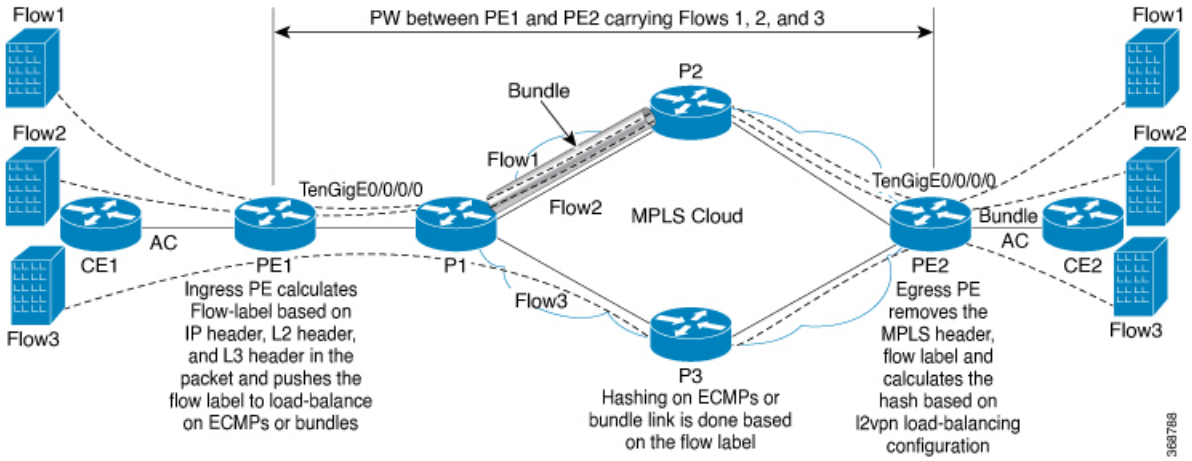
Summary

The process of distributing flows over Equal-Cost Multi-Paths (ECMPs) and bundle links in a FAT Pseudowire (PW) involves the use of flow labels to enhance traffic load balancing.

The key components involved in the process are:

- Flow label: A unique identifier derived from source and destination MAC and IP addresses, inserted into the label stack.
- Ingress PE (PE1):Adds the flow label to the traffic.
- Egress PE (PE2): Discards the flow label and uses a single EVPN label for unicast traffic.
- Core routers (P routers): Use the flow label for load balancing.

Workflow



These stages describe how flow-based load balancing operates to optimize traffic distribution across PE device:

1. Flow label generation:
   - The ingress PE (PE1) generates a flow label for each unique incoming flow based on source and destination MAC and IP addresses.
   - The flow label is inserted after the VC label and before the control word (if any).
2. Traffic forwarding:
   - PE1 forwards the traffic with the flow label to the core routers.
   - Core routers use the flow label, along with other information like MAC and IP addresses, to perform load balancing across ECMPs and bundle links.
3. Traffic handling at egress PE:
   - The egress PE (PE2) receives the traffic and discards the flow label.
   - PE2 uses a single EVPN label for all unicast traffic, ensuring compatibility with mixed traffic types (with or without flow labels).

The process enables efficient load balancing of traffic across ECMPs and bundle links, improving network performance and resource.

Restrictions for configuring flow label for EVPN E-Line

To ensure proper configuration and avoid unsupported scenarios, follow these restrictions:

- Unsupported services
  - This feature is not supported for EVPN Point-to-Multipoint (P2MP) of VPLS and Ethernet LAN (E-LAN) services.
  - This feature is not supported for EVPN flexible cross-connect service.
  - This feature is not supported for EVPN E-Line multi-homing.

- Supported configuration
  - This feature is supported only for EVPN E-Line single homing.
  - Ensure that AC bundle interfaces are configured with ESI-0 only

Failure to comply with these restrictions may result in configuration errors or unsupported network behavior.

Configure flow label for EVPN E-Line

Configure a flow label for EVPN E-Line on both PE1 and PE2 to enable load balancing and efficient traffic management.

EVPN E-Line is a point-to-point Ethernet VPN service that uses MPLS encapsulation. Configuring a flow label ensures proper load balancing across the network.

Before you begin

- Ensure that PE1 and PE2 are running compatible software versions.
- Verify that the interfaces and EVPN instances are pre-configured.

Procedure

Step 1 Configure L2VPN xconnect group.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/0
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 2 source 1
Router(config-l2vpn-xc-p2p)# commit
```

Step 2 Configure the EVPN instance.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-instance)# load-balancing
Router(config-evpn-instance-lb)# flow-label static
Router(config-evpn-instance-lb)# commit
```

Step 3 Flow label for EVPN E-Line running configuration.

Example:

```
l2vpn
 xconnect group evpn-vpws
  p2p evpn1
   interface TenGigE0/0/0/0
```

```
   neighbor evpn evi 1 target 2 source 1
   !
  !
evpn
 evi 1
   load-balancing
   flow-label static
  !
!
```

Step 4 Use the **show l2vpn xconnect detail** command to verify that EVPN E-Line flow label is configured.

Example:

```
Router# show l2vpn xconnect detail
Group evpn-vpws, XC evpn1, state is up; Interworking none
  AC: TenGigE0/0/0/0, state is up
    Type Ethernet
    MTU 1500; XC ID 0x1; interworking none
    Statistics:
      packets: received 21757444, sent 0
      bytes: received 18226521128, sent 0
  EVPN: neighbor 100.100.100.2, PW ID: evi 1, ac-id 2, state is up ( established )
    XC ID 0xc0000001
    Encapsulation MPLS
   Encap type Ethernet, control word disabled
    Sequencing not set
    LSP : Up
    Flow Label flags configured (Tx=1,Rx=1) statically

      EVPN         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        64002                          64002
      MTU          1500                           1500
      Control word disabled                       disabled
      AC ID        1                              2
      EVPN type    Ethernet                       Ethernet

      ------------ ------------------------------ ------------------------------
    Create time: 30/10/2018 03:04:16 (00:00:40 ago)
    Last time status changed: 30/10/2018 03:04:16 (00:00:40 ago)
    Statistics:
      packets: received 0, sent 21757444
      bytes: received 0, sent 18226521128
```

Private line emulation over EVPN E-Line single-homing

A Packet Label Edge (PLE) service is a mechanism that

- enables transparent packet transfer across different port modes over MPLS networks
- supports EVPN E-Line single-homing service for PLE client traffic, and
- facilitates pseudowire channel setup between endpoints during L2VPN cross-connect establishment.

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
Private Line Emulation over EVPN-VPWS Single Homed	Release 26.2.1	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.
Private Line Emulation over EVPN-VPWS Single Homed	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Private Line Emulation over EVPN-VPWS Single Homed	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
Private Line Emulation over EVPN-VPWS Single Homed	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The Private Line Emulation over EVPN-VPWS Single Homed functionality is now extended to the Cisco 8712-MOD-M routers.
Private Line Emulation over EVPN-VPWS Single Homed	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The Private Line Emulation over EVPN-VPWS Single Homed 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
Private Line Emulation over EVPN-VPWS Single Homed	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100])(select variants only*) *The Private Line Emulation over EVPN-VPWS Single Homed functionality is now extended to routers with the 88-LC1-36EH line cards.
Private Line Emulation over EVPN-VPWS Single Homed	Release 7.11.1	Introduced in this release on: Cisco 8011-2X2XP4L PLE Service Endpoint Router. You can now configure EVPN VPWS to carry the client traffic from ports like FC, OTN, SDH, SONET, or Ethernet and forward the traffic to the core network by using Private Line Emulation (PLE). PLE emulates the switching capabilities of FC, OTN, SDH, SONET, or Ethernet ports without needing a dedicated equipment and allows interconnecting optical networks with Ethernet networks. This feature introduces the **port-mode** command. This release introduces new and modified YANG data models for PLE. For the list of supported data models, see *Supported Yang Data Models for PLE*. You can access these data models from the Github repository.

Key components and processes of PLE

Packet Line Emulation (PLE) service is a mechanism designed to enable the seamless transfer of packets from various port modes over MPLS networks. PLE integrates the use of EVPN-VPWS, BGP, pseudowires, and CEM to provide a robust mechanism for transparent packet transfer across MPLS networks, while maintaining compatibility with native client interfaces. This is achieved through the following key components and processes:

- Traffic handling: PLE client traffic is carried using an EVPN-VPWS single-homed setup. The traffic flow begins at the PLE initiator and terminates at the PLE endpoint.

- BGP session for route exchange: PLE endpoints establish a BGP session to exchange EVPN routing information, ensuring proper connectivity and signaling between endpoints.
- Pseudowire setup: A pseudowire channel is established between endpoints when a Layer 2 VPN cross-connect is configured. This cross-connect links the PLE client, represented as a Circuit Emulation (CEM) interface, to the remote node.
- Circuit Emulation (CEM): CEM provides native client interfaces and allows data to be transmitted over Ethernet or MPLS networks. It encapsulates client bitstreams into packet payloads with pseudowire emulation headers, enabling the transport of circuits over a Packet Switched Network (PSN).
- Encapsulation and transport: PLE client traffic is encapsulated by the PLE initiator and carried over EVPN-VPWS Layer 2 services running on Segment Routing or MPLS tunnels. Label imposition and disposition facilitate traffic flow between the client and core networks.
- Traffic extraction: At the PLE terminator node, the encapsulated bitstreams are extracted from EVPN packets and delivered to the PLE client interface. This process is governed by the client attribute and CEM profile, ensuring accurate delivery.

These stages are crucial for understanding traffic flow and the mechanisms of encapsulation and decapsulation, which are employed to emulate traditional private line services within an EVPN:

Restrictions for PLE over EVPN E-Line single-homing

These restrictions apply exclusively to the Cisco 8011-2X2XP4L PLE Service Endpoint Router running IOS XR Release 7.11.1:

- Load balancing: PLE traffic in the core does not support load balancing. This limitation arises because PLE is incompatible with ECMP or core bundles containing more than one member link.
- Software offloading: Supported only for SR-TE performance monitoring. Consequently, Fast Reroute (FRR) convergence is not achievable.

For more information on label stacking, see *MPLS Configuration Guide for Cisco 8000 Series Routers*.

Supported hardware for PLE

PLE is supported on Cisco 8011-2X2XP4L PLE Service Endpoint Router with SFP+ optical transceivers and supports these port mode options:

- Ethernet – 10GE
- Fiber channel (FC) – 1G, 2G, 4G, 8G, 16G, and 32G
- Optical Transport Network (OTN) – OTU2 and OTU2e
- Synchronous Digital Hierarchy (SDH) – STM16 and STM64
- Synchronous Optical Networking (SONET) – OC48 and OC192

PLE forwarding flow - how MPLS label imposition works

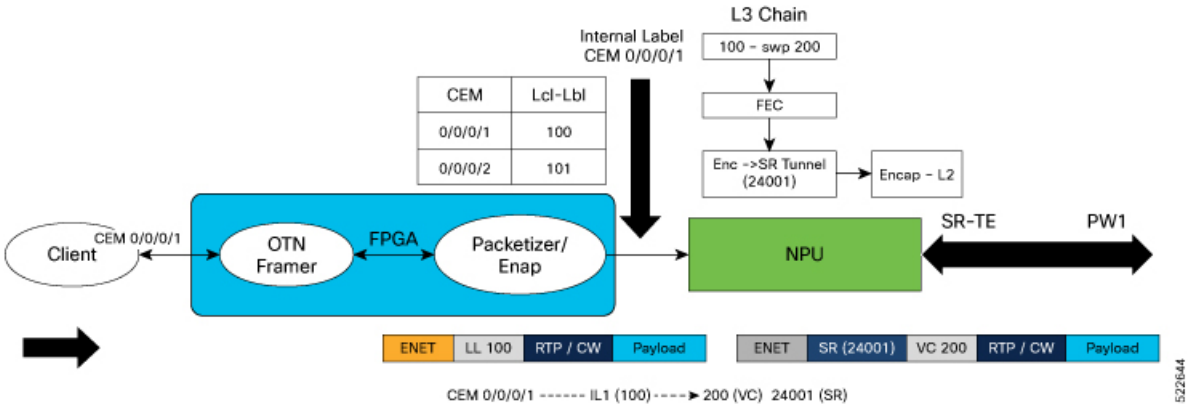
Summary

The MPLS label imposition process involves adding an MPLS label to a data packet to enable its forwarding within an MPLS network. The key components involved in this process are:

- PE router: Adds an MPLS label to packets entering the MPLS network.
- Field Programmable Gate Array (FPGA): Acts as a forwarding block, assigning an internal local label to traffic from the client.
- Network Processing Unit (NPU): Replaces the internal local label with a Virtual Circuit (VC) label and forwards the traffic using a transport label.

Workflow

Figure 2: PLE forwarding flow – imposition



The MPLS label imposition process involves these stages:

1. Traffic reception
   - Traffic from the client, which can be in various port modes, such as, FC, OTN, SDH, SONET, Ethernet, is received by the FPGA.
   - The FPGA assigns an internal local label, for example, 100 to the traffic and forwards it to the NPU.
2. Label replacement
   - The NPU receives the traffic with the internal local label from the FPGA.
   - In the forwarding L3 chain, the NPU replaces the internal local label, for example, 100, with a Virtual Circuit (VC) label, for example 200, also known as the PW label.
3. Traffic forwarding
   - The NPU adds a transport label, for example, 24001, to the packet.
   - The traffic is forwarded towards the core network using the transport label.

   The MPLS label imposition process ensures that packets are correctly labeled and forwarded within the MPLS network, enabling efficient and seamless data transmission.

PLE forwarding flow - how MPLS label disposition works

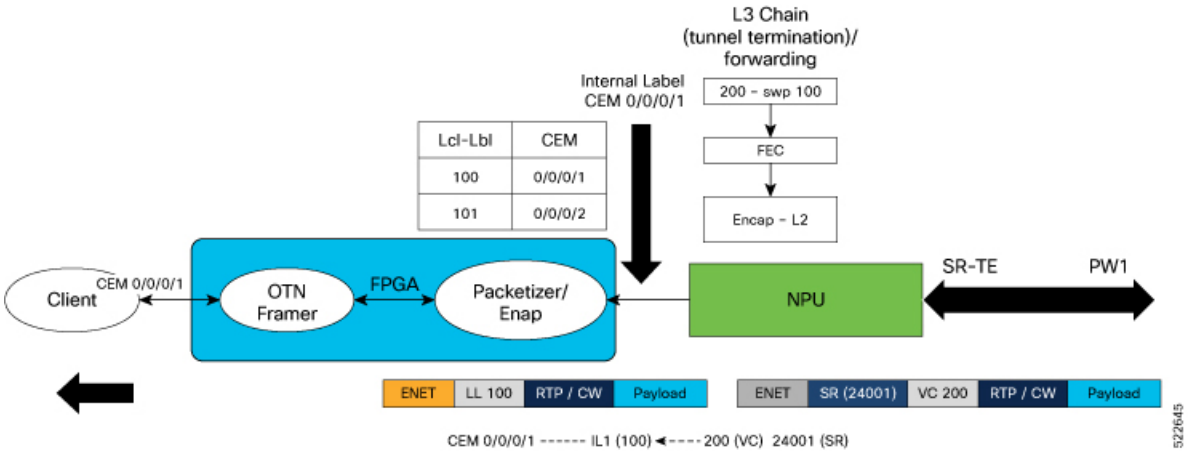
Summary

The PLE forwarding flow involves the disposition of MPLS labels to forward traffic from the core network to the client network. This process ensures efficient traffic delivery by removing MPLS labels and forwarding packets to the client. The key components involved in the process are:

- PE router: Receives MPLS packets, makes forwarding decisions, and removes MPLS labels.
- NPU: Determines the outgoing interface based on VC label allocation.
- FPGA: Maps internal local labels to the appropriate CEM interface for client delivery.

Workflow

Figure 3: PLE forwarding flow – disposition



The process involves these stages:

1. Traffic reception: The NPU receives traffic with a VC label.
2. Outgoing interface determination: The NPU determines the outgoing interface based on the VC label allocation.
3. Label replacement: The VC label, for example, 200, is replaced with an internal local label, for example, 100, and sent to the FPGA.
4. Label mapping: The FPGA maps the internal local label to the CEM interface, example 0/0/0/1.
5. Traffic forwarding: The traffic is forwarded to the client through the mapped CEM interface.

   The PLE forwarding flow ensures that traffic is efficiently delivered from the core network to the client network by removing MPLS labels and forwarding packets to the appropriate client interface.

PLE transport mechanism

Summary

The PLE forwarding flow involves the disposition of MPLS labels to forward traffic from the core network to the client network. This process ensures efficient traffic delivery by removing MPLS labels and forwarding packets to the client. The key components involved in the process are:

- Circuit-style SR-TE policies: Configured statically as preferred paths within pseudowire classes.
- Pseudowire class: Associates SR-TE policies with working or protected pseudowires.
- Co-router bidirectional paths: Ensures traffic flows in both directions with guaranteed latency.

Workflow

The process involves these stages:

1. Policy configuration: Circuit-style SR-TE policies are configured statically as preferred paths within pseudowire classes.
2. Pseudowire association: An SR-TE policy is associated per pseudowire by assigning the corresponding pseudowire class.
3. Traffic transport: The configured SR-TE policies ensure guaranteed latency, bandwidth, and end-to-end path protection for client traffic.

   The PLE transport mechanism provides a reliable and efficient method for transporting client traffic over networks, ensuring guaranteed performance and protection.

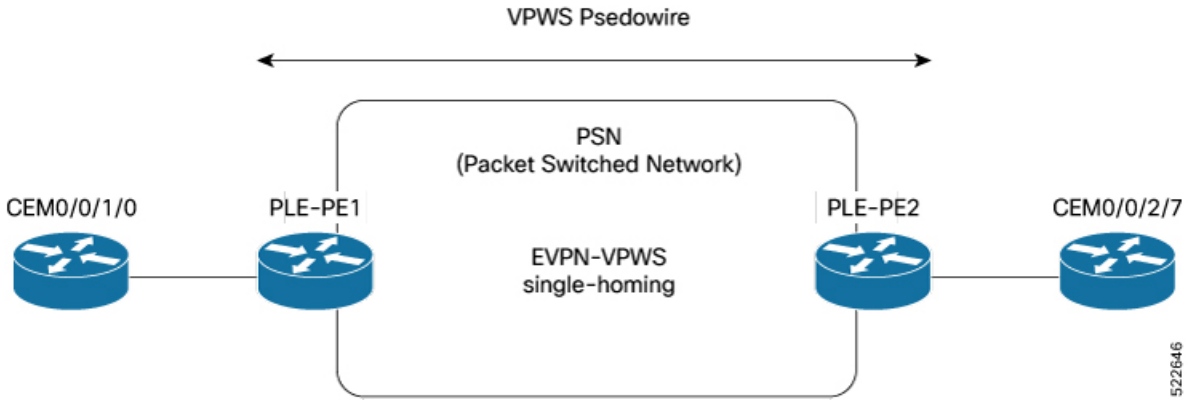
For more information on SR-TE policies, see the *Configure SR-TE Policies* section in the *Segment Routing Configuration Guide for Cisco 8000 Series Routers*.

Configure PLE over EVPN E-Line

Configure PLE over EVPN E-Line to enable efficient traffic management and synchronization

This task involves setting up PLE over EVPN E-Line with prerequisites, topology setup, and configuration steps.

Perform these tasks to configure EVPN-VPWS over SR-TE policy with explicit path. For more information on SR-TE policies, see the *Configure SR-TE Policies* section in the *Segment Routing Configuration Guide for Cisco 8000 Series Routers, IOS XR*.



In this topology, CEM interfaces are connected to PLE interfaces. The PLE interfaces, PE1 and PE2, are connected through EVPN-VPWS single homing. The PLE interface can be: Ethernet, OTN, FC, or SONET/SDH.

Before you begin

- Install all mandatory Cisco RPMs, such as RSVP for MPLS-TE.

  For more information, see the *Implementing RSVP for MPLS-TE* section in the *MPLS Configuration Guide for Cisco 8000 Series Routers*.
- Ensure clock synchronization between routers using SyncE or PTP to prevent data traffic drops.
- Verify that the core interface bandwidth exceeds the access interface bandwidth. For example, if CE traffic is 10G, the core interface must support at least 25G.

Procedure

Step 1 Enable frequency synchronization.

Example:

```
/* Enable frequency synchronization on PLE-PE1 */
Router(config)# frequency synchronization
Router(config-freqsync)# quality itu-t option 1
Router(config-freqsync)# exit
Router(config)# interface TwentyFiveGigE0/0/0/24
Router(config-if)# frequency synchronization
Router(config-if-freqsync)# quality transmit exact itu-t option 1 PRC
```

SyncE or PTP must be UP. Use the **show frequency synchronization interfaces** command to verify that the clock is transmitted.

Example:

```
/* Enable frequency synchronization on PLE-PE2 */
Router(config)# frequency synchronization
Router(config-freqsync)# quality itu-t option 1
Router(config-freqsync)# exit
Router(config)# interface TwentyFiveGigE0/0/0/32
Router(config-if)# frequency synchronization
```

```
Router(config-if-freqsync)# selection input
Router(config-if-freqsync)# priority 1
Router(config-if-freqsync)# wait-to-restore 0
```

Use the **show frequency synchronization selection** command to verify if PLE-PE2 is LOCKED to PLE-PE1's clock.

Step 2 Bring up the optics controller in CEM packet mode.

Configure the optics controller based on the port mode type (Ethernet, OTN, FC, or SONET/SDH). The examples show port mode configuration for all the types of port modes. Use the relevant command according to the port mode type of the PLE interface.

Example:

Bring up the optics controller in CEM packet mode with appropriate speed on PLE-PE1.

Ethernet:

```
Router(config)# controller Optics0/0/1/0
Router(config-Optics)# port-mode Ethernet framing cem-packetize rate 10GE
Router(config-Optics)# exit
Router(config)# controller Optics0/0/1/5
Router(config-Optics)# port-mode Ethernet framing cem-packetize rate 1GE
```

OTN:

```
Router(config)# controller Optics0/0/2/0
Router(config-Optics)# port-mode otn framing cem-packetize rate otu2
Router(config-Optics)# exit
Router(config)# controller Optics0/0/2/0
Router(config-Optics)# port-mode otn framing cem-packetize rate otu2e
```

Fiber Channel:

```
Router(config)# controller Optics0/0/1/6
Router(config-Optics)# port-mode FC framing cem-packetize rate FC1
```

Note
Port mode FC32 is supported only on the even ports (Port 0, 2, 4, and 6) of the MPA.

SONET/SDH:

```
Router(config)# controller optics 0/0/2/4
Router(config-Optics)# port-mode sonet framing cem-packetize rate OC48
Router(config-Optics)# exit
Router(config)# controller optics 0/0/2/5
Router(config-Optics)# port-mode sdh framing cem-packetize rate STM16
```

Bring up the optics controller in CEM packet mode with appropriate speed on PLE-PE2.

Ethernet:

```
Router(config)# controller Optics0/0/2/7
Router(config-Optics)# port-mode Ethernet framing cem-packetize rate 10GE
Router(config-Optics)# exit
Router(config)# controller Optics0/0/1/5
Router(config-Optics)# port-mode Ethernet framing cem-packetize rate 1GE
```

OTN:

```
Router(config)# controller Optics0/0/2/0
Router(config-Optics)# port-mode otn framing cem-packetize rate otu2
Router(config-Optics)# exit
Router(config)# controller Optics0/0/2/0
Router(config-Optics)# port-mode otn framing cem-packetize rate otu2e
!
```

Fiber Channel:

```
Router(config)# controller Optics0/0/1/6
Router(config-Optics)# port-mode FC framing cem-packetize rate FC1
```

Note
Port mode FC32 is supported only on the even ports (Port 0, 2, 4, and 6) of the MPA.

SONET/SDH:

```
Router(config)# controller optics 0/0/2/4
Router(config-Optics)# port-mode sonet framing cem-packetize rate OC48
Router(config-Optics)# exit
Router(config)# controller optics 0/0/2/5
Router(config-Optics)# port-mode sdh framing cem-packetize rate STM16
```

Step 3 Configure access and core interfaces.

Example:

Configure the access interface for the client and then the core interface.

Configure the access and core interfaces on PLE-PE1.

Access interface: Repeat this for each port mode configuration.

```
Router(config)# interface CEM0/0/1/0
Router(config-if)# l2transport
```

Core interface:

```
Router(config)# interface TwentyFiveGigE0/0/0/24
Router(config-if)# ipv4 address 14.1.0.1 255.255.255.252
```

Example:

Configure the access and core interfaces on PLE-PE2.

Access interface: Repeat this for each port mode configuration.

```
Router(config)# interface CEM0/0/2/7
Router(config-if)# l2transport
```

Core interface:

```
Router(config)# interface TwentyFiveGigE0/0/0/32
Router(config-if)# ipv4 address 14.1.0.2 255.255.255.252
```

Step 4 Configure loopback interface to establish BGP-EVPN neighborship.

Example:

```
/* Configure loopback interface on PLE-PE1 */
Router(config)# interface Loopback0
Router(config-if)# ipv4 address 1.1.1.1 255.255.255.255

/* Configure loopback interface on PLE-PE2 */
Router(config)# interface Loopback0
Router(config-if)# ipv4 address 1.1.1.4 255.255.255.255
!
```

Step 5 Configure IS-IS IGP.

Example:

Configure IS-IS IGP to advertise the configured loopback and core interfaces.

Note
You cannot configure Topology-Independent Loop-Free Alternate (TI-LFA) on the links used by circuit-styled SR-TE tunnel. The adjacency SID label is unprotected for circuit-styled SR-TE, which does not support TI-LFA.

Configure IS-IS IGP on PLE-PE1.

```
Router(config)# router isis core
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 49.0000.0000.0000.0001.00
Router(config-isis)# nsr
Router(config-isis)# nsf cisco
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# segment-routing bundle-member-adj-sid
Router(config-isis-af)# commit
Router(config-isis-af)# exit
Router(config-isis)# interface Loopback0
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# prefix-sid index 1
Router(config-isis-if-af)# exit
Router(config-isis)# interface TwentyFiveGigE0/0/0/24
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# adjacency-sid absolute 28121 >>>>> Adjacency-SID must be unprotected for
 circuit-styled SR-TE
Router(config-isis-if-af)# commit
Router(config-isis-if-af)# exit
```

Configure IS-IS IGP on PLE-PE2.

```
Router(config)# router isis core
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 49.0000.0000.0000.0004.00
Router(config-isis)# nsr
Router(config-isis)# nsf cisco
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# segment-routing bundle-member-adj-sid
Router(config-isis-af)# commit
Router(config-isis-af)# exit
Router(config-isis)# interface Loopback0
```

```
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# prefix-sid index 4
Router(config-isis-if-af)# exit
Router(config-isis)# interface TwentyFiveGigE0/0/0/32
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# adjacency-sid absolute 28211 >>>>> Adjacency-SID must be unprotected for
 circuit-styled SR-TE
Router(config-isis-if-af)# commit
Router(config-isis-if-af)# exit
```

Step 6 Configure performance measurement to enable the liveness monitoring of the SR policy.

Example:

Configure performance measurement on PLE-PE1.

```
Router(config)# performance-measurement
Router(config-perf-meas)# liveness-profile sr-policy name RED
Router(config-pm-ld-srpolicy)# probe
Router(config-pm-ld-srpolicy-probe)# measurement-mode loopback
Router(config-pm-ld-srpolicy-probe)# burst-interval 3000
Router(config-pm-ld-srpolicy-probe)# exit
Router(config-pm-ld-srpolicy)# exit

Router(config-perf-meas)# liveness-profile sr-policy name BLUE
Router(config-pm-ld-srpolicy)# probe
Router(config-pm-ld-srpolicy-probe)# measurement-mode loopback
Router(config-pm-ld-srpolicy-probe)# burst-interval 30
```

Configure performance measurement on PLE-PE2.

```
Router(config)# performance-measurement
Router(config-perf-meas)# liveness-profile sr-policy name RED
Router(config-pm-ld-srpolicy)# probe
Router(config-pm-ld-srpolicy-probe)# measurement-mode loopback
Router(config-pm-ld-srpolicy-probe)# burst-interval 3000
Router(config-pm-ld-srpolicy-probe)# exit
Router(config-pm-ld-srpolicy)# exit

Router(config-perf-meas)# liveness-profile sr-policy name BLUE
Router(config-pm-ld-srpolicy)# probe
Router(config-pm-ld-srpolicy-probe)# measurement-mode loopback
Router(config-pm-ld-srpolicy-probe)# burst-interval 30
```

Step 7 Configure segment routing traffic engineering tunnels.

Example:

Configure circuit-styled SR-TE tunnels. SR-TE is supported only with explicit path specified by adjacency SID labels. The adjacency SID labels must be unprotected for circuit-styled SR-TE. This example shows configuration of explicit path between PE1 and PE2.

Configure segment routing traffic engineering tunnels on PLE-PE1.

```
Router(config)# segment-routing
Router(config-sr)# global-block 80000 111999
Router(config-sr)# local-block 25000 28999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list pe1-pe2-forward-path
```

```
Router(config-sr-te-sl)# index 1 mpls label 28121
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list pe1-pe2-reverse-path
Router(config-sr-te-sl)# index 1 mpls label 28211
Router(config-sr-te-sl)# exit
Router(config-sr-te)# policy pe1-pe2-circuit-styled-srte
Router(config-sr-te-policy)# color 10 end-point ipv4 1.1.1.4
Router(config-sr-te-policy)# path-protection
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 10
Router(config-sr-te-policy-path-pref)# explicit segment-list pe1-pe2-forward-path >>>>> Explicit
path
Router(config-sr-te-policy-path-pref)# reverse-path segment-list pe1-pe2-reverse-path
Router(config)# performance-measurement
Router(config-perf-meas)# liveness-detection
Router(config-perf-meas)# liveness-profile backup name RED
Router(config-perf-meas)# liveness-profile name BLUE
```

Configure segment routing traffic engineering tunnels on PLE-PE2.

```
Router(config)# segment-routing
Router(config-sr)# global-block 80000 111999
Router(config-sr)# local-block 25000 28999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list pe1-pe2-forward-path
Router(config-sr-te-sl)# index 1 mpls label 28211
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list pe1-pe2-reverse-path
Router(config-sr-te-sl)# index 1 mpls label 28121
Router(config-sr-te-sl)# exit
Router(config-sr-te)# policy pe1-pe2-circuit-styled-srte
Router(config-sr-te-policy)# color 10 end-point ipv4 1.1.1.1
Router(config-sr-te-policy)# path-protection
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 10
Router(config-sr-te-policy-path-pref)# explicit segment-list pe1-pe2-forward-path >>>>> Explicit
path
Router(config-sr-te-policy-path-pref)# reverse-path segment-list pe1-pe2-reverse-path
Router(config)# performance-measurement
Router(config-perf-meas)# liveness-detection
Router(config-perf-meas)# liveness-profile backup name RED
Router(config-perf-meas)# liveness-profile name BLUE
```

Step 8 Configure BGP EVPN neighbor session.

Example:

Configure L2VPN EVPN address family under BGP to establish a BGP-EVPN neighbor session.

Configure BGP EVPN neighbor session on PLE-PE1.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 1.1.1.1
Router(config-bgp)# bgp graceful-restart
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 1.1.1.4
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# exit
```

```
Router(config-bgp)# graceful-restart
Router(config-bgp)# address-family l2vpn evpn
```

Configure BGP EVPN neighbor session on PLE-PE2.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 1.1.1.4
Router(config-bgp)# bgp graceful-restart
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 1.1.1.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# exit
Router(config-bgp)# graceful-restart
Router(config-bgp)# address-family l2vpn evpn
```

Step 9 Configure EVPN VPWS.

Example:

Configure EVPN VPWS with PW class and xconnect service to carry the PLE client traffic.

Configure EVPN VPWS on PLE-PE1.

```
Router(config)# l2vpn
Router((config-l2vpn)# pw-class pw-cs-srte
Router((config-l2vpn-pwc)# encapsulation mpls
Router((config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_10_ep_1.1.1.6
Router(config)# xconnect group evpn_vpws
Router(config)# p2p p1
Router(config)# interface CEM0/0/1/0
Router(config)# neighbor evpn evi 10 target 1 source 2
Router(config)# pw-class pw-cs-srte
```

Configure EVPN VPWS on PLE-PE2.

```
Router(config)# l2vpn
Router((config-l2vpn)# pw-class pw-cs-srte
Router((config-l2vpn-pwc)# encapsulation mpls
Router((config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_10_ep_1.1.1.1
Router(config)# xconnect group evpn_vpws
Router(config)# p2p p1
Router(config)# interface CEM0/0/2/7
Router(config)# neighbor evpn evi 10 target 1 source 2
Router(config)# pw-class pw-cs-srte
```

Step 10 Configure QoS policy on CEM interface.

Example:

Configure QoS policy to manage congestion on PLE client traffic. In QoS for PLE, you can mark the MPLS experimental with only the topmost label and set the traffic class with only the default class.

Configure QoS policy on PLE-PE1.

Access interface configuration

```
Router(config)# policy-map ple-policy
```

```
Router(config-pmap)# class class-default
Router(config-pmap-c)# set mpls experimental topmost 7
Router(config-pmap-c)# set traffic-class 2
Router(config-pmap-c)# end-policy-map
!
Router(config)# interface CEM0/0/1/0
Router(config-if)# l2transport
Router(config-if)# service-policy input ple-policy
!
```

Core interface configuration

```
Router(config)# class-map match-any tc2
Router(config-cmap)# match traffic-class 2
Router(config-cmap)# end-class-map
!
Router(config)# policy-map core
Router(config-pmap)# class tc2
Router(config-pmap-c)# priority level 1
Router(config-pmap-c)# shape average percent 100
Router(config-pmap-c)# end-policy-map
!
Router(config)# interface TwentyFiveGigE0/0/0/24
Router(config-if)# mtu 9200
Router(config-if)# service-policy output core
Router(config-if)# ipv4 address 13.30.1.1 255.255.255.252
```

Configure QoS policy on PLE-PE2.

Access interface configuration

```
Router(config)# policy-map ple-policy
Router(config-pmap)# class class-default
Router(config-pmap-c)# set mpls experimental topmost 7
Router(config-pmap-c)# set traffic-class 2
Router(config-pmap-c)# end-policy-map
!
Router(config)# interface CEM0/0/2/7
Router(config-if)# l2transport
Router(config-if)# service-policy input ple-policy
```

Core interface configuration

```
Router(config)# class-map match-any tc2
Router(config-cmap)# match traffic-class 2
Router(config-cmap)# end-class-map
!
Router(config)# policy-map core
Router(config-pmap)# class tc2
Router(config-pmap-c)# priority level 1
Router(config-pmap-c)# shape average percent 100
Router(config-pmap-c)# end-policy-map
!
Router(config)# interface TwentyFiveGigE0/0/0/32
Router(config-if)# mtu 9200
Router(config-if)# service-policy output core
Router(config-if)# ipv4 address 46.10.1.2 255.255.255.252
```

Step 11 Use these show commands to view the configuration.

Example:

Use the **show isis neighbors** command to verify the IS-IS configuration.

```
Router# show isis neighbors
Fri Nov 12 09:04:13.638 UTC

IS-IS core neighbors:
System Id      Interface        SNPA           State Holdtime Type IETF-NSF
PLE-Core-PE2   TF0/0/0/24       *PtoP*         Up    28       L2   Capable

Total neighbor count: 1
```

```
Router# show isis segment-routing label table

Fri Nov 12 09:25:18.488 UTC

IS-IS core IS Label Table
Label         Prefix                 Interface
----------    ----------------       ---------
16001         1.1.1.1/32             Loopback0
16004         1.1.1.4/32
```

```
Router# show mpls forwarding prefix 1.1.1.4/32
Fri Nov 12 09:25:54.898 UTC
Local  Outgoing    Prefix            Outgoing     Next Hop        Bytes
Label  Label       or ID             Interface                    Switched
------ ----------- ----------------- ------------ --------------- ------------
16004  Pop         SR Pfx (idx 4)    TF0/0/0/24   14.1.0.2        104332
```

Use the **show performance-measurement sr-policy color 203** command to verify the performance measurement.

```
Router# show performance-measurement sr-policy color 203
Mon Mar 14 17:54:32.403 IST

------------------------------------------------------------------
0/RP0/CPU0
------------------------------------------------------------------

SR Policy name: srte_c_203_ep_1.1.1.1
Color : 203
Endpoint : 1.1.1.1
Number of candidate-paths : 1

Candidate-Path:
Instance : 8
Preference : 10
Protocol-origin : Configured
Discriminator : 10
Profile Keys:
Profile name : BLUE
Profile type : SR Policy Liveness Detection
Source address : 1.1.1.6
Number of segment-lists : 1
Liveness Detection: Enabled
Session State: Up
Last State Change Timestamp: Mar 14 2022 17:53:45.207
Missed count: 0

------------------------------------------------------------------
0/0/CPU0
------------------------------------------------------------------
```

Use the **show segment-routing traffic-eng policy color 10 tabular** command to verify the SR-TE configuration.

```
Router# show segment-routing traffic-eng policy color 10 tabular
Fri Nov 12 09:15:57.366 UTC
```

```
 Color                              Endpoint  Admin   Oper                Binding
                                              State  State                   SID
------ ---------------------------------------- ------ ------ ----------------------
    10                               1.1.1.4     up     up                   24010
```

Use the **show bgp l2vpn evpn neighbors brief** command to verify BGP EVPN neighbor session configuration.

```
Router# show bgp l2vpn evpn neighbors brief
Fri Nov 12 09:10:22.999 UTC

Neighbor        Spk    AS Description                          Up/Down  NBRState
1.1.1.4           0   100                                      15:51:52 Established
```

Use the **show l2vpn xconnect** command to verify the EVPN VPWS configuration.

```
Router# show l2vpn xconnect
Fri Nov 12 09:02:44.982 UTC
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
------------------------------------------------------------------------------------
evpn_vpws  p1         UP   CE0/0/1/0              UP       EVPN 10,1,24012        UP
------------------------------------------------------------------------------------
```

Use the **show policy-map targets** to verify QoS policy is configured.

This show command displays information about interfaces on which the policy maps are applied.

```
Router# show policy-map targets
Thu Jun 16 21:47:31.407 IST
1) Policymap: ple-p1 Type: qos
Targets (applied as main policy):
CEM0/0/1/0 input
Total targets: 1

Targets (applied as child policy):
Total targets: 0

2) Policymap: core Type: qos
Targets (applied as main policy):
TwentyFiveGigE0/0/0/24
Total targets: 1

Targets (applied as child policy):
Total targets: 0
```

Use **show policy-map interface TwentyFiveGigE0/0/0/24** command to view the core interface information and to verify the traffic class (TC) mapping in CEM interface.

```
Router# show policy-map interface TwentyFiveGigE0/0/0/24
Thu Jun 16 21:37:52.915 IST
TwentyFiveGigE0/0/0/24 direction input: Service Policy is not installed
TwentyFiveGigE0/0/0/24 output: core
Class tc2
    Classification Statistics    (packets/bytes)          (rate - kbps)
        Matched           :      39654778/42113374236        6816279
        Transmitted       :      39654778/42113374236        6816279
        Total Dropped     :             0/0                  0
    Queueing Statistics
        Queue ID                     : 1370
```

```
        Taildropped(packets/bytes)    : 0/0
Class class-default
    Classification Statistics    (packets/bytes)          (rate - kbps)
        Matched           :             0/0                  0
        Transmitted       :             0/0                  0
        Total Dropped     :             0/0                  0
    Queueing Statistics
        Queue ID                      : 1368
        Taildropped(packets/bytes)    : 0/0
Policy Bag Stats time: 1655395669491    [Local Time: 06/16/22 21:37:49:491]
```

Supported Yang data models for PLE

Here is the list of new and modified Yang data models supported for PLE. You can access the data models from the Github repository.

Configuration Files - New:

- Cisco-IOS-XR-controller-fc-cfg.yang
- Cisco-IOS-XR-fibrechannelmib-cfg.yang
- Cisco-IOS-XR-interface-cem-cfg.yang
- Cisco-IOS-XR-cem-class-cfg.yang

Configuration Files - Modified:

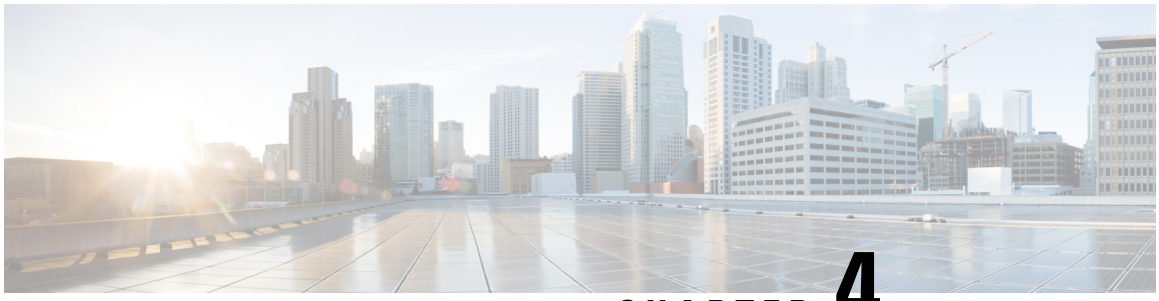
- Cisco-IOS-XR-controller-odu-cfg.yang
- Cisco-IOS-XR-controller-otu-cfg.yang
- Cisco-IOS-XR-controller-sonet-cfg.yang
- Cisco-IOS-XR-drivers-icpe-ethernet-cfg.yang

Operational Files - New:

- Cisco-IOS-XR-controller-fc-oper.yang
- Cisco-IOS-XR-interface-cem-oper.yang

Operational Files - Modified:

- Cisco-IOS-XR-controller-odu-oper.yang
- Cisco-IOS-XR-controller-otu-oper.yang



CHAPTER 4

EVPN MPLS Multihoming

EVPN MPLS multihoming modes

EVPN MPLS multihoming modes are redundancy techniques in EVPN deployments that

- connect a customer edge device to multiple provider edge devices
- provide redundant connectivity for enhanced reliability, and
- ensure uninterrupted traffic flow during network failures.

Modes of EVPN multihoming

These multihoming modes are supported:

- Single-active: Only one PE device in the group attached to the Ethernet segment forwards traffic to and from that segment. This mode prevents loops by allowing a single active forwarder.
- All-active: All PEs connected to the Ethernet segment are permitted to forward traffic simultaneously. This mode enables load sharing and active-active redundancy.
- Port-active: Traffic is sent and received only by the PE that is in active mode on a specific port or interface. This mode supports single-active redundancy with load balancing at the port or interface level.
- Single-flow-active: The PE that first advertises a host MAC address in a VLAN forwards traffic for that specific flow. This mode optimizes forwarding by directing each flow through a single active PE.

EVPN MPLS multihoming mode services

EVPN MPLS multihoming supports both E-LAN and E-LINE services.

This table compares the key features and use cases of EVPN E-LAN and E-LINE services.

Table 7: EVPN E-LAN and E-LINE services

Feature	EVPN E-LAN	EVPN E-Line
Connectivity type	Multipoint-to-Multipoint	Point-to-Point
Typical use cases	Multi-site connectivity, interconnecting multiple locations	Data center interconnects, connecting two sites
MAC address learning	Across all endpoints in the E-LAN	Limited to two endpoints
Supported modes	• All-active • Single-active • Port-active • Single flow-active	• All-active • Single-active • Port-active

While the configuration specifics differ between E-LAN and E-LINE services, their conceptual framework remains identical. Therefore, for conceptual understanding, the E-LAN documentation serves as the primary reference point for both services.

EVPN E-LAN single-active multihoming mode

EVPN E-LAN single-active multihoming mode is a network redundancy method that

- enables PE nodes locally connected to an Ethernet segment to load balance traffic to and from the segment based on an EVI
- ensures that within an EVI, only one PE forwards traffic to and from the Ethernet segment, and
- supports efficient use of network resources by managing active forwarding paths.

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-LAN single-active multihoming mode	Release 26.2.1	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
EVPN E-LAN single-active multihoming mode	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 • 8011-12G12X4Y-D
EVPN E-LAN single-active multihoming mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN E-LAN single-active multihoming mode	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The EVPN E-LAN single-active multi-homing functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN E-LAN single-active multihoming mode	Release 24.3.1	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*) *The EVPN E-LAN single-active multi-homing functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN E-LAN single-active multihoming mode	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The single-active multi-homing mode offers redundant connectivity on a single link at a time with failover to the second link in case the active link fails. In this mode, only a single PE among a group of PEs attached to an Ethernet segment forwards traffic to and from that Ethernet Segment. *This feature is supported only on routers with the 88-LC1-36EH line cards.

EVPN E-LAN single-active multihoming mode for redundant connectivity

EVPN E-LAN single-active multihoming mode provides network redundancy in EVPN environments, especially in ring topologies:

- Only one PE device, the active PE, handles traffic forwarding and reception for an Ethernet segment at a time, preventing network loops.

- When the active PE's link fails, traffic switches to a standby PE. During switchover, the standby PE learns the MAC addresses, which can cause a brief interruption.
- The standby PE quickly learns MAC addresses from the failed path, supporting rapid convergence and minimizing traffic loss.
- A CE device can connect to multiple PEs for redundancy. The first PE to advertise a host's MAC address in a VLAN becomes the active forwarder for that MAC.
- This mode also enables load balancing for traffic to and from the Ethernet segment, based on the EVI.

Benefits of single-active multihoming mode

Single-active multihoming mode offers several key advantages for network management and service provision:

- Redundant connectivity: Provides a backup link that automatically takes over if the active link fails, ensuring continuous service availability.
- Simplified traffic management: Directs traffic to a single uplink, simplifying the monitoring and management of data flows.
- Enhanced network control: Allows for precise control over bandwidth restrictions and data usage accounting, which is beneficial for implementing policing and metering.
- Billing flexibility: Facilitates accurate billing for business customers by integrating with internal billing systems, ensuring proper account management.
- High availability: Maintains a reliable connection by utilizing redundant paths, enhancing overall network resilience.

How EVPN E-LAN single-active multihoming mode works

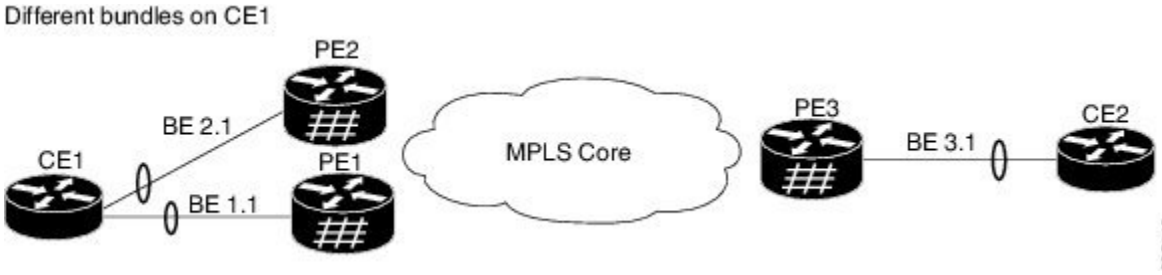
Summary

The EVPN E-LAN single-active multihoming process involves a network topology where a CE1 device is multihomed to two provider edge devices,PE1 and PE2, connected through an MPLS core to PE3. This process ensures efficient traffic flow and redundancy.

The key components involved in the process are:

- CE1: A customer edge device multihomed to PE1 and PE2.
- PE1 and PE2: Provider edge devices connected to CE1 and PE3, responsible for advertising routes and managing traffic flow.
- PE3: A provider edge device connected to CE2 through an Ethernet interface bundle.
- CE2: A customer edge device connected to PE3.

Workflow



The process involves the following stages:

1. Route advertisement:
   - PE1 and PE2 advertise Type 4 routes to elect the designated forwarder (DF). PE1 is elected as the DF, while PE2 becomes the non-DF.
2. Traffic management:
   - CE1 sends an ARP broadcast request to both PE1 and PE2.
   - PE1, as the DF, forwards the ARP request from CE1.
   - PE2, being the non-DF, drops the traffic from CE1.
3. Traffic flow:
   - All traffic is sent through PE1, with PE2 acting as a standby device.
   - PE1 advertises MAC routes to PE3.
   - PE3 sends and receives traffic through PE1 and forwards it to CE2 over the Ethernet interface bundle.
4. Redundancy management:
   - In case of a link failure where PE1 goes down, PE2 becomes active to maintain the traffic flow.

The EVPN E-LAN single-active multihoming process ensures efficient and reliable traffic flow while providing redundancy. When the active link (PE1) fails, the router automatically switches to the standby link (PE2) to maintain network connectivity.

Configure EVPN single-active multihoming mode

Set up EVPN single-active multihoming on PE routers for efficient network redundancy and load balancing.

This task involves configuring BGP sessions and MPLS LDP, setting up EVPN EVI parameters, and enabling single-active mode on PE routers.

Perform the following configuration on PE1, PE2, and PE3.

1. Configure BGP session and MPLS Label Distribution Protocol (LDP) to enable EVPN.
2. Configure bridge domain, EVI, and advertisement of MAC routes.
3. Enter the bundle interface mode and configure the Ethernet segment identifier (ESI) for the interface.

4. Ensure that you configure the same ESI on all the PEs.
5. Enable single-active mode by using the **load-balancing-mode single-active** command.

Procedure

Step 1 Configure BGP session and MPLS LDP to enable EVPN on PE routers.

Example:

PE1 configuration.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 54.54.54.54
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp)# neighbor 51.51.51.51
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp)# neighbor 55.55.55.55
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
```

Configure MPLS LDP.

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 55.55.55.55
Router(config-ldp)# interface FourHundredGigE0/0/0/2
```

PE2 configuration.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 55.55.55.55
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp)# neighbor 51.51.51.51
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp)# neighbor 54.54.54.54
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
```

Configure MPLS LDP.

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 55.55.55.55
Router(config-ldp)# interface FourHundredGigE0/0/0/2
```

PE3 configuration.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 51.51.51.51
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp)# neighbor 54.54.54.54
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp)# neighbor 55.55.55.55
Router(config-bgp-nbr)# remote-as 100
```

```
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
```

Configure MPLS LDP.

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 51.51.51.51
Router(config-ldp)# interface FourHundredGigE0/0/0/2
```

Step 2 Configure bridge domain, EVI, and advertise MAC routes.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1.1
Router(config-l2vpn-bg-bd-ac)# evi 1
Router(config-l2vpn-bg-bd-ac)# root
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg2
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1.2
Router(config-l2vpn-bg-bd-ac)# evi 2
```

Configure EVPN EVI parameters and advertise MAC routes.

```
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# exit
Router(config-evpn)# evi 2
Router(config-evpn-evi)# advertise-mac
```

Step 3 Enter the bundle interface mode and configure the same ESI on all the PE routers.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
```

Step 4 Enable single-active mode.

Example:

```
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# commit
```

Step 5 Running configuration of EVPN single-active multihoming mode.

Example:

```
/* PE1 Configuration */
router bgp 100
 bgp router-id 54.54.54.54
 address-family l2vpn evpn
 !
 neighbor 51.51.51.51
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
```

```
 neighbor 55.55.55.55
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
mpls ldp
 router-id 54.54.54.54
 interface FourHundredGigE0/0/0/2
 !
!

/* PE2 Configuration */
router bgp 100
 bgp router-id 55.55.55.55
 address-family l2vpn evpn
 !
 neighbor 51.51.51.51
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
 neighbor 54.54.54.54
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
mpls ldp
 router-id 55.55.55.55
 interface FourHundredGigE0/0/0/2
 !
!

/* PE3 Configuration */
router bgp 100
 bgp router-id 51.51.51.51
 address-family l2vpn evpn
 !
 neighbor 54.54.54.54
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
 neighbor 55.55.55.55
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
mpls ldp
 router-id 51.51.51.51
 interface FourHundredGigE0/0/0/3
 !
!
```

Configuration of bridge domain, EVI, and advertise MAC routes on all the PEs.

```
l2vpn
 bridge group bg1
  bridge-domain bd1
   interface Bundle-Ether1.1
   !
   evi 1
   !
  !
 !
 bridge group bg2
  bridge-domain bd2
   interface Bundle-Ether1.2
   !
   evi 2
   !
  !
 !
!
evpn
 evi 1
  advertise-mac
  !
 !
 evi 2
  advertise-mac
  !
 !
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 40.00.00.00.00.00.00.00.01
   load-balancing-mode single-active
  !
 !
!
```

Step 6 Use the **show evpn ethernet-segment interface BE1 carving detail** to verify that single-active mode is configured on PE1.

Example:

```
Router# show evpn ethernet-segment interface BE1 carving detail

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0040.0000.0000.0000.0001 BE1                               54.54.54.54
                                                           55.55.55.55
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether1
     Interface MAC  : 008d.9c38.7205
     IfHandle       : 0x0f00003c
     State          : Up
     Redundancy     : Not Defined
  ESI ID            : 1
  ESI type          : 0
     Value          : 0040.0000.0000.0000.0001
  ES Import RT      : 4000.0000.0000 (from ESI)
  Topology          :
     Operational    : MH, Single-active
     Configured     : Single-active (AApS)
  Service Carving   : Auto-selection
     Multicast      : Disabled
```

```
Convergence       :
Peering Details   : 2 Nexthops
   54.54.54.54 [MOD:P:00:T]
   55.55.55.55 [MOD:P:00:T]
Service Carving Synchronization:
   Mode           : NONE
   Peer Updates   :
           54.54.54.54 [SCT: N/A]
           55.55.55.55 [SCT: 2024-03-12 10:42:30.1710254]
Service Carving Results:
   Forwarders     : 2
   Elected        : 1
          EVI E   :        2
   Not Elected    : 1
          EVI NE  :        1
EVPN-VPWS Service Carving Results:
   Primary        : 0
   Backup         : 0
   Non-DF         : 0
MAC Flush msg     : STP-TCN
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Revert timer      : 0 sec [not running]
HRW Reset timer   : 5 sec [not running]
Local SHG label   : 24004
Remote SHG labels : 1
            24004 : nexthop 55.55.55.55
Access signal mode: Bundle OOS
```

The EVPN single-active multihoming configuration is applied across PE routers, ensuring optimized redundancy and load balancing.

Disabling MAC flush messages is particularly relevant in the context of EVPN single-active multihoming mode, as it helps prevent network disruptions at the CE by mitigating issues like BPDU guard triggers when configuring multihoming scenarios.

MAC flush message disablement

Disabling MAC flush messages is an EVPN E-LAN feature that

- prevents MAC flush messages from being sent for an Ethernet segment
- addresses undesired behavior such as triggering BPDU guard at the CE, and
- is implemented using the **mac-flush-message disable** command during the configuration of EVPN single-active multihoming on PE routers.

Disable MAC flush messages for EVPN single-active multihoming mode

Set up EVPN and L2VPN configurations to manage Ethernet segments effectively with MAC flush messages disabled.

This task is performed to configure EVPN and L2VPN settings on a router to handle Ethernet segments with specific identifiers and load balancing mode. It includes setting up bridge groups and domains, configuring interfaces, and ensuring MAC flush messages are disabled for the specified segments

Procedure

Step 1 Configure L2VPN.

Example:

```
Router(config)#l2vpn
Router(config-l2vpn)#bridge group 100
Router(config-l2vpn-bg)#bridge-domain 100
Router(config-l2vpn-bg-bd)#interface Bundle-Ether2.10
Router(config-l2vpn-bg-bd)#evi 100
Router(config-l2vpn-bg-bd-evi)#commit
```

Step 2 Configure EVPN single-active multihoming mode and disable MAC flush message.

Example:

```
Router(config)#evpn
Router(config-evpn)#evi 100
Router(config-evpn-instance)#advertise-mac
Router(config-evpn-instance-mac)#exit
Router(config-evpn-instance)#exit
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 36.37.00.00.00.00.00.11.00
Router(config-evpn-ac-es)#load-balancing-mode single-active
Router(config-evpn-ac-es)#exit
Router(config-evpn-ac)# mac-flush-message disable
Router(config-evpn-ac)# commit
```

Step 3 Running configuration.

Example:

```
l2vpn
 bridge group 100
  bridge-domain 100
   interface Bundle-Ether1.10
   !
   evi 100
   !
  !
 !
!
evpn
 evi 100
  advertise-mac
  !
 !
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 36.37.00.00.00.00.00.11.00
   load-balancing-mode single-active
  !
  mac-flush-message disable
  !
```

```
 !
!
```

Step 4 Use the **show evpn ethernet-segment detail** command to verify the MAC flush message status.

Example:

```
Router#show evpn ethernet-segment detail

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

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0036.3700.0000.0000.1100 BE1                               10.1.1.1
                                                           10.2.2.2
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether1
     Interface MAC  : 0008.3302.3208
     IfHandle       : 0x02000160
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 36.3700.0000.0000.1100
  ES Import RT      : 3637.0000.0000 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
    Operational     : MH, Single-active
     Configured     : Single-active (AApS)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       :
     Mobility-Flush : Count 0, Skip 0, Last n/a
  Peering Details   : 2 Nexthops
     10.1.1.1 [MOD:P:00]
     10.2.2.2 [MOD:P:00]
 Service Carving Results:
     Forwarders     : 1
     Elected        : 1
     Not Elected    : 0
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flush msg     : Disabled
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : 24007
```

```
Remote SHG labels : 1
           24007 : nexthop 10.2.2.2
Access signal mode: Bundle OOS (Default)
```

Check that the MAC flush message is disabled in the output.

EVPN E-LAN all-active multihoming mode

An EVPN E-LAN all-active multihoming mode is a networking model that

- enables multiple simultaneous active connections from an EVPN to a single Ethernet LAN
- allows all PE routers attached to a particular Ethernet segment to forward traffic to and from that Ethernet segment, and
- provides redundancy and load balancing by allowing traffic distribution across all available links, enhancing network reliability and efficiency by utilizing all potential paths without requiring a failover mechanism.

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-LAN all-active multihoming mode	Release 26.2.1	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 E-LAN all-active multihoming mode	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 • 8011-12G12X4Y-D
EVPN E-LAN all-active multihoming mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN E-LAN all-active multihoming mode	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The EVPN E-LAN all-active multi-homing functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
E-LAN all-active multihoming mode	Release 24.3.1	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*) * The EVPN E-LAN all-active multi-homing functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
E-LAN all-active multihoming mode	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The all-active multi-homing mode enables redundant network connectivity by allowing a CE device to connect to more than one PE device. In this mode, all the links actively forward the traffic. * This feature is supported only on routers with the 88-LC1-36EH line cards.

Business connectivity challenges

Consider a scenario where the enterprise XYZ is operating in two different cities, with its headquarters (HQ) in Denver and a branch office in Dallas.

The business customer hosts all voice and video services in their active data center at HQ, Denver. The branch site in Dallas has minimal services for regional operations and might need to offload certain tasks to HQ. Additionally, the HQ site is used to update storage over a high-speed connection. Enterprise XYZ has approached the service provider for Layer 2 connectivity services, requiring direct L2 services for data center workloads at both HQ and the branch site.

The HQ in Denver is critical for the business, necessitating maximum throughput and high availability. The remote branch in Dallas can operate with a single uplink.

All-active multihoming as the ideal solution

To address these requirements, the service provider plans to deploy Layer 2 connectivity that seamlessly balances workloads across multiple sites while providing reliable and redundant connections with high availability. EVPN all-active multihoming mode is chosen as the solution for this scenario.

In all-active multihoming mode, both the PE devices attached to the Ethernet segment are allowed to receive and send traffic. This mode ensures redundant connectivity with no traffic disruption in the event of a network failure. It also offers high availability, efficient bandwidth utilization, and faster convergence, making it ideal for scenarios like HQ in Denver where maximum throughput and reliability are essential.

How EVPN E-LAN all-active multihoming works

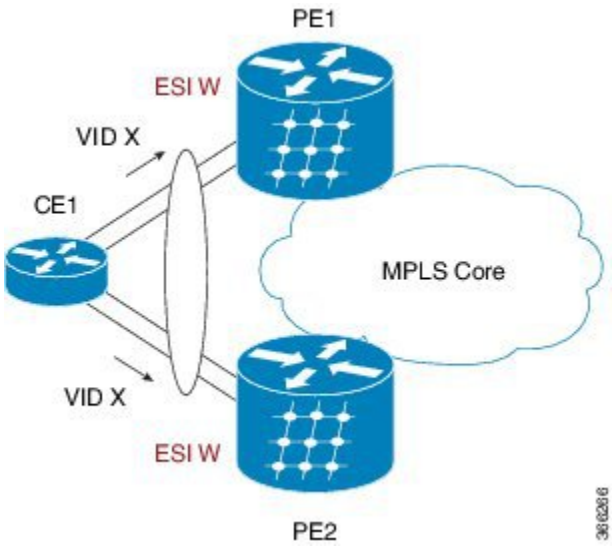
Summary

The EVPN E-LAN all-active multihoming mode enables load balancing and redundancy by allowing multiple PE devices to forward traffic concurrently. PE devices use identical Ethernet Segment Identifiers (ESIs) and bundle interfaces to achieve this.

The key components involved in the process are:

- CE1: Single bundles towards two EVPN PE devices, allowing connectivity to the MPLS core.
- PE1 and PE2: Utilize identical ESIs and attach to the Ethernet segment using bundle interfaces, enabling simultaneous traffic forwarding within the same EVI.
- MPLS core: The central network component that provides connectivity and routes traffic between PEs and other network entities.

Workflow



The process involves these stages:

1. Configuration of CE1: CE1 is configured with single bundle interfaces directed towards both PE1 and PE2.
2. ESI configuration on PEs: Both PE1 and PE2 are configured with identical ESIs, allowing them to recognize and manage the same Ethernet segment.
3. Traffic forwarding: In this all-active mode, both PE1 and PE2 can concurrently forward traffic within the same EVI, providing load balancing and redundancy.
4. Load balancing: Traffic is distributed across both PEs, leveraging the Active/Active per Flow (AApF) mechanism for optimal utilization of resources.

Result

The EVPN E-LAN all-active multihoming mode ensures efficient load balancing, increased redundancy, and robust connectivity within the MPLS network by enabling simultaneous traffic forwarding through multiple PEs.

Configure EVPN E-LAN all-active multihoming mode

Set up an EVPN E-LAN all-active multihoming mode on PE1 and PE2.

This task involves configuring BGP sessions, MPLS Label Distribution Protocol (LDP), and EVPN EVI parameters on both PE1 and PE2 to enable EVPN E-LAN all-active multihoming.

Before you begin

- Ensure all equipment is powered on and accessible.
- Verify network connectivity between PE1 and PE2.
- You must configure BGP when you set up EVPN with an EVI under a bridge-domain because BGP enables EVPN to work properly. Without BGP, the EVPN instance only functions locally and cannot connect to remote PEs.

Procedure

Step 1 Configure BGP session and MPLS LDP on PE1.

Example:

Configure BGP session on PE1.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 54.54.54.54
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp)# neighbor 51.51.51.51
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp)# neighbor 55.55.55.55
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
```

Configure MPLS LDP on PE1.

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 54.54.54.54
Router(config-ldp)# interface FourHundredGigE0/0/0/2
```

Step 2 Configure BGP session and MPLS LDP on PE2.

Example:

Configure BGP session on PE2.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 55.55.55.55
Router(config-bgp)# address-family l2vpn evpn
```

```
Router(config-bgp)# neighbor 51.51.51.51
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp)# neighbor 54.54.54.54
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
```

Configure MPLS LDP on PE2.

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 55.55.55.55
Router(config-ldp)# interface FourHundredGigE0/0/0/2
```

Step 3 Configure bridge domain and EVI on PE1 and PE2.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface Bundle-Ether11.1
Router(config-l2vpn-bg-bd-ac)# evi 1
Router(config-l2vpn-bg-bd-ac)# root
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg2
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface Bundle-Ether11.2
Router(config-l2vpn-bg-bd-ac)# evi 2
```

Step 4 Configure EVPN EVI and advertise MAC routes on PE1 and PE2.

Example:

```
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# exit
Router(config-evpn)# evi 2
Router(config-evpn-evi)# advertise-mac
```

Step 5 Configure the same ESI on PE1 and PE2.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether11
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# commit
```

Step 6 EVPN E-LAN all-active multihoming mode running configuration.

Example:

```
/* PE1 Configuration */
router bgp 100
 bgp router-id 54.54.54.54
 address-family l2vpn evpn
 !
 neighbor 51.51.51.51
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
```

```
  !
 !
 neighbor 55.55.55.55
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
mpls ldp
 router-id 54.54.54.54
 interface FourHundredGigE0/0/0/2
 !
!/* PE2 Configuration */
router bgp 100
 bgp router-id 55.55.55.55
 address-family l2vpn evpn
 !
 neighbor 51.51.51.51
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
 neighbor 54.54.54.54
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
mpls ldp
 router-id 55.55.55.55
 interface FourHundredGigE0/0/0/2
 !
!
/* Configuration on PE1 and PE2 */
l2vpn
 bridge group bg1
  bridge-domain bd1
   interface Bundle-Ether11.1
   !
   evi 1
   !
  !
 !
 bridge group bg2
  bridge-domain bd2
   interface Bundle-Ether11.2
   !
   evi 2
   !
  !
 !
!
evpn
 evi 1
  advertise-mac
  !
 !
 evi 2
  advertise-mac
  !
 !
```

```
 interface Bundle-Ether11
  ethernet-segment
   identifier type 0 40.00.00.00.00.00.00.00.01
  !
 !
!
```

Step 7 Use the **show evpn ethernet-segment int Bundle-Ether 11 carving detail** command to verify that EVPN all-active multihoming mode is configured on PE1 and PE2.

Example:

```
Router#show evpn ethernet-segment int Bundle-Ether 11 carving detail

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0040.0000.0000.0000.0001 BE11                               54.54.54.54
                                                            55.55.55.55
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether11
     Interface MAC  : 008d.9c38.7205
     IfHandle       : 0x0f00001c
     State          : Up
     Redundancy     : Not Defined
  ESI ID            : 1
  ESI type          : 0
     Value          : 0040.0000.0000.0000.0001
  ES Import RT      : 4000.0000.0000 (from ESI)
  Topology          :
     Operational    : MH, All-active
     Configured     : All-active (AApF) (default)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       :
  Peering Details   : 2 Nexthops
     54.54.54.54 [MOD:P:00:T]
     55.55.55.55 [MOD:P:00:T]
  Service Carving Synchronization:
     Mode           : NONE
     Peer Updates   :
              54.54.54.54 [SCT: N/A]
              55.55.55.55 [SCT: N/A]
  Service Carving Results:
     Forwarders     : 2
     Elected        : 1
            EVI E   :        2
     Not Elected    : 1
            EVI NE  :        1
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flush msg     : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Revert timer      : 0 sec [not running]
  HRW Reset timer   : 5 sec [not running]
  Local SHG label   : 24004
  Remote SHG labels : 1
```

```
            24004 : nexthop 55.55.55.55
Access signal mode: Bundle OOS
```

EVPN E-LAN port-active multihoming mode

An EVPN E-LAN port-active multihoming is a network model that

- supports single-active redundancy load balancing at the port-level or interface-level
- provides faster convergence during a link failure, and
- enables protocol simplification by having only one physical port active at a given time.

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-LAN port-active multihoming mode	Release 26.2.1	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 E-LAN port-active multihoming mode	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
EVPN E-LAN port-active multihoming mode	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN E-LAN port-active multihoming mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
EVPN E-LAN port-active multihoming mode	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q200])(select variants only*) *The EVPN E-LAN port-active multi-homing mode is now extended to: • 8712-MOD-M • 8201-32FH • 8201-24H8FH • 8202-32FH-M • 8608 • 88-LC0-34H14FH • 88-LC0-36FH • 88-LC0-36FH-M
EVPN E-LAN port-active multihoming mode	Release 24.3.1	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*) *The EVPN E-LAN port-active multi-homing is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN E-LAN port-active multihoming mode	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The port-active multi-homing mode enables single-active redundancy load balancing at the port-level or the interface-level. In this mode, one of the PEs remains active at the port-level. *This feature is supported only on routers with the 88-LC1-36EH line cards.

Port-active mode for network traffic management

Port-active mode streamlines operations by providing a simpler alternative to ICCP-based MC-LAG solutions. This mode allows you to:

- enable precise control over QoS features by ensuring only one PE device actively forwards traffic at a time
- reduce complexity and operational overhead compared to multichassis approaches that rely on ICCP and LDP, and

- improve overall traffic management efficiency by using selective port activation and a clear standb-active model.

Active and standby mode operation at port level

The EVPN E-LAN port-active mode enables one PE to be active and another to be standby on a per-port basis. Only the active PE sends and receives traffic, while the standby PE remains passive. The designated forwarder (DF) election mechanism ensures proper role assignment, supporting either modulo or HRW algorithms for per-port elections. By default, the modulo algorithm is used for per port DF election.

Benefits of EVPN E-LAN port-active multihoming

The benefits of this mode include protocol simplification, efficient traffic handling, improved QoS support, robust role assignment, and enhanced network stability, as demonstrated by:

- Protocol simplification: Eliminates the need for ICCP, reducing protocol complexity and operational overhead.
- Efficient traffic handling: Only the active PE port handles traffic, preventing duplication and ensuring clear traffic paths.
- Improved QoS support: Facilitates QoS features that require a single active forwarding point per port.
- Robust role assignment:Uses a Designated Forwarder (DF) election mechanism with either modulo or Highest Random Weight (HRW) algorithms to dynamically assign active and standby roles per port.
- Enhanced network stability: By having a standby PE port ready to take over, it improves network resilience and failover capabilities.

This approach streamlines network design and operation while maintaining high availability and performance at the port level.

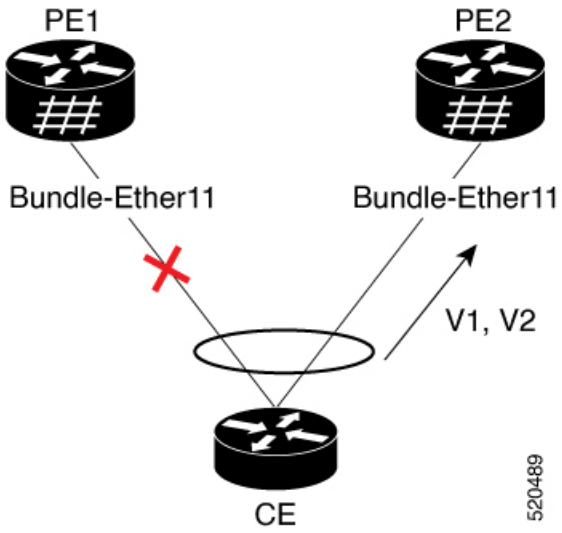
How EVPN E-LAN port-active multihoming mode works

Summary

The process of aggregating links in a multihomed topology involves managing traffic flow between customer edge and provider edge devices. The key components involved in the process are:

- CE: Utilizes single link aggregation and connects to multiple provider edge devices.
- PE1: Initially in standby mode; its interface is not forwarding traffic.
- PE2: Initially in active mode; its interface forwards traffic from the CE.

Workflow



The process involves these stages:

1. Initial configuration:
   - The CE connects to PE1 and PE2 using link aggregation.
   - Only one CE interface forwards traffic, with the other in standby.
2. Traffic management:
   - PE2 operates in active mode, carrying traffic from the CE.
   - PE1 remains in standby mode, not carrying traffic.
3. Configuration changes:
   - Removing port-active configuration from both PE1 and PE2.
   - Re-adding port-active configuration to both PEs.
4. Interface selection:
   - After reconfiguration, PE2 is chosen as the active interface again.

Result

The process ensures efficient traffic management in multihomed topologies, with PE2 consistently selected as the active PE device, maintaining optimal service operations and connectivity.

Configure EVPN port-active multihoming mode

Set up EVPN port-active multihoming on PE routers to enable efficient load balancing and redundancy.

This configuration is essential for deploying EVPN in a network environment where multihoming is required, enabling single-active mode for efficient resource use.

Before you begin

- Ensure you have access to PE1 and PE2 with administrative privileges.

Procedure

Step 1 Configure BGP session and MPLS LDP on PE1.

Example:

Configure BGP session.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 55.55.55.55
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp)# neighbor 51.51.51.51
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp)# neighbor 54.54.54.54
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
```

Configure MPLS LDP .

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 55.55.55.55
Router(config-ldp)# interface FourHundredGigE0/0/0/2
```

Step 2 Configure BGP session and MPLS LDP on PE2.

Example:

Configure BGP session.

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 54.54.54.54
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp)# neighbor 51.51.51.51
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp)# neighbor 55.55.55.55
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
```

Configure MPLS LDP .

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 54.54.54.54
Router(config-ldp)# interface FourHundredGigE0/0/0/2
```

Step 3 Configure bridge domain and EVI on PE1 and PE2.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface Bundle-Ether11.1
```

```
Router(config-l2vpn-bg-bd-ac)# evi 1
Router(config-l2vpn-bg-bd-ac)# root
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg2
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface Bundle-Ether11.2
Router(config-l2vpn-bg-bd-ac)# evi 2
```

Step 4 Configure EVPN EVI and advertise the MAC routes on PE1 and PE2.

Example:

```
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# exit
Router(config-evpn)# evi 2
Router(config-evpn-evi)# advertise-mac
```

Step 5 Configure the same ESI on all the PE routers.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether11
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode port-active
Router(config-evpn-ac-es)# commit
```

Step 6 EVPN port-active multihoming mode running configuration.

Example:

```
/* PE1 Configuration */
router bgp 100
 bgp router-id 55.55.55.55
 address-family l2vpn evpn
 !
 neighbor 51.51.51.51
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
 neighbor 54.54.54.54
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
mpls ldp
 router-id 55.55.55.55
 interface FourHundredGigE0/0/0/2
 !
!
/* PE2 Configuration */
router bgp 100
 bgp router-id 54.54.54.54
 address-family l2vpn evpn
 !
 neighbor 51.51.51.51
  remote-as 100
```

```
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
 neighbor 55.55.55.55
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
mpls ldp
 router-id 54.54.54.54
 interface FourHundredGigE0/0/0/2
 !
!
/* Configuration on all the PEs */
l2vpn
 bridge group bg1
  bridge-domain bd1
   interface Bundle-Ether11.1
   !
   evi 1
   !
  !
 !
 bridge group bg2
  bridge-domain bd2
   interface Bundle-Ether11.2
   !
   evi 2
   !
  !
 !
!
evpn
 evi 1
  advertise-mac
  !
 !
 evi 2
  advertise-mac
  !
 !
 interface Bundle-Ether11
  ethernet-segment
   identifier type 0 40.00.00.00.00.00.00.00.01
   load-balancing-mode port-active
  !
 !
!
```

Step 7 Use the **show bundle BE11** command to verify that port-active mode is configured.

Example:

Verify that PE2 is active and the status shows as Up.

```
Router# show bundle BE11
Bundle-Ether11
  Status:                                  Up
  Local links <active/standby/configured>:   1 / 0 / 1
  Local bandwidth <effective/available>:     400000000 (400000000) kbps
  MAC address (source):                      008d.9c38.7205 (Chassis pool)
  Inter-chassis link:                        No
```

```
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      64
  Wait while timer:                          2000 ms
  Load balancing:
    Link order signaling:                    Not configured
    Hash type:                               Default
    Locality threshold:                      None
  LACP:                                      Operational
    Flap suppression timer:                  Off
    Cisco extensions:                        Disabled
    Non-revertive:                           Disabled
  mLACP:                                     Not configured
  IPv4 BFD:                                  Not configured
  IPv6 BFD:                                  Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  FH0/0/0/3             Local            Active       0x8000, 0x0001   400000000
      Link is Active
```

Verify that PE1 is in standby mode.

```
Router#show bundle BE11
Bundle-Ether11
  Status:                                    EVPN Hot-Standby
  Local links <active/standby/configured>:   0 / 1 / 1
  Local bandwidth <effective/available>:     0 (0) kbps
  MAC address (source):                      003f.ee3b.5a05 (Chassis pool)
  Inter-chassis link:                        No
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      64
  Wait while timer:                          2000 ms
  Load balancing:
    Link order signaling:                    Not configured
    Hash type:                               Default
    Locality threshold:                      None
  LACP:                                      Operational
    Flap suppression timer:                  Off
    Cisco extensions:                        Disabled
    Non-revertive:                           Disabled
  mLACP:                                     Not configured
  IPv4 BFD:                                  Not configured
  IPv6 BFD:                                  Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  FH0/0/0/6             Local            Standby      0x8000, 0x0001   400000000
      Link is in standby due to bundle out of service state
```

This output shows the port-active mode configuration.

```
Router#show evpn ethernet-segment int Bundle-Ether 11 carving detail

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0040.0000.0000.0000.0001 BE11                              54.54.54.54
                                                           55.55.55.55
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether11
     Interface MAC  : 008d.9c38.7205
```

```
   IfHandle       : 0x0f00005c
   State          : Up
   Redundancy     : Not Defined
ESI ID            : 1
ESI type          : 0
   Value          : 0040.0000.0000.0000.0001
ES Import RT      : 4000.0000.0000 (from ESI)
Topology          :
   Operational    : MH
   Configured     : Port-Active
Service Carving   : Auto-selection
   Multicast      : Disabled
Convergence       :
Peering Details   : 2 Nexthops
   54.54.54.54 [MOD:P:00:T]
   55.55.55.55 [MOD:P:00:T]
Service Carving Synchronization:
   Mode           : NTP_SCT
   Peer Updates   :
           54.54.54.54 [SCT: 2024-03-12 10:58:28.1710255]
           55.55.55.55 [SCT: 2024-03-12 10:58:47.1710255]
Service Carving Results:
   Forwarders     : 2
   Elected        : 2
          EVI E   :        1,       2
   Not Elected    : 0
EVPN-VPWS Service Carving Results:
   Primary        : 0
   Backup         : 0
   Non-DF         : 0
MAC Flush msg     : STP-TCN
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Revert timer      : 0 sec [not running]
HRW Reset timer   : 5 sec [not running]
Local SHG label   : 24004
Remote SHG labels : 1
            24004 : nexthop 55.55.55.55
Access signal mode: Bundle Hot-Standby
```

EVPN E-LAN single-flow-active multihoming mode

An EVPN E-LAN single-flow-active multihoming mode is a networking architecture that

- provides redundancy and load balancing for Ethernet VPNs by allowing multiple links from a customer site to connect to a service provider network
- ensures that only one link is active for a given flow at any time, thereby preventing loops and ensuring efficient traffic management, and
- offers seamless failover capabilities, automatically switching to another link if the active link fails.

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-LAN Single-Flow-Active Multi-Homing	Release 26.2.1	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 E-LAN Single-Flow-Active Multi-Homing	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 • 8011-12G12X4Y-D
EVPN E-LAN Single-Flow-Active Multi-Homing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN E-LAN Single-Flow-Active Multi-Homing	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The EVPN E-LAN single-flow-active multi-homing functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN E-LAN Single-Flow-Active Multi-Homing	Release 24.3.1	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*) *The EVPN E-LAN single-flow-active multi-homing 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
EVPN E-LAN Single-Flow-Active Multi-Homing	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100])(select variants only*) This feature introduces EVPN E-LAN single-flow-active multi-homing load balancing mode to connect PE devices in an access network that run Layer 2 access gateway protocols. In this mode, only the PE that first advertises the host MAC address in a VLAN forwards the traffic in a specific flow. When the primary link fails, the traffic quickly switches to the standby PE that learns the MAC address from the originated path, thereby providing fast convergence. The feature introduces the **load-balancing-mode** command with keyword, **single-flow-active**. *This feature is supported only on routers with the 88-LC1-36EH line cards.

Traffic management in EVPN E-LAN single-flow-active multihoming mode

EVPN E-LAN single-flow-active multihoming mode optimizes traffic management in ring topologies by improving failover response and reducing traffic loss in case of link failure.

Key points

In traditional ring topologies:

- Only one PE device, the active PE, handles all traffic to prevent loops.
- If the active PE fails, the standby PE takes over, but there is a delay because it must learn the MAC addresses from connected hosts.
- This learning process causes temporary traffic loss until switchover completes.

With EVPN E-LAN single-flow-active multihoming mode:

- PE devices are connected to the access network to provide seamless switchover.
- Upon failure of the active link, traffic immediately switches to the standby PE.
- Immediate switchover minimizes traffic loss.

MAC address learning in EVPN E-LAN single-flow-active multihoming mode

Both active and standby PEs learn the MAC addresses of the connected hosts. The PE that learns the MAC address of the host directly is called the primary (active) PE. The primary PE advertises the learned MAC addresses to the peer PE, referred to as the standby PE. As the standby PE learns the MAC address of the host through the active PE, this learned path is referred to as the reoriginated path.

Fast convergence in EVPN E-LAN single-flow-active multihoming mode

When the primary link fails, convergence happens quickly, and traffic is sent through the standby PE through the reoriginated path.

Limitations of EVPN E-LAN single-flow-active multihoming mode

These limitations apply to the EVPN E-LAN single-flow-active multihoming mode:

- The EVPN E-LAN single-flow active multihoming mode is not supported for EVPN VPWS.
- The EVPN E-LAN single-flow-active multihoming mode is not supported on the Q100 and Q200 based systems.
- Starting from Release 24.1.1, only the G.8032 is supported for EVPN E-LAN single-flow-active multihoming mode.

How EVPN E-LAN single flow-active mode works

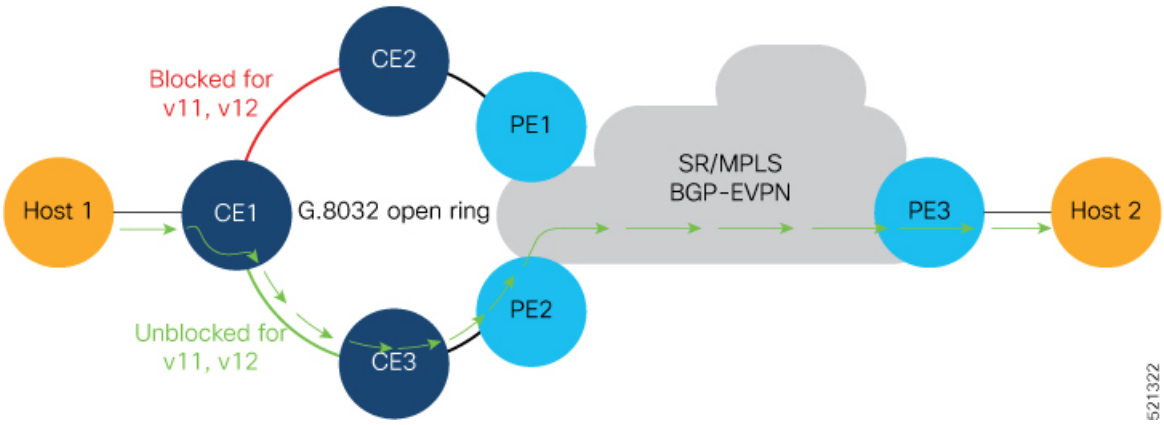
Summary

EVPN E-LAN single flow-active mode enables fast convergence in a ring topology, ensuring efficient network traffic management by utilizing multihoming and dynamic local preference settings. The key components involved in this process are:

- Host 1: Connected to CE1, sending traffic across the network.
- CE1: Multihomed and connected to both PE1 and PE2.
- PE1 and PE2: Multihoming devices configured with the same Ethernet Segment ID and EVPN instance.

Workflow

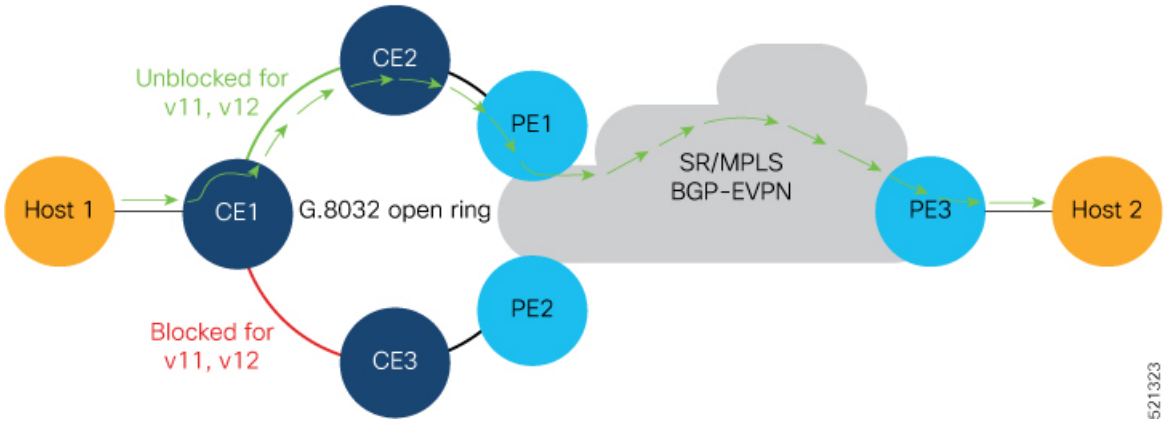
Figure 4: How EVPN E-LAN single flow-active mode works



The process involves these stages:

1. Traffic flow initiation:
    - Host 1 sends traffic to CE1.
    - CE1 sends traffic to PE2 through CE3, as the CE1-CE3 link is in the forwarding state.
2. MAC address learning:
    - PE2 learns Host 1's MAC address and advertises it to PE1.
    - PE2, acting as the active PE, sets the BGP local preference value to 100.
3. Traffic routing:
    - PE1, acting as the stand-by PE, sets its BGP local preference to 80.
    - PE1 sends traffic to PE3, which forwards it to Host 2.
4. Failure scenario:
    - If the CE1-CE3 or CE3-PE2 link fails, traffic reroutes through PE1.
    - CE1-CE2 link changes to the forwarding state.
    - PE1 learns Host 1’s MAC address directly, sets its BGP local preference to 100, and routes traffic to Host 2 through PE3.

Figure 5: Failure Scenario



Result

The EVPN E-LAN single flow-active mode ensures fast convergence by dynamically adjusting local preferences and multihoming configurations, allowing efficient traffic management and minimizing network disruption during link failures.

Configure EVPN E-LAN single-flow-active multihoming

Set up EVPN E-LAN with single-flow-active multihoming across PE1 and PE2 routers.

This task involves configuring EVPN instances on both routers to enable single-flow-active mode for enhanced load balancing and redundancy.

Procedure

Step 1 Configure advertisement of MAC routes.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit
```

Configure both PE1 and PE2 with the same EVI of 100.

Step 2 Configure single-flow-active load-balancing mode.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface bundle-ether 1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 36.37.00.00.00.00.00.11.01
Router(config-evpn-ac-es)# load-balancing-mode single-flow-active
Router(config-evpn-ac-es)# commit
```

Configure both PE1 and PE2 with the same ESI 0 36.37.00.00.00.00.00.11.01.

Step 3 Configure bridge domain and associate the evi to the bridge domain.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 100
Router(config-l2vpn-bg)# bridge-domain 100
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1.2
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)# evi 100
Router(config-l2vpn-bg-bd-evi)# root
Router(config)# interface Bundle-Ether1.2 l2transport
Router(config-l2vpn-subif)#encapsulation dot1q 2
Router(config-l2vpn-subif)#commit
```

Step 4 Running configuration of EVPN E-LAN single-flow-active multihoming.

Example:

```
evpn
 evi 100
  advertise-mac
  !
 !
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 36.37.00.00.00.00.00.11.01
   load-balancing-mode single-flow-active
  convergence
      mac-mobility
   !
  !
 !
!
l2vpn
  bridge group 100
   bridge-domain 100
    interface Bundle-Ether1
   !
   evi 100
   !
  !
 !
 interface Bundle-Ether1.2 l2transport
  encapsulation dot1q 2
!
!
```

Step 5 Use the **show evpn ethernet-segment interface be 1 detail** to verify that EVPN E-LAN single-flow-active multihoming mode is configured.

Example:

```
Router#show  evpn  ethernet-segment  interface  be 1 detail
Legend:
B   - No Forwarders EVPN-enabled,
C   - MAC missing (Backbone S-MAC PBB-EVPN / Grouping ES-MAC vES),
RT  - ES-Import Route Target missing,
E   - ESI missing,
H   - Interface handle missing,
I   - Name (Interface or Virtual Access) missing,
M   - Interface in Down state,
O   - BGP End of Download missing,
P   - Interface already Access Protected,
```

```
Pf  - Interface forced single-homed,
R   - BGP RID not received,
S   - Interface in redundancy standby state,
X   - ESI-extracted MAC Conflict
SHG - No local split-horizon-group label allocated
Hp  - Interface blocked on peering complete during HA event
Rc  - Recovery timer running during peering sequence

Ethernet Segment Id            Interface                          Nexthops
0 36.37.00.00.00.00.00.11.01  BE1                                 172.16.0.4
                                                                  172.16.0.5
ES to BGP Gates   : Ready
ES to L2FIB Gates : P
Main port         :
Interface name    : Bundle-Ether1
Interface MAC     : b0a6.51e5.00dd
IfHandle          : 0x2000802c
State             : Up
Redundancy        : Not Defined
ESI type          : 0
Value             : 07.0807.0807.0807.0800
ES Import RT      : 0708.0708.0708 (from ESI)
Source MAC        : 0000.0000.0000 (N/A)
Topology          :
Operational     : MH, Single-flow-active
Configured      : Single-flow-active
Service Carving   : Auto-selection
Multicast         : Disabled
Convergence       : MAC-Mobility
Mobility-Flush    : Debounce 1 sec, Count 0, Skip 0
                  : Last n/a
Peering Details   : 2 Nexthops
172.16.0.4 [MOD:P:00:T]
172.16.0.5 [MOD:P:00:T]
Service Carving Synchronization:
Mode              : NONE
Peer Updates      :
172.16.0.4 [SCT: N/A]
172.16.0.5 [SCT: N/A]
Service Carving Results:
Forwarders    : 1
Elected         : 0
Not Elected   : 0
EVPN-VPWS Service Carving Results:
Primary           : 0
Backup            : 0
Non-DF            : 0
MAC Flushing mode: STP-TCN
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
HRW Reset timer   : 5 sec [not running]
Local SHG label   : 24007
Remote SHG labels: 1
24010             : nexthop 172.16.0.5
Access signal mode: Bundle OOS (Default)
```

EVPN E-Line single-active multihoming mode

EVPN E-Line single-active multihoming mode is a network redundancy model that

- enables PE nodes locally connected to an Ethernet segment to load balance traffic to and from the segment based on an EVI
- ensures that within an EVI, only one PE forwards traffic to and from the Ethernet segment, and
- supports efficient use of network resources by managing active forwarding paths.

The EVPN E-Line single-active multihoming mode uses the same conceptual framework as the EVPN E-LAN single-active multihoming mode. For more details, refer to the EVPN E-LAN single-active multihoming mode section.

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-Line single-active multihoming mode	Release 26.2.1	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 E-Line single-active multihoming mode	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
EVPN E-Line single-active multihoming mode	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 • 8011-12G12X4Y-D
EVPN E-Line single-active multihoming mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
EVPN E-Line single-active multihoming mode	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q200]) (select variants only*) *The EVPN E-Line single-active multi-homing mode is now extended to: • 8712-MOD-M • 8201-32FH • 8201-24H8FH • 8202-32FH-M • 8608 • 88-LC0-34H14FH • 88-LC0-36FH • 88-LC0-36FH-M
EVPN E-Line single-active multihoming mode	Release 24.3.1	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*) *The EVPN E-Line single-active multi-homing mode is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN E-Line single-active multihoming mode	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The single-active multi-homing mode offers redundant connectivity on a single link at a time with failover to the second link in case the active link fails. In this mode, only a single PE among a group of PEs attached to an Ethernet segment forwards traffic to and from that Ethernet Segment. *This feature is supported only on routers with the 88-LC1-36EH line cards.

Configure EVPN E-Line single-active multihoming mode

Configure an EVPN E-Line service in single-active multihoming mode to provide redundancy while allowing only one active link per Ethernet segment.

In single-active multihoming mode, only one PE device in a redundant pair actively forwards traffic for a given Ethernet segment, ensuring loop-free forwarding and simplified failover. This configuration is typically used in point-to-point (p2p) services where load sharing is not required.

Procedure

Step 1 Create a cross-connect group on all the PE devices.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xg1
```

Step 2 Configure point-to-point (p2p) cross-connect and assign an interface to the cross-connect on all the PE devices.

a) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE1.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether10.2
```

b) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE2.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether10.2
```

c) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE3.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether20.1
```

d) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE4.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether20.1
```

Step 3 Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on all PE devices.

a) Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on PE1.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)#  root
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
```

b) Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on PE2.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)#  root
Router(config)# evpn
```

```
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
```

c) Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on PE3.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)#  root
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
```

d) Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on PE4.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)#  root
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
```

Step 4 Enable single-active load balancing mode on all the PE devices.

Example:

```
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# commit
```

Step 5 Running configuration of EVPN E-Line single-active multihoming mode.

Example:

```
/* On PE1 */
!
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00
   load-balancing-mode single-active
!


/* On PE2 */
!
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00
   load-balancing-mode single-active
!
```

```
/* On PE3 */
!
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.14.00
   load-balancing-mode single-active
!

/* On PE4 */
!
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.14.00
   load-balancing-mode single-active
!
```

EVPN E-Line all-active multihoming mode

An EVPN E-Line all-active multihoming mode is a networking model that

- enables multiple simultaneous active connections from an EVPN to a single Ethernet LAN
- allows all PE routers attached to a particular Ethernet segment to forward traffic to and from that Ethernet segment, and
- provides redundancy and load balancing by allowing traffic distribution across all available links, enhancing network reliability and efficiency by utilizing all potential paths without requiring a failover mechanism.

The EVPN E-Line all-active multihoming mode uses the same conceptual framework as the EVPN E-Line all-active multihoming mode. For more details, refer to the EVPN E-LAN all-active multihoming mode, on page 59 section.

Table 13: Feature History Table

Feature Name	Release Information	Feature Description
E-Line all-active multihoming mode	Release 26.2.1	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
E-Line all-active multihoming mode	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
E-Line all-active multihoming mode	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 • 8011-12G12X4Y-D
E-Line all-active multihoming mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
E-Line all-active multihoming mode	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700) ; Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q200])(select variants only*) *The EVPN E-line all-active multi-homing mode is now extended to: • 8712-MOD-M • 8201-32FH • 8201-24H8FH • 8202-32FH-M • 8608 • 88-LC0-34H14FH • 88-LC0-36FH • 88-LC0-36FH-M

Feature Name	Release Information	Feature Description
E-Line all-active multihoming mode	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The E-Line all-active multi-homing mode is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
E-Line all-active multihoming mode	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) In all-active multihoming mode, multiple PE devices connected to the same CE are simultaneously active. Traffic is distributed across all active links, optimizing bandwidth usage and ensuring high availability. *This feature is supported only on routers with the 88-LC1-36EH line cards.

Configure EVPN E-Line all-active multihoming mode

Configure an EVPN E-Line service in all-active multihoming mode to enable load balancing and redundancy across multiple links.

In all-active multihoming mode, all PE devices in a redundant set actively forward traffic for the same Ethernet segment. This allows efficient bandwidth utilization and rapid failover while maintaining loop-free connectivity.

Before you begin

You must configure BGP when you set up EVPN with an EVI under a bridge-domain because BGP enables EVPN to work properly. Without BGP, the EVPN instance only functions locally and cannot connect to remote PEs.

Procedure

Step 1 Create a cross-connect group on all the PE devices.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xg1
```

Step 2 Configure point-to-point (p2p) cross-connect and assign an interface to the cross-connect on all the PE devices.

a) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE1.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether10.2
```

b) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE2.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether10.2
```

c) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE3.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether20.1
```

d) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE4.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether20.1
```

Step 3 Enable EVPN E-Line endpoint on all PE devices.

a) Enable EVPN E-Line endpoint on the p2p cross-connect for PE1.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 5 source 6
```

b) Enable EVPN E-Line endpoint on the p2p cross-connect for PE2.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 5 source 6
```

c) Enable EVPN E-Line endpoint on the p2p cross-connect for PE3.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 6 source 5
```

d) Enable EVPN E-Line endpoint on the p2p cross-connect for PE4.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 6 source 5
```

Step 4 Configure the ESI on all PE devices.

a) Configure the ESI on PE1.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
```

b) Configure the ESI on PE2.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
```

c) Configure the ESI on PE3.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
```

d) Configure the ESI on PE4.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
```

Step 5 Running configuration of EVPN E-Line all-active multihoming mode.

Example:

```
/* On PE1 */
!
configure
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00

!


/* On PE2 */
!
configure
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00

!

/* On PE3 */
!
configure
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether20.1
```

```
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
      identifier type 0 00.01.00.ac.ce.55.00.14.00

!

/* On PE4 */
!
configure
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
      identifier type 0 00.01.00.ac.ce.55.00.14.00

!
```

EVPN E-Line port-active multihoming mode

An EVPN E-Line port-active multihoming is a network model that

- supports single-active redundancy load balancing at the port-level or interface-level
- provides faster convergence during a link failure, and
- enables protocol simplification by having only one physical port active at a given time.

The EVPN E-Line port-active multihoming mode uses the same conceptual framework as the EVPN E-LAN port-active multihoming mode. For more details, refer to the EVPN E-LAN port-active multihoming mode, on page 66 section.

Table 14: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-Line port-active multihoming mode	Release 26.2.1	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 E-Line port-active multihoming mode	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.

Feature Name	Release Information	Feature Description
EVPN E-Line port-active multihoming mode	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 • 8011-12G12X4Y-D
EVPN E-Line port-active multihoming mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN E-Line port-active multihoming mode	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*) *The EVPN E-Line port-active multi-homing mode is now extended to: • 8712-MOD-M • 8201-32FH • 8201-24H8FH • 8202-32FH-M • 8608 • 88-LC0-34H14FH • 88-LC0-36FH • 88-LC0-36FH-M
EVPN E-Line port-active multihoming mode	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The EVPN E-Line port-active multi-homing mode is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
EVPN E-Line port-active multihoming mode	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The port-active multi-homing mode enables single-active redundancy load balancing at the port-level or the interface-level. In this mode, one of the PEs remains active at the port-level. This feature enables protocol simplification as only one of the physical ports is active at a given time. *This feature is supported only on routers with the 88-LC1-36EH line cards.

Configure EVPN E-Line port-active multihoming mode

Configure an EVPN E-Line service in port-active multihoming mode to provide redundancy while ensuring only one active link per port for a given Ethernet segment.

In port-active multihoming mode, only one port in an Ethernet segment actively forwards traffic, while others remain in standby mode. This mode is typically used in access networks where per-port active-standby control is required for operational or design reasons.

Procedure

Step 1 Create a cross-connect group on all the PE devices.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xg1
```

Step 2 Configure point-to-point (p2p) cross-connect and assign an interface to the cross-connect on all the PE devices.

a) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE1.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether10.2
```

b) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE2.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether10.2
```

c) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE3.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether20.1
```

d) Point-to-point (p2p) cross-connect and assign an interface to the cross-connect on PE4.

Example:

```
Router(config-l2vpn-xc)# p2p e1_5-6
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether20.1
```

Step 3 Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on all PE devices.

a) Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on PE1.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)#  root
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
```

b) Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on PE2.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)#  root
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
```

c) Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on PE3.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)#  root
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
```

d) Enable EVPN E-Line endpoint on the p2p cross-connect and configure the ESI on PE4.

Example:

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)#  root
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
```

Step 4 Enable port-active load balancing mode on all the PE devices.

Example:

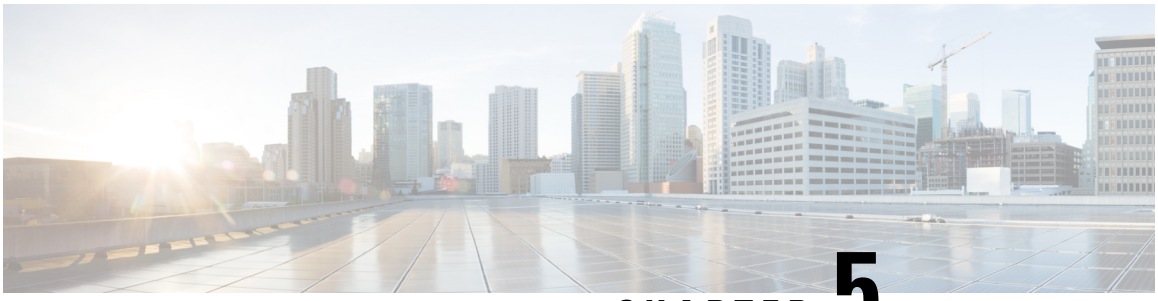
```
Router(config-evpn-ac-es)# load-balancing-mode port-active
Router(config-evpn-ac-es)# commit
```

Step 5 Running configuration of EVPN E-Line port-active multihoming mode.

Example:

```
/* On PE1 */
!
```

```
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00
   load-balancing-mode port-active
!


/* On PE2 */
!
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00
   load-balancing-mode port-active
!

/* On PE3 */
!
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.14.00
   load-balancing-mode port-active
!

/* On PE4 */
!
l2vpn xconnect group xg1
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.14.00
   load-balancing-mode port-active
!
```



CHAPTER 5

Migration of VPLS and VPWS Networks to EVPN

Seamless migration of VPLS network to an EVPN network

A migration from VPLS to EVPN is a process that

- enables service providers to gradually upgrade their VPLS networks to EVPN without disrupting services
- facilitates the coexistence of legacy VPLS and an EVPN-VPLS dual-stack configurations, and
- leverages MP-BGP for efficient MAC learning and propagation.

Table 15: Feature History Table

Feature Name	Release Information	Feature Description
Seamless Migration of VPLS Network to EVPN Network	Release 26.2.1	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.
Seamless Migration of VPLS Network to EVPN Network	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Feature Name	Release Information	Feature Description
Seamless Migration of VPLS Network to EVPN Network	Release 25.3.1	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 configure local static MPLS labels for unicast IP traffic under the EVPN EVI configuration, which ensures remote PEs use a consistent, common label for the same EVPN service, improving forwarding consistency and operational control. The feature introduces these changes: **CLI**: • **mpls static label (EVPN ELAN)**
Seamless Migration of VPLS Network to EVPN Network	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
Seamless Migration of VPLS Network to EVPN Network	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The seamless VPLS-to-EVPN migration is now extended to the Cisco 8712-MOD-M routers.
Seamless Migration of VPLS Network to EVPN Network	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The seamless VPLS-to-EVPN migration is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Seamless Migration of VPLS Network to EVPN Network	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The seamless VPLS-to-EVPN migration is now extended to routers with the 88-LC1-36EH line cards.
Seamless Migration of VPLS Network to EVPN Network	Release 7.11.1	You can now provision an EVPN service on existing VPLS-enabled PEs individually, thus ensuring a seamless VPLS-to-EVPN migration without traffic disruption. This feature is supported only on Q200-based line cards.

Transitioning from VPLS to EVPN

Although VPLS is a widely deployed Layer 2 VPN technology, customers are increasingly migrating their VPLS networks to EVPN to benefit from improved scalability and simplified deployment. Recognizing the importance of preserving existing investments in VPLS, service providers are exploring ways to seamlessly integrate their legacy VPLS networks with new EVPN-based networks.

Key benefits of migration

The migration of VPLS to an EVPN network offers these benefits:

- Incremental migration: Service providers can migrate PE nodes from VPLS to EVPN gradually, ensuring no service disruption.
- Dual-Stack coexistence: Legacy VPLS and EVPN-VPLS can coexist in the same MPLS network, enabling a smooth transition.
- Control plane efficiency: EVPN uses MP-BGP for MAC learning and propagation, unlike VPLS, which relies on the data plane.

Technical highlights

These points highlight the key aspects of VPLS to an EVPN migration:

- EVPN instance grouping: VPN instances in EVPN are grouped by EVPN Instance ID (EVI-ID) and associated with route targets and route distinguishers.
- MAC learning: EVPN employs a control plane for MAC learning, while VPLS uses a flood-and-learn technique in the data plane.
- Route Types:
  - Type-2: Advertises customer MAC addresses.
  - Type-3: Handles broadcast, unknown unicast, and multicast (BUM) traffic using ingress replication multicast routes.

Migration process

These stages describe the migration process for transitioning from VPLS to EVPN.

- Gradual PE node upgrade: Upgrade one PE node at a time without requiring a network-wide software update.
- Route exchange: An EVPN-enabled PEs advertise both BGP VPLS autodiscovery (AD) routes and EVPN multicast routes (Type-3) for seamless integration.
- BUM traffic handling: Type-3 routes ensure that PEs with matching route targets receive BUM traffic.

Table 16: Comparison of VPLS and EVPN

Feature	VPLS	EVPN
MAC learning	Data plane	Control lane
Protocol	Flood and learn	MP-BGP

Feature	VPLS	EVPN
Route Types	Not applicable	Type-2 (MAC), Type-3 (BUM)

MPLS static label support for EVPN ELAN

From Release 25.3.1, you can assign a static MPLS label to an EVPN service. This static assignment overrides the default dynamic label allocation by Cisco IOS XR software.

When you configure a static MPLS label on Provider Edge (PE) routers, this ensures that remote PEs receive traffic for the same service with a consistent, common label.

You can configure static MPLS labels for unicast IP traffic in EVPN ELAN by assigning local static labels under the EVPN EVI configuration mode. The range for these static MPLS labels is from 16 to 1,048,575.

Configure static MPLS labels within the range of 16 to 15,000 to avoid conflicts with existing dynamic labels and the default Segment Routing Local Block (SRLB) range of 15,000 to 15,999. For more information, see About the Segment Routing Local Block in the *Segment Routing Configuration Guide for Cisco 8000 Series Routers*.

Migrating VPLS network to an EVPN network

Summary

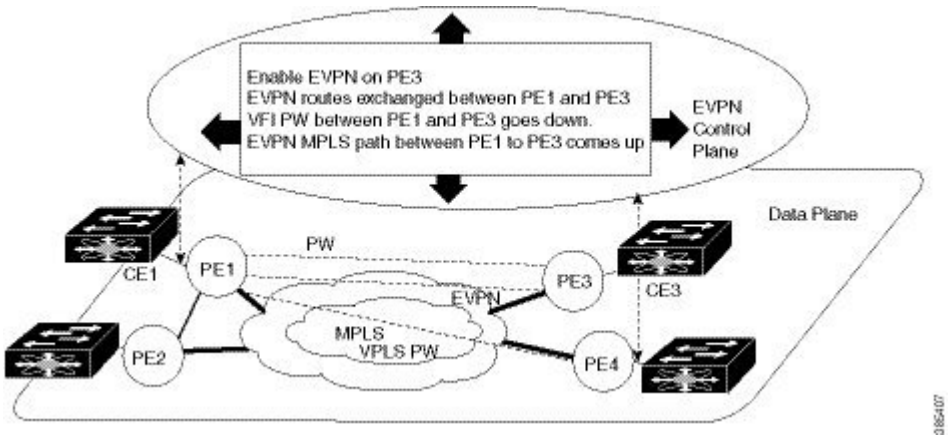
The migration process involves transitioning a VPLS network to an EVPN network. This process ensures seamless integration and avoids traffic disruption.

The key components involved in the process are:

- PE nodes: Devices such as PE1, PE2, PE3, and PE4 that form the network.
- VPLS Pseudowires (PW): Connections between PE nodes in the VPLS network.
- EVPN service: A service that replaces VPLS for enhanced network functionality.

Workflow

Figure 6: Seamless migration of VPLS network to EVPN network



These are the stages of migration process:

1. Initial setup: Ensure that PE1, PE2, PE3, and PE4 are interconnected in a full-meshed topology using VPLS pseudowires.
2. Enable EVPN on PE1:
   - Activate EVPN in a VPN instance of the VPLS service on PE1.
   - PE1 starts advertising the EVPN inclusive multicast route to other PE nodes.
   - Since no inclusive multicast routes are received from other PE nodes, VPLS pseudowires between PE1 and other PE nodes remain active.
   - PE1 forwards traffic using VPLS pseudowires and advertises all MAC addresses learned from CE1 using EVPN Route Type-2.
3. Enable EVPN on PE3:
   - Activate EVPN on PE3.
   - PE3 starts advertising an inclusive multicast route to other PE nodes.
   - PE1 and PE3 discover each other through EVPN routes and shut down pseudowires between them.
   - EVPN service replaces VPLS service between PE1 and PE3.
4. Seamless integration: PE1 continues running VPLS service with PE2 and PE4 while starting EVPN service with PE3 in the same VPN instance.
5. Migrate the remaining nodes: Repeat the process for PE2 and PE4 until all PE nodes are enabled with the EVPN service.
6. Complete the migration: After all nodes are migrated, the VPLS service is completely replaced with the EVPN service, and all VPLS pseudowires are shut down.

   The migration process ensures a seamless transition from VPLS to EVPN, enhancing network efficiency and functionality without disrupting traffic.

Configure EVPN on the existing VPLS network

Enable EVPN on an existing VPLS network to enhance Layer 2 VPN capabilities.

This task involves configuring EVPN on an existing VPLS network by setting up the L2VPN EVPN address-family, EVI, and corresponding BGP route-targets, and verifying the configuration.

Before you begin

- Ensure that the PE routers are operational and configured for VPLS.
- Verify that the required BGP configurations are in place.

Procedure

Step 1 Configure L2VPN EVPN address-family.

Example:

```
Router# configure
Router(config)#router bgp 65530
Router(config-bgp)#nsr
Router(config-bgp)#bgp graceful-restart
Router(config-bgp)#bgp router-id 200.0.1.1
Router(config-bgp)#address-family l2vpn evpn
Router(config-bgp-af)#exit
Router(config-bgp)#neighbor 200.0.4.1
Router(config-bgp-nbr)#remote-as 65530
Router(config-bgp-nbr)#update-source Loopback0
Router(config-bgp-nbr)#address-family l2vpn evpn
Router(config-bgp-nbr-af)#commit
```

Step 2 Running configuration of L2VPN EVPN address-family.

Example:

```
configure
 router bgp 65530
  nsr
  bgp graceful-restart
  bgp router-id 200.0.1.1
  address-family l2vpn evpn
  !
  neighbor 200.0.4.1
   remote-as 65530
   update-source Loopback0
   address-family l2vpn evpn
   !
 !
```

Step 3 Use the **show bgp l2vpn evpn summary** command to verify that the BGP neighbor is functional.

Example:

```
Router# show bgp l2vpn evpn summary
BGP router identifier 200.0.1.1, local AS number 65530
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 1
BGP NSR Initial initsync version 4294967295 (Not Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.

Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker               1          1          1          0           1           0

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
200.0.4.1         0      65530       2       2        0    0    0 00:00:09          0
```

Step 4 Configure EVI under EVPN configuration mode.

Example:

To enable EVPN on PE1, configure EVI. Also, configure advertise-mac, else the MAC routes (Type-2) are not advertised.

```
Router# configure
```

```
Router(config)#evpn
Router(config-evpn)#evi 1
Router(config-evpn-evi)#advertise-mac
Router(config-evpn-evi)#commit
```

Step 5 EVI running configuration.

Example:

```
configure
 evpn
  evi
   advertise-mac
   !
  !
 !
```

Step 6 Use the **show evpn summary** command to verify the number of configured EVIs and the advertised local and remote MAC routes.

Example:

```
Router#show evpn summary
-----------------------------
Global Information
-----------------------------
Number of EVIs                  : 6
Number of Local EAD Entries     : 0
Number of Remote EAD Entries    : 0
Number of Local MAC Routes      : 4
          MAC                   : 4
          MAC-IPv4              : 0
          MAC-IPv6              : 0
Number of Local ES:Global MAC   : 1
Number of Remote MAC Routes     : 0
          MAC                   : 0
          MAC-IPv4              : 0
          MAC-IPv6              : 0
Number of Remote SOO MAC Routes    :0
Number of Local IMCAST Routes   : 4
Number of Remote IMCAST Routes  : 4
Number of Internal Labels       : 0
Number of ES Entries            : 1
Number of Neighbor Entries      : 4
EVPN Router ID                  : 200.0.1.1
BGP ASN                         : 65530
PBB BSA MAC address             : 0026.982b.c1e5
Global peering timer            :      3 seconds
Global recovery timer           :     30 seconds
```

```
Router#show evpn evi vpn-id 1 mac
Mon Feb 20 21:36:23.574 EST

EVI        MAC address    IP address                    Nexthop                          Label
---------- -------------- ------------------------------------- ---------------------------------
1      0033.0000.0001      ::                        200.0.1.1                        45106
```

Step 7 Configure EVI under the corresponding L2VPN 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)#exit
Router(config-l2vpn-bg-bd)#evi 1
Router(config-l2vpn-bg-bd-evi)#exit
Router(config-l2vpn-bg-bd)#vfi v1
Router(config-l2vpn-bg-bd-vfi)#neighbor 172.16.0.1 pw-id 12
Router(config-l2vpn-bg-bd-vfi-pw)#neighbor 192.168.0.1 pw-id 13
Router(config-l2vpn-bg-bd-vfi-pw)#mpls static label local 20001 remote 10001
Router(config-l2vpn-bg-bd-vfi-pw)#commit
```

Step 8 EVI runniing configuration under the corresponding L2VPN bridge domain.

Example:

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface HundredGigE0/0/0/0
    !
    evi 1
    !
   vfi v1
    neighbor 172.16.0.1 pw-id 12
    neighbor 192.168.0.1 pw-id 13
     mpls static label local 20001 remote 10001
     !
    !
```

Step 9 Use the **show l2vpn bridge-domain** command to verify EVPN and VPLS status.

Example:

```
Router# show l2vpn bridge-domain
Legend: pp = Partially Programmed.
Bridge group: vplstoevpn, bridge-domain: vplstoevpn, id: 0, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 1 (1 up), VFIs: 1, PWs: 2 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
  List of ACs:
    Hu0/0/0/0, state: up, Static MAC addresses: 0, MSTi: 5
  List of Access PWs:
  List of VFIs:
    VFI vpls (up)
      Neighbor 172.16.0.1 pw-id 12, state: down, Static MAC addresses: 0
      Neighbor 192.168.0.1 pw-id 13, state: up, Static MAC addresses: 0
```

The output indicates that the VPLS PW "neighbor 172.16.0.1 pw-id 12" is replaced by EVPN service, as the EVPN control plane discovered that both local PE and remote PE (172.16.0.1) have enabled EVPN service on the L2VPN instance.

Seamless migration of VPWS network to EVPN network

A migration from VPWS to EVPN is a process that

- enables gradual and incremental migration of PE nodes from legacy VPWS to EVPN-VPWS without service disruption
- allows migration of Attachment Circuits (ACs) connected to legacy VPWS pseudowires (PWs) using targeted-LDP or BGP-AD signaling to EVPN-VPWS services, and
- groups VPN instances by EVPN Instance VPN ID (EVI), identified by an Ethernet tag or AC-ID, and associates them with route-targets and route-distinguisher.

Table 17: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Seamless Integration with Legacy VPWS	Release 26.2.1	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 Seamless Integration with Legacy VPWS	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN Seamless Integration with Legacy VPWS	Release 25.3.1	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 configure local static MPLS labels for EVPN VPWS under the L2VPN cross-connect P2P EVPN EVI configuration, which ensures remote PEs use a consistent, common label for the same EVPN service, improving forwarding consistency and operational control. The feature introduces these changes: **CLI**: • **mpls static label (EVPN VPWS)**
EVPN Seamless Integration with Legacy VPWS	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
EVPN Seamless Integration with Legacy VPWS	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The seamless migration of VPWS to EVPN-VPWS services functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN Seamless Integration with Legacy VPWS	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The seamless migration of VPWS to EVPN-VPWS services 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
EVPN Seamless Integration with Legacy VPWS	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: Q200, P100])c(select variants only*) *The seamless migration of VPWS to EVPN-VPWS services functionality is now extended to routers with the Q200 and 88-LC1-36EH line cards.
EVPN Seamless Integration with Legacy VPWS	Release 7.8.1	When expanding an existing L2VPN network, users may want to deploy EVPN-VPWS to provide additional Layer 2 point-to-point Ethernet services, and at the same time some of their customer traffic may still need to be terminated on the existing L2VPN PEs on their network. Users can migrate the PE nodes from L2VPN VPWS to EVPN-VPWS, without disruption in traffic. The seamless migration offers users the option to use either VPWS or EVPN-VPWS services on PE nodes. This allows the coexistence of legacy VPWS and EVPN-VPWS dual-stack in the core for a given L2 Attachment Circuit (AC) over the same MPLS network. This feature introduces the **vpws-seamless-integration** command.

Transitioning from VPWS to EVPN

Although VPWS is a widely deployed Layer 2 VPN technology, customers are increasingly transitioning their VPWS networks to EVPN to take advantage of enhanced scalability, operational flexibility, and simplified deployment. To protect existing investments in VPWS infrastructure, service providers are focusing on strategies to ensure smooth integration between their legacy VPWS networks and modern EVPN-based networks.

Key highlights

These are the key highlights of migrating VPWS to EVPN:

- Migration flexibility: Users can migrate PE nodes from VPWS to EVPN incrementally.

- Attachment Circuit (AC) migration: Supports the migration of ACs connected to legacy VPWS pseudowires (PWs) using either targeted-LDP signaling or BGP Auto-Discovery (BGP-AD) signaling to EVPN-VPWS.
- VPN instance grouping: In EVPN-VPWS, VPN instances are grouped by EVPN Instance VPN ID (EVI) and identified using an Ethernet tag or Attachment Circuit ID (AC-ID). The EVI is associated with route targets and route distinguishers.
- Cross-connect functionality: During migration, the EVPN-VPWS PE router can perform either VPWS or EVPN-VPWS Layer 2 cross-connect for a given AC.
- Route advertisement: When both EVPN-VPWS and BGP-AD PWs are configured for the same AC, the EVPN-VPWS PE advertises both the BGP VPWS Auto-Discovery (AD) route and the BGP EVPN Auto-Discovery (EVI/EAD) route, prioritizing EVPN-VPWS pseudowires over BGP-AD VPWS pseudowires.

MPLS static label support for EVPN VPWS

From Release 25.3.1, you can assign a static MPLS label to an EVPN service. This static assignment overrides the default dynamic label allocation by Cisco IOS XR software.

When you configure a static MPLS label on Provider Edge (PE) routers, this ensures that remote PEs receive traffic for the same service with a consistent, common label.

The MPLS static label support for EVPN VPWS provides the same static label functionality as legacy P2P services.

You can configure local static MPLS labels for EVPN VPWS under the L2VPN cross-connect P2P EVPN EVI configuration mode. The range for these static MPLS labels is from 16 to 1,048,575.

Configure static MPLS labels within the range of 16 to 15,000 to avoid conflicts with existing dynamic labels and the default Segment Routing Local Block (SRLB) range of 15,000 to 15,999. For more information, see About the Segment Routing Local Block in the *Segment Routing Configuration Guide for Cisco 8000 Series Routers*.

Migrating VPWS network to EVPN network

In the current topology, PE1, PE2, and PE3 are provider edge devices connected through legacy pseudowires. The user plans to replace PE1 with new hardware and enable EVPN on it. This phased migration ensures minimal disruption to the network.

Summary

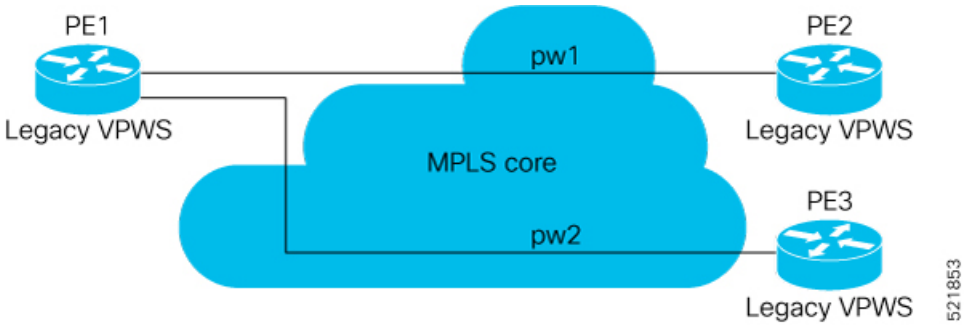
This process outlines the steps to migrate a legacy VPWS network to EVPN in a phased manner. The migration is designed to be seamless and can span multiple years.

The key components involved in the process are:

- Provider edge eevices (PE1, PE2, PE3): These are the MPLS network devices where the legacy VPWS cross-connects are operational.
- Legacy pseudowires (PW1, PW2): These are the existing connections between the provider edge devices.
- New hardware for PE1: This replaces the legacy VPWS on PE1 and enables EVPN functionality.

Workflow

Figure 7: VPWS nodes



These are the stages of migration process:

1. Preparation:
   - Identify the legacy VPWS nodes to be migrated.
   - Plan the migration timeline, considering that it may span multiple years.
2. Replacement of PE1:
   - Replace the existing PE1 hardware with new equipment.
   - Ensure that the new hardware is compatible with EVPN-VPWS.
3. Enabling EVPN-VPWS on PE1:
   - Configure the new PE1 to support EVPN-VPWS.
   - Verify the configuration to ensure seamless integration with the existing network.
4. Testing and validation:
   - Test the connectivity between PE1 and other provider edge devices (PE2 and PE3).
   - Validate the functionality of EVPN-VPWS on the new PE1.
5. Phased migration of other nodes:
   - Gradually replace other legacy VPWS nodes (PE2, PE3) with EVPN-VPWS-enabled hardware.
   - Repeat the configuration, testing, and validation steps for each node.

   The migration process enables the network to transition from a legacy VPWS setup to an EVPN-VPWS architecture, ensuring improved scalability, flexibility, and operational efficiency.

How EVPN-VPWS migration works

Summary

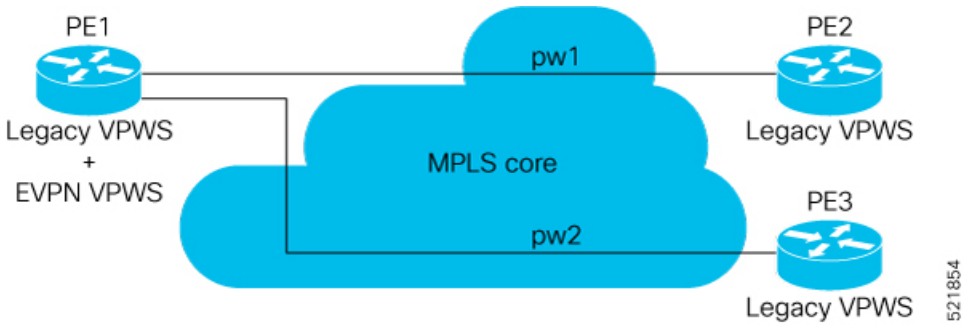
The process of migrating to EVPN-VPWS involves enabling EVPN-VPWS on PE1, followed by the gradual upgrade of other PE nodes.

The key components involved in this process are:

- PE1: Initiates EVPN-VPWS and advertises routes.
- PE2 and PE3: Operate in legacy VPWS mode until upgraded.
- BGP VPWS Auto-Discovery route: Facilitates route advertisement.

Workflow

Figure 8: PE1 enabled with EVPN-VPWS



The process involves these stages:

1. Enabling EVPN-VPWS on PE1:
   - PE1 starts advertising EVPN EVI or Ethernet-AD routes to other PE nodes.
   - PE1 advertises the BGP VPWS Auto-Discovery route and the BGP EVPN Ethernet-AD per EVI route for a given pseudowire (PW).
2. Interoperability with legacy VPWS:
   - Since PE2 and PE3 are not yet migrated, PE1 does not receive any EVI/EAD routes from these nodes.
   - Legacy VPWS continues to operate between PE1, PE2, and PE3.
   - PE1 forwards traffic using the legacy VPWS mechanism.
3. Upgrading PE2 to EVPN-VPWS:
   - After one year, PE2 is upgraded to EVPN-VPWS.
   - PE2 begins advertising EVPN EVI or Ethernet-AD routes, enabling EVPN-VPWS communication with PE1.

   The migration ensures a seamless transition from legacy VPWS to EVPN-VPWS, maintaining traffic forwarding during the process and enabling advanced EVPN-VPWS features post-upgrade.

How EVPN-VPWS integration works

Summary

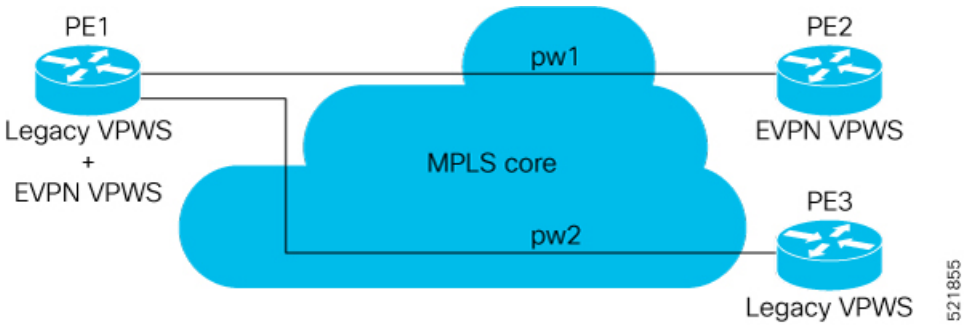
The process of integrating EVPN-VPWS with legacy VPWS involves multiple stages, ensuring seamless migration while maintaining network stability.

The key components involved in the process are:

- PE1 and PE2: These devices facilitate the transition from legacy VPWS to EVPN-VPWS.
- PE3: A device that remains on legacy VPWS during the initial migration stages.
- EVPN-VPWS service: A high-priority service that replaces legacy VPWS.

Workflow

Figure 9: PE2 enabled with EVPN-VPWS



The process involves these stages:

1. Upgrade completion: After the upgrade is completed, PE2 begins advertising EVI/EAD routes to other PE nodes.
2. Discovery and integration:
   - PE1 and PE2 discover each other through EVPN routes.
   - EVPN-VPWS service replaces the legacy VPWS service between PE1 and PE2, achieving seamless integration.
3. Service prioritization: EVPN-VPWS service takes precedence over the legacy VPWS network.
4. Legacy VPWS shutdown: PE1 and PE2 shut down the legacy VPWS between them to prevent duplicate packets from remote CE devices.
5. Coexistence with PE3: PE3 remains on legacy VPWS, and PE1 continues running legacy VPWS service with PE3.
6. Dual-Stack coexistence: The migration continues for remaining PE nodes, with legacy VPWS and EVPN-VPWS coexisting in the core for a given L2 AC.
7. Future upgrades:
   - After a year, PE3 is upgraded to enable EVPN-VPWS service.

- All PE devices are eventually replaced with EVPN-VPWS services.

8. Configuration retention: The user retains both legacy and EVPN-VPWS-related configurations on PE1 and PE2 nodes.
9. Rollback capability: If network issues arise, the migration can be rolled back. This reverts PE2, and subsequently PE1 and PE2, to the legacy VPWS configuration.

   The migration ensures a seamless transition to EVPN-VPWS while maintaining network stability and providing rollback options for troubleshooting.

Configure EVPN on existing VPWS

Enable seamless migration and intergration of legacy VPWS to EVPN to ensure smooth migration and coexistence.

This task involves configuring EVPN-VPWS alongside legacy VPWS and migrating PE devices incrementally.

Before you begin

- Ensure that the network devices support EVPN-VPWS.
- Verify that the required BGP configurations are in place.

Procedure

Step 1 Migrate VPWS to EVPN-VPWS on PE1.

Example:

In this example, both legacy VPWS and EVPN-VPWS coexist on PE1.

```
/* VPWS configuration on PE1 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
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 Bundle-Ether1.1 remote-ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# exit
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2 remote-ce-id 4
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit

/* Migrate VPWS to EVPN-VPWS on PE1 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
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)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1 remote-ce-id 2
```

```
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# root
Router(config)# l2vpn xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 5
Router(config-l2vpn-xc-p2p-pw)# commit

/* VPWS configuration on PE2 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
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 Bundle-Ether1.1 remote-ce-id 1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit

/* VPWS configuration on PE3 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 4
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2 remote-ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

Step 2 Use the **show l2vpn xconnect** command to check the VPWS status.

As PE2 and PE3 are not migrated to EVPN-VPWS, legacy VPWS continues to run between the PE devices.

Example:

This show output indicates that only legacy VPWS is up and EVPN-VPWS is down on BE1.1.

```
Router# show l2vpn xconnect

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
-------------------------  -----------------------------   -----------------------------
evpn-vpws    evpn1      DN   BE1.1                  UP       EVPN 4,5,24004          DN
-------------------------------------------------------------------------------------
legacy-vpws  vpws1      UP   BE1.1                  UP       192.168.0.4     534296  UP
-------------------------------------------------------------------------------------
legacy-vpws  vpws1      UP   BE1.2                  UP       192.168.12.110  685694  UP
-------------------------------------------------------------------------------------
```

Step 3 Migrate VPWS to EVPN-VPWS on PE1 and PE2.

Example:

In this example, both legacy VPWS and EVPN-VPWS coexist on PE1. PE2 is migrated to EVPN-VPWS.

```
/* VPWS configuration on PE1 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
```

```
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 Bundle-Ether1.1 remote-ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# exit
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2 remote-ce-id 4
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit

/* Migrate VPWS to EVPN-VPWS on PE1 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
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)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1 remote-ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# root

Router(config)# l2vpn xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 5
Router(config-l2vpn-xc-p2p-pw)# commit


/* Migrate VPWS to EVPN-VPWS on PE2 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
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)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1 remote-ce-id 1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# root

Router(config)# l2vpn xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 5
Router(config-l2vpn-xc-p2p-pw)# commit
```

Step 4 Use the **show l2vpn xconnect** to check the VPWS status.

After the migration, legacy VPWS and EVPN-VPWS coexist on PE1. PE2 is migrated to EVPN-VPWS and PE3 runs with legacy VPWS. EVPN-VPWS service runs between PE1 and PE2. Legacy VPWS service runs between PE1 and PE3.

Example:

This example shows that EVPN-VPWS is up on BE1.1. The legacy VPWS is also advertised on BE1.1 with the status as Standby (**SB(SI)**).

```
Router# show l2vpn xconnect

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
-------------------------  ------------------------------   -----------------------------
evpn-vpws    evpn1      UP   BE1.1                  UP       EVPN 4,5,24004           UP
-------------------------------------------------------------------------------------------
legacy-vpws  vpws1      DN   BE1.1                  SB(SI)   192.168.0.4     534296   UP
-------------------------------------------------------------------------------------------
legacy-vpws  vpws1      UP   BE1.2                  UP       192.168.12.110  685694   UP
-------------------------------------------------------------------------------------------
```

Use the **show l2vpn forwarding interface** *interface-type interface-path-id* **detail location** *node-id* command to identify whether EVPN-VPWS or VPWS is used for forwarding the traffic.

In this example, **evi: 1** indicates that EVPN-VPWS is used for forwarding the traffic.

```
Router# show l2vpn forwarding interface Bundle-Ether1.1 detail location 0/2/CPU0
Wed Apr 28 09:08:37.512 EDT
Local interface: Bundle-Ether1.1, Xconnect id: 0x800001, Status: up
  Segment 1
    AC, Bundle-Ether1.1, status: Bound
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
  Segment 2
    MPLS, Destination address: 192.168.0.4, evi: 4, ac-id: 5, status: Bound
Pseudowire label: 24001
    Control word enabled
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
```

In this example, **pw-id: 1** indicates that VPWS is used for forwarding the traffic.

```
Router# show l2vpn forwarding interface Bundle-Ether1.1  detail location 0/2/CPU0
Wed Apr 28 09:09:45.204 EDT
Local interface: Bundle-Ether1.1, Xconnect id: 0x800001, Status: up
  Segment 1
    AC, Bundle-Ether1.1, status: Bound
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
  Segment 2
    MPLS, Destination address: 192.168.0.4, pw-id: 1, status: Bound
Pseudowire label: 24000
    Control word disabled
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
```

Use the **l2vpn logging pseudowire** command to track the migration of AC from one PW to another.

```
Router(config)# l2vpn logging pseudowire
RP/0/0/CPU0:Jan 18 15:35:15.607 EST:
l2vpn_mgr[1234]: %L2-EVPN-5-VPWS_SEAMLESS_INTEGRATION_STATE_CHANGE :
GigabitEthernet0/2/0/8.1 - Active XC is now service-1:evpn-vpws-1, standby XC is service-1:legacy-vpws-1
```

Step 5 Migrate VPWS to EVPN-VPWS on PE1, PE2, and PE3.

Example:

In this example, both legacy VPWS and EVPN-VPWS coexist on PE1. PE2 and PE3 are migrated to EVPN-VPWS.

```
/* VPWS configuration on PE1 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
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 Bundle-Ether1.1 remote-ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# exit
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2 remote-ce-id 4
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit

/* Migrate VPWS to EVPN-VPWS on PE1 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
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)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.1 remote-ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# exit
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2 remote-ce-id 4
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# root

Router(config)# l2vpn xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 5
Router(config-l2vpn-xc-p2p-pw)# commit
Router(config-l2vpn-xc-p2p-pw)# root
Router(config)# l2vpn xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn2
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 7
Router(config-l2vpn-xc-p2p-pw)# commit

/* Migrate VPWS to EVPN-VPWS on PE3 */
Router# configure
Router(config)# l2vpn xconnect group legacy-vpws
Router(config-l2vpn-xc)# mp2mp vpws1
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 4
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# vpws-seamless-integration
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface Bundle-Ether1.2 remote-ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# root
Router(config)# l2vpn xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn2
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether 1.2
```

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 service 7
Router(config-l2vpn-xc-p2p-pw)# commit
```

Step 6 Use the **l2vpn logging pseudowire** command to verify that all the PE devices forward traffic between them using EVPN-VPWS.

This example shows that EVPN-VPWS is up and legacy VPWS is down.

Example:

```
Router# show l2vpn xconnect

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
-------------------------  -----------------------------   -----------------------------
evpn-vpws    evpn1      UP   BE1.1                  UP       EVPN 4,5,24004          UP
-------------------------------------------------------------------------------------
legacy-vpws  vpws1      DN   BE1.1                  UP       192.168.0.4     534296  DN
-------------------------------------------------------------------------------------
evpn-vpws    evpn2      UP   BE1.2                  UP       EVPN 4,7,24008          UP
-------------------------------------------------------------------------------------
legacy-vpws  vpws1      DN   BE1.2                  UP       192.168.12.110  685694  DN
-------------------------------------------------------------------------------------
```

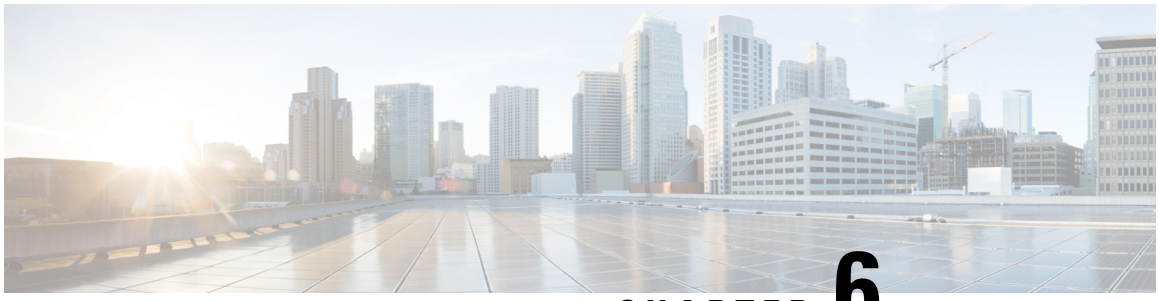
Step 7 TLDP PW to EVPN-VPWS migration

Similar to migrating VPWS to EVPN, you can also migrate Targeted Label Distribution Protocol (TLDP) PW to EVPN-VPWS on all the PE routers incrementally.

You can perform this task on all the PE routers incrementaly. This configuration example shows the TLDP PW to EVPN-VPWS migration on PE1:

Example:

```
Router# configure
Router(config)# l2vpn xconnect group 1
Router(config-l2vpn-xc)# p2p p1
Router(config-l2vpn-xc-p2p)# interface BE1.1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# vpws-seamless-integration
```



CHAPTER 6

Fundamentals and Core Features of EVPN

This chapter summarizes essential EVPN E-LAN features, including BUM ingress replication, split-horizon groups, core isolation, and cost-out. Users can learn how to efficiently manage BUM traffic, prevent loops, ensure core stability, and perform seamless maintenance in EVPN networks.

BUM ingress replication for EVPN E-LAN

EVPN BUM ingress replication is a network process that

- replicates broadcast, unknown unicast, and multicast (BUM) traffic at the ingress provider edge (PE) device
- uses ingress replication to deliver BUM traffic to all remote PE devices within the EVPN E-LAN instance, and
- eliminates the need for multicast routing protocols or multicast trees in the provider network.

Table 18: Feature History Table

Feature Name	Release Information	Feature Description
BUM Ingress Replication for EVPN Single-Homing	Release 26.2.1	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.
BUM Ingress Replication for EVPN Single-Homing	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.

Feature Name	Release Information	Feature Description
BUM Ingress Replication for EVPN Single-Homing	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
BUM Ingress Replication for EVPN Single-Homing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
BUM Ingress Replication for EVPN Single-Homing	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) *This feature is now supported on the Cisco 8712-MOD-M routers.
BUM Ingress Replication for EVPN Single-Homing	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 BUM ingress replication functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
BUM Ingress Replication for EVPN Single-Homing	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The BUM ingress replication functionality is now extended to routers with the 88-LC1-36EH line cards.
BUM Ingress Replication for EVPN Single-Homing	Release 7.11.1	You can optimize Broadcast, Unknown Unicast, and Multicast (BUM) traffic by ensuring that traffic that a device receives is replicated and forwarded to only those CE devices in an EVPN network, if and when they require it. This reduction in the unnecessary forwarding of BUM traffic prevents the flooding of BUM traffic to all devices on the EVPN network. This feature is supported only on Q200-based line cards. This feature is enabled by default.

Feature highlights of BUM ingress replication for EVPN E-LAN

EVPN BUM ingress replication enhances network efficiency and security by:

- optimizing resource allocation through forwarding BUM traffic only to necessary EVPN instances or VRFs, thereby minimizing traffic flooding,
- reducing congestion and preserving bandwidth within the broadcast domain,
- lowering overhead and mitigating potential performance issues,
- ensuring effective BUM traffic management to support network scalability without overwhelming broadcast or multicast traffic,
- improving network security by selectively duplicating BUM traffic to designated recipients, limiting exposure to unintended devices and reducing unauthorized access risks, and
- preventing broadcast storms and network disruptions by restricting traffic forwarding to intended devices only.

How BUM ingress replication works

Summary

The key components involved in the EVPN BUM traffic process are:

- Ingress router: learns MAC addresses related to BUM traffic and collects multicast group membership information.
- BGP: signals and distributes MAC and multicast information to other routers within the EVPN network.

Workflow

These stages describe how BUM ingress replication works.

1. The ingress router learns MAC addresses associated with BUM traffic and gathers multicast group membership details.
2. Instead of flooding BUM traffic to all ports in the EVPN, the ingress router replicates the traffic only to the specific EVPN instances or VRFs where it is required.
3. BGP distributes the MAC and multicast information to other routers in the EVPN network.
4. The ingress router replicates and forwards the BUM traffic to the designated EVPN instances or VRFs, ensuring traffic reaches only the appropriate destinations.

Result

This process optimizes network efficiency by limiting BUM traffic replication to necessary locations, reducing unnecessary flooding and improving overall EVPN traffic management.

Split-horizon groups for EVPN E-LAN

A split-horizon group is a network loop prevention method that

- places forwarding or flooding restrictions between bridge ports based on group membership
- aggregates attachment circuits (ACs) and pseudowires (PWs) into one of three groups called split-horizon groups, and
- influences the flooding and forwarding behavior within bridge domains between members of a split-horizon group.

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
Split-Horizon Groups for EVPN Single-Homing	Release 26.2.1	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 for EVPN Single-Homing	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Split-Horizon Groups for EVPN Single-Homing	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D This feature is supported on Cisco 8711-48Z-M routers
Split-Horizon Groups for EVPN Single-Homing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
Split-Horizon Groups for EVPN Single-Homing	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.

Feature Name	Release Information	Feature Description
Split-Horizon Groups for EVPN Single-Homing	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: 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 for EVPN Single-Homing	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 for EVPN Single-Homing	Release 7.11.1	You can prevent unnecessary BUM traffic flooding and conserve bandwidth for single-homed EVPN scenarios by ensuring that the traffic isn't sent back to the CE device from which it originated. Depending on the type of traffic you need to be forwarded or distributed, you can configure split-horizon group 0 (SG 0), SG 1, or SG 2. This feature is supported only on Q200-based line cards. The feature introduces the **split-horizon group** command.

Split-horizon behavior in bridge domains

When applying the split-horizon method to bridge domains, these behaviors define traffic handling:

- Traffic between members of a split-horizon group is forwarded as either unicast or flooded.
- Forwarding restrictions are applied to prevent loops within the network.
- Group membership controls how traffic flows between members.

Split-horizon group forwarding and flooding behaviors

Traffic flooding occurs for broadcast, multicast, and unknown unicast destination addresses. Unicast traffic refers to frames sent to bridge ports where the destination MAC address is known.

Flooding traffic includes:

- Frames with unknown unicast destination MAC addresses.
- Frames sent to Ethernet multicast addresses, such as Spanning Tree Bridge Protocol Data Units (BPDUs).
- Ethernet broadcast frames with the MAC address FF-FF-FF-FF-FF-FF.

Within the network, members are organized into groups that control traffic flow:

- Members within the same group are prohibited from sending traffic to each other.
- Members in different groups can send traffic to each other without restrictions.

This grouping segments the network into unique routes, which improves traffic control and reduces packet congestion, thereby enhancing network performance and reliability.

The table summarizes the behavior of frames received on one member of a split-horizon group, including whether traffic is forwarded to other members of the same group, and the assignment of Bridge Ports (BPs) to groups

Split-horizon group	Behavior
0	Default AC group with no forwarding or flooding restrictions. Forwards and floods traffic within the group and between all groups. All L2 ACs are added to this group by default and cannot be manually assigned via CLI.
1	Default VFI (core) PW group. Forwarding or flooding of traffic is restricted between bridge ports in this group but allowed to all other groups. All EVPN EVI virtual ports and VFI PWs are added to this group and cannot be manually assigned via CLI.
2	Optional AC group. Forwarding or flooding of traffic is restricted between bridge ports in this group but allowed to all other groups. ACs can be manually added to this group via CLI, but not VFI PWs.

Configure the split-horizon groups for EVPN E-LAN

Configure split-horizon groups to control traffic forwarding and prevent loops within L2VPN bridge domains.

Use this task when setting up L2VPN bridge domains that require split-horizon group configuration to isolate traffic between interfaces and pseudowires.

Procedure

Step 1 Create a bridge group and enter a bridge domain within the bridge group.

Example:

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

Step 2 Configure interfaces and assign them to the bridge domain.

a) Assign interface HundredGigE 0/0/0/24 to the default split-horizon group 0.

Example:

```
Router(config-l2vpn-bg-bd-ac)# interface HundredGigE 0/0/0/24<- (split-horizon group 0, default)
```

b) Assign interface HundredGigE 0/0/0/24.1 without specifying a split-horizon group (inherits default).

Example:

```
Router(config-l2vpn-bg-bd-ac)# interface HundredGigE 0/0/0/24.1
```

Step 3 Configure an Ethernet Virtual Instance (EVI) and assign a split-horizon group.

Example:

```
Router(config-l2vpn-bg-bd)# evi 200
Router(config-l2vpn-bg-bd-evi)# split-horizon group<- (split-horizon group 2)
```

Step 4 Configure a pseudowire (PW) and assign it to a split-horizon group.

Example:

```
Router(config-l2vpn-bg-bd-evi)# neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-bg-bd-pw)# split-horizon group <- (split-horizon group 2)
```

Step 5 Configure a neighbor for the VFI with the default split-horizon group 1.

Example:

```
Router(config-l2vpn-bg-bd-pw)# vfi vf
Router(config-l2vpn-bg-bd-vfi)# neighbor 172.16.0.1 pw-id 10001 <- (split-horizon group 1, default)
Router(config-l2vpn-bg-bd-vfi)# commit
```

Step 6 Running configuration of split-horizon groups for EVPN E-LAN.

Example:

```
l2vpn
  bridge group bg
   bridge-domain bd
    interface HundredGigE 0/0/0/24 <- (split-horizon group 0, default)
    interface HundredGigE 0/0/0/24.1
   !

     split-horizon group <- (split-horizon group 2)
    neighbor 10.0.0.1 pw-id 1
     split-horizon group <- (split-horizon group 2)
    vfi vf
      neighbor 172.16.0.1 pw-id 10001 <- (split-horizon group 1, default)
  !
   evi 200
!
```

Step 7 Use the **show l2vpn bridge-domain detail** command to verify the split-horizon group configuration.

Example:

```
Router# show l2vpn bridge-domain detail | i "AC:|Split Horizon|PW:|VFI"
MAC withdraw for Access PW: enabled
Split Horizon Group: none
P2MP PW: disabled
ACs: 2 (2 up), VFIs: 1, PWs: 2 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
AC: HundredGigE 0/0/0/24, state is up
Split Horizon Group: none
AC: HundredGigE 0/0/0/24.1, state is up
Split Horizon Group: enabled
PW: neighbor 10.0.0.1, pw-id 1, state is up ( established )
Split Horizon Group: enabled
List of VFIs:
VFI vf (up)
```

```
PW: neighbor 172.16.0.1, pw-id 10001, state is up ( established )
Split Horizon Group: none
```

The split-horizon groups are configured for your EVPN E-LAN, isolating traffic as intended and preventing L2 forwarding loops.

Core isolation techniques in EVPN

Core isolation techniques in EVPN are network methods that

- segregate control and data plane operations to enhance network stability
- monitor interface states to maintain isolation in EVPN E-LAN environments, and
- detect and isolate faulty peers through peer failure detection to prevent network disruption.

This section outlines key core isolation methods in EVPN, focusing on interface tracking for EVPN E-LAN environments and peer failure detection capabilities that enhance network resilience and stability.

Core isolation by interface tracking for EVPN E-LAN

Core isolation is a network management technique that

- monitors the state of interfaces connecting to the core provider edge
- isolates the core provider edge device upon interface failure, and
- ensures the rest of the network remains operational and unaffected.

Table 20: Feature History Table

Feature Name	Release Information	Feature Description
Core Isolation by Interface Tracking for EVPN Single-Homing	Release 26.2.1	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.
Core Isolation by Interface Tracking for EVPN Single-Homing	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.

Feature Name	Release Information	Feature Description
Core Isolation by Interface Tracking for EVPN Single-Homing	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Core Isolation by Interface Tracking for EVPN Single-Homing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
Core Isolation by Interface Tracking for EVPN Single-Homing	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The core isolation by interface tracking functionality is now extended to the Cisco 8712-MOD-M routers.
Core Isolation by Interface Tracking for EVPN Single-Homing	Release 24.3.1	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*) *The core isolation by interface tracking functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Core Isolation by Interface Tracking for EVPN Single-Homing	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The core isolation by interface tracking functionality is now extended to routers with the 88-LC1-36EH line cards.

Feature Name	Release Information	Feature Description
Core Isolation by Interface Tracking for EVPN Single-Homing	Release 7.11.1	You can now monitor the connectivity of the customer-facing interface on the PE router using object tracking (OT). If the interface fails or becomes unavailable, the router isolates the customer site from the rest of the EVPN network. Isolating the core from the network prevents the customer site from advertising its routes to other sites on the EVPN single-home network, ensuring that traffic isn't sent to the failed PE router. Object tracking involves continuous monitoring of network interfaces, including links or ports on routers. By actively observing the status of these interfaces, administrators can dynamically adjust the network configuration based on their availability and health. This feature is supported only on Q200-based line cards. Use Object Tracking commands to track and monitor the connected interfaces.

Core provider edge isolation technique

The core provider edge isolationtechnique uses tracking objects assigned to the relevant interfaces. The tracking objects continuously check interface status, such as link up or link down, enabling isolation of the core provider edge device if any connected interface experiences issues or failures.

By isolating the core provider edge device in this way, network stability and continuity are maintained, preventing localized interface problems from impacting the broader network.

Supported functions of object tracking

Object tracking supports these capabilities:

- Monitors specific network elements (such as static routes or interface statuses) as tracked objects.
- Each tracked object is assigned a unique name using the track command in configuration mode.
- The router monitors the state of each tracked object:
  - up—the object is functioning as expected.
  - down—the object is not functioning.

When a tracked object's state changes, the tracking process receives a notification. The tracked object can be associated with a wide variety of actions, without requiring a direct relationship between the objects themselves.

Boolean logic functions for tracking lists:

- AND function—the tracked list is considered up only if every object in the subset is up.
- OR function—the tracked list is considered up if at least one object in the subset is up.

Benefits of object tracking

Object tracking offers significant advantages for network management by enabling continuous oversight and responsive control of critical network elements. The key benefits include:

- Real-time monitoring of network components, ensuring the reachability and availability of essential objects within the network.
- Automation of network management tasks, allowing dynamic failover and load balancing actions to be executed automatically based on the state changes of tracked objects.
- Enhanced network availability through rapid detection and response to status changes, facilitating proactive measures that reduce potential downtime.
- Policy-driven actions that trigger specific responses when a tracked object's state changes, providing flexible and tailored network management aligned with current network conditions.

How core isolation by interface tracking works

This process addresses how to detect and isolate link failures affecting traffic flow from CE1 to PE1, focusing on distinguishing between local and remote link failures.

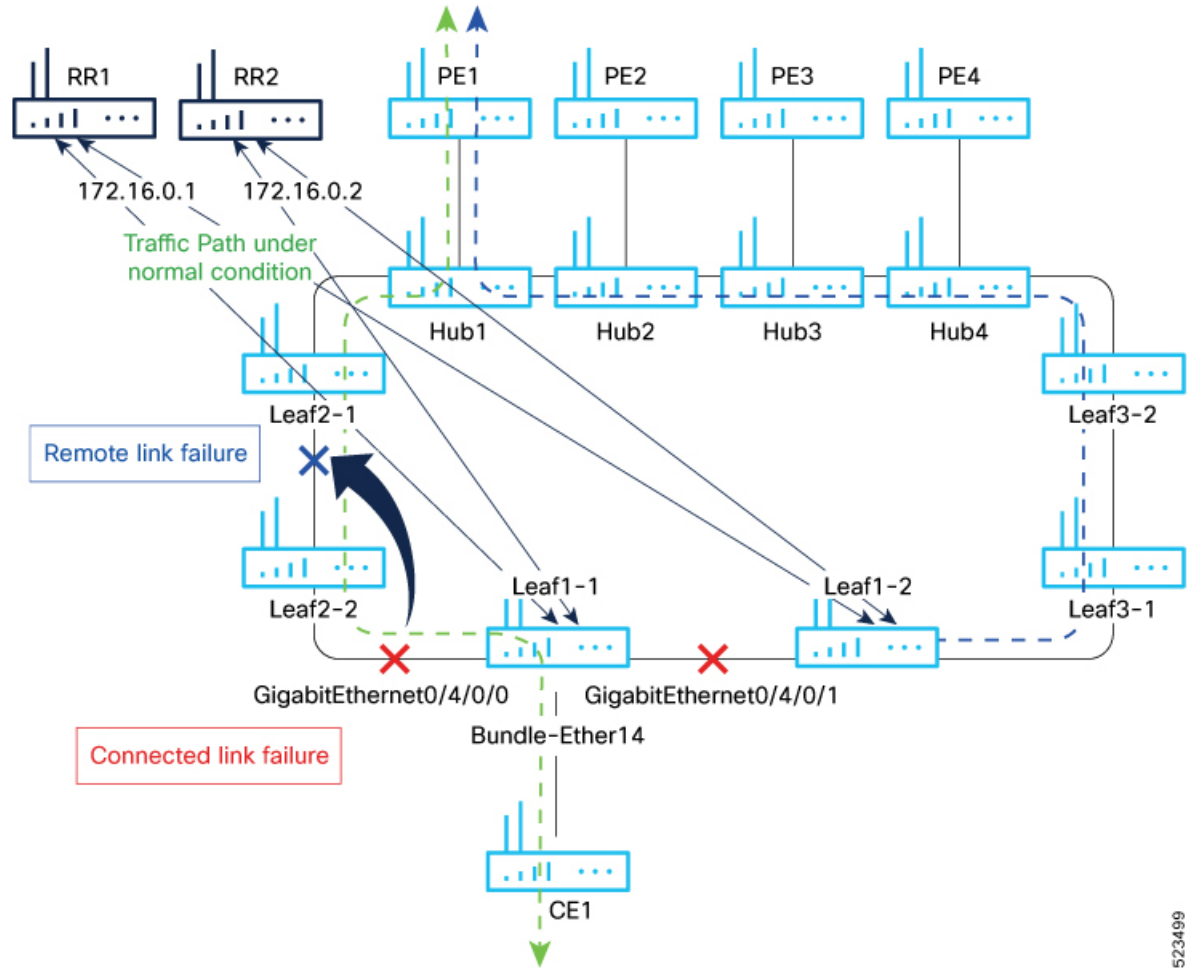
Summary

The key components involved in the core isolation process are:

- CE1: The customer edge device that initiates traffic from Leaf1-1.
- Leaf1-1: The leaf switch that forwards traffic and monitors link connectivity.
- Route reflectors (RRs): Devices maintaining BGP sessions with Leaf1-1.
- Bundle-Ether14: The interface bundle connecting Leaf1-1 to CE1.

Workflow

Figure 10: Core isolation by interface tracking



These stages describe how core isolation by interface tracking works:

1. CE1 sends traffic through Leaf1-1 towards PE1.
2. Leaf1-1 monitors connectivity on both local and remote links.
3. If Leaf1-1 loses connectivity on both local and remote links, BGP sessions to both route reflectors go down.
4. Upon BGP session failure, Leaf1-1 disables the Bundle-Ether14 interface connected to CE1.
5. Interface tracking is used to detect local link failures by monitoring connected interfaces.
6. For remote link failures, interface tracking alone is insufficient; BGP session tracking is required to identify these failures.

Result

This process enables accurate detection and isolation of link failures by combining interface and BGP session tracking, ensuring appropriate interface shutdowns to maintain network stability.

Configure EVPN core isolation by interface tracking

Configure EVPN core isolation to monitor interface and BGP neighbor states and trigger actions based on their status.

Use this task to isolate the EVPN core by tracking interface line protocol states and BGP neighbor address-family states, combining these tracked objects with Boolean logic to control interface behavior.

Note

- An object must exist before it can be added to a tracked list.

  A tracked list contains one or more objects. The Boolean expression enables tracking objects using either AND or OR operators. For example, when tracking two interfaces, using the AND operator, up means that *both* interfaces are up, and down means that *either* interface is down.

- The NOT operator is specified for one or more objects and negates the state of the object.
- After configuring the tracked object, you must associate the neighbor or interface whose state must be tracked.

Before you begin

In this example, Leaf1-1 brings the down the AC connected to CE1 when:

Both local interfaces GigabitEthernet0/4/0/0 and GigabitEthernet0/4/0/1of Leaf1-1 are down.

OR

Leaf1-1 BGP sessions to both RRs are down.

Perform this task on Leaf1-1:

Procedure

Step 1 Configure BGP for EVPN.

Example:

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# commit
```

Step 2 Track the line protocol state of interfaces.

Example:

```
Router# configure
Router(config)# track interface-1
Router(config-track)# type line-protocol state
Router(config-track-line-prot)# interface HundredGigE0/0/0/24
```

```
Router(config-track-line-prot)#exit
Router(config-track)#exit
Router(config)# track interface-2
Router(config-track)# type line-protocol state
Router(config-track-line-prot)# interface HundredGigE0/0/0/25
Router(config-track-line-prot)#exit
Router(config-track)#exit
Router(config)# track interface-group-1
Router(config-track)# type list boolean or
Router(config-track-list-boolean)# object interface-1
Router(config-track-list-boolean)# object interface-2
Router(config-track-list-boolean)# commit
```

Step 3 Track neighbor address-family state.

Example:

```
Router# configure
Router(config)# track neighbor-A
Router(config-track)# type bgp neighbor address-family state
Router(config-track-bgp-nbr-af)# address-family l2vpn evpn
Router(config-track-bgp-neighbor)# neighbor 172.16.0.1
Router(config-track-bgp-neighbor)# exit
Router(config-track-bgp-nbr-af)# exit
Router(config-track)# exit
Router(config)# track neighbor-B
Router(config-track)# type bgp neighbor address-family state
Router(config-track-bgp-nbr-af)# address-family l2vpn evpn
Router(config-track-bgp-neighbor)# neighbor 172.16.0.2
Router(config-track-bgp-neighbor)# exit
Router(config-track-bgp-nbr-af)# exit
Router(config-track)# exit
Router(config)# track neighbor-group-1
Router(config-track)# type list boolean or
Router(config-track-list-boolean)# object neighbor-A
Router(config-track-list-boolean)# object neighbor-B
Router(config-track-list-boolean)# commit
```

Step 4 Track objects for both interfaces and neighbors.

Example:

```
Router# configure
Router(config)# track core-group-1
Router(config-track)# type list boolean and
Router(config-track-list-boolean)# object neighbor-group-1
Router(config-track-list-boolean)# object interface-group-1
Router(config-track-list-boolean)# action
Router(config-track-action)# track-down error-disable interface Bundle-Ether14 auto-recover
Router(config-track-action)# commit
```

Step 5 Running configuration of EVPN core isolation.

Example:

```
router bgp 100
 address-family l2vpn evpn
 !
 neighbor 172.16.0.1
  remote-as 100
  address-family l2vpn evpn
  !
 !
```

```
 neighbor 172.16.0.2
  remote-as 100
  address-family l2vpn evpn
  !
 !
!

track interface-1
 type line-protocol state
  interface HundredGigE0/0/0/24
 !
!
track interface-2
 type line-protocol state
  interface HundredGigE0/0/0/25
 !
!
track interface-group-1
 type list boolean or
  object interface-1
  object interface-2
 !
!

track neighbor-A
 type bgp neighbor address-family state
  address-family l2vpn evpn
   neighbor 172.16.0.1
  !
 !
!
track neighbor-B
 type bgp neighbor address-family state
  address-family l2vpn evpn
   neighbor 172.16.0.1
  !
 !
!

track neighbor-group-1
 type list boolean or
  object neighbor-A
  object neighbor-B
  !
 !
!
track core-group-1
 type list boolean and
  object neighbor-group-1
  object interface-group-1
 !
 action
  track-down error-disable interface Bundle-Ether14 auto-recover
  !
 !
```

Step 6 Use these **show track**, **show track brief**, and **show bgp track** commands to verify and track the status of interfaces and core group. These show output examples display the status of interfaces and tracks as UP.

Example:

```
Router# show track

Track neighbor-A
```

```
        BGP Neighbor AF L2VPN EVPN NBR 172.16.0.1 vrf default
        Reachability is UP
                Neighbor Address Reachablity is Up
                BGP Neighbor Address-family state is Up
        4 changes, last change UTC Tue May 26 2020 20:14:33.171

Track neighbor-B
        BGP Neighbor AF L2VPN EVPN NBR 172.16.0.2 vrf default
        Reachability is UP
                Neighbor Address Reachablity is Up
                BGP Neighbor Address-family state is Up
        4 changes, last change UTC Tue May 26 2020 20:14:27.527

Track core-group-1
        List boolean and is UP
        2 changes, last change 20:14:27 UTC Tue May 26 2020
                object interface-group-1 UP
                object neighbor-group-1 UP

Track interface-1
        Interface HundredGigE0/0/0/24 line-protocol
        Line protocol is UP
        2 changes, last change 20:13:32 UTC Tue May 26 2020

Track interface-2
        Interface HundredGigE0/0/0/25 line-protocol
        Line protocol is UP
        2 changes, last change 20:13:28 UTC Tue May 26 2020

Track interface-group-1
        List boolean or is UP
        2 changes, last change 20:13:28 UTC Tue May 26 2020
                object interface-2 UP
                object interface-1 UP

Track neighbor-group-1
        List boolean or is UP
        2 changes, last change 20:14:27 UTC Tue May 26 2020
                object neighbor-A UP
                object neighbor-B UP


Router# show track brief
Track                           Object                                          Parameter         Value

-------------------------------------------------------------------------------------------------------------
neighbor-A                      bgp nbr L2VPN EVPN 172.16.0.1 vrf default reachability     Up

neighbor-B                      bgp nbr L2VPN EVPN 172.16.0.1 vrf default reachability     Up

core-group-1                    list                                            boolean and      Up
interface-1                     interface HundredGigE0/0/0/24                   line protocol    Up
interface-2                     interface HundredGigE0/0/0/25                   line protocol    Up
interface-group-1               list                                            boolean or       Up
neighbor-group-1                list                                            boolean or       Up
-------------------------------------------------------------------------------------------------------------

Router# show bgp track
Wed May 27 05:05:51.285 UTC

VRF                   Address-family       Neighbor         Status     Flags

default               L2VPN EVPN           172.16.0.1     UP         0x01
default               L2VPN EVPN           172.16.0.2     UP         0x01
```

```
Processed 2 entries
```

The router monitors the specified interfaces and BGP neighbors. If both interfaces or both BGP sessions go down, the configured interface is error-disabled automatically, isolating the EVPN core.

EVPN core isolation through peer failure detection

A core link failure detection and isolation technique is a network capability that

- detects failures in core links within the EVPN network
- isolates the affected provider edge (PE) device from the network, and
- maintains the stability, security, and efficiency of the EVPN network.

This capability is especially valuable in large-scale deployments such as data centers. When a core link failure occurs on a PE device, EVPN disables the PE's Ethernet Segment (ES) associated with the access interface connected to the customer edge (CE) device. This action effectively isolates the faulty PE device, ensuring continued operation and security of the overall EVPN network by segregating different segments or sites within the core network.

Table 21: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Core Isolation through Peer Failure Detection	Release 26.2.1	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 Core Isolation through Peer Failure Detection	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
EVPN Core Isolation through Peer Failure Detection	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN Core Isolation through Peer Failure Detection	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
EVPN Core Isolation through Peer Failure Detection	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The EVPN Core Isolation through Peer Failure Detection functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN Core Isolation through Peer Failure Detection	Release 24.3.1	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*) *The EVPN Core Isolation through Peer Failure Detection functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN Core Isolation through Peer Failure Detection	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The EVPN Core Isolation through Peer Failure Detection functionality is now extended to the 88-LC1-36EH line cards.
EVPN Core Isolation through Peer Failure Detection	Release 7.11.1	You can now isolate the provider edge (PE) device from the network when there is a core link failure, preventing traffic disruptions and data leakage that could result from a compromised or malfunctioning peer. Upon detecting a link failure, the affected PE device is isolated from the core network, and the EVPN brings down the PE's Ethernet Segment (ES), which is associated with the access interface attached to the customer edge (CE) device. This feature is supported only on Q200-based line cards. The feature introduces the **core-isolation-group** command.

Core link failure impact on provider edge device

When a core link failure occurs on a PE device, EVPN disables the PE’s Ethernet Segment (ES) linked to the access interface connected to the CE device.

- Core interfaces are configured within an EVPN group and linked to the Ethernet segment, which acts as an attachment circuit (AC) connected to the CE.
- If all core interfaces in the group fail, EVPN disables the associated access interfaces and prevents the CE device from using those links within its bundles.
- When all interfaces in the group are down, EVPN disables the entire bundle and withdraws the ES-EAD route.

Restrictions for EVPN core isolation through peer failure detection group and interface

When configuring EVPN core isolation using peer failure detection groups and interfaces, adhere to these restrictions:

- Limit the number of EVPN groups to a maximum of 24.
- Assign no more than 12 core interfaces to each group.
- You may reuse core interfaces across multiple groups as needed.
- Use bundle interfaces exclusively for core interfaces.
- Include only core interfaces in an EVPN group; do not add access interfaces.
- Restrict access interfaces to bundle interfaces only.

How EVPN core isolation protection works

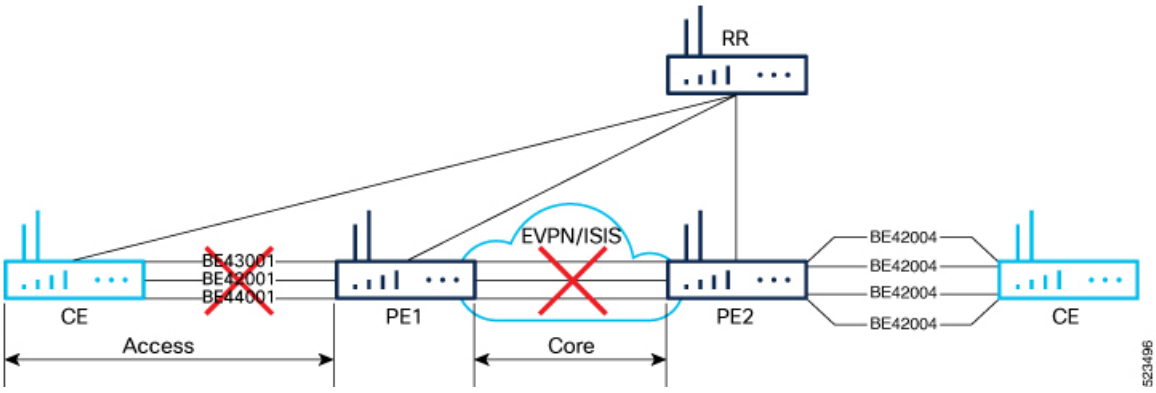
Summary

The key components involved in the EVPN core isolation protection process are:

- CE: The device connected to the provider edge router that sends and receives traffic.
- PE1: A router connected to CE and part of the EVPN-MPLS core network.
- PE2: Another router in the EVPN/MPLS core network running EVPN with PE1.
- Core interfaces: Interfaces in the MPLS core network, which can be Gigabit Ethernet or bundle interfaces.
- Access interface: The interface connecting PE1 to CE, which must be a bundle interface.
- BGP session: The Border Gateway Protocol session between PE routers used to exchange routing information.
- Ethernet A-D Ethernet Segment (ES-EAD) routes: Routes advertised by PE1 to signal its presence and services in the EVPN core.

Workflow

Figure 11: EVPN core isolation protection



These stages describe how EVPN core isolation protection works.

1. The CE connects to PE1, which along with PE2 runs EVPN over the MPLS core network. Core interfaces can be Gigabit Ethernet or bundle interfaces, while the access interface is a bundle interface only.
2. When the core links of PE1 fail, EVPN detects the failure and isolates PE1 from the core network by shutting down the access interface. This prevents CE from sending traffic to PE1.
3. The BGP session between PE1 and PE2 also goes down, causing BGP to invalidate all routes advertised by the failed PE1.
4. When the core interfaces and BGP sessions are restored, PE1 re-advertises its Ethernet Segment routes (ES-EAD), triggering service restoration and reintegration into the core network.

Result

This process ensures that when core links fail, the affected PE1 is isolated to prevent traffic loss or loops, and when connectivity is restored, PE1 seamlessly rejoins the EVPN core network, maintaining service stability and integrity.

Configure EVPN core isolation through peer failure detection

Configure core interfaces within EVPN groups and link these groups to attachment circuits (AC) connected to EVPN single-homing CE devices.

This task applies when setting up EVPN core isolation groups and associating them with bundle interfaces to manage connectivity and redundancy.

Procedure

Step 1 Configure the EVPN core isolation groups and assign core interfaces.

a) Assign core interfaces for group 42001.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# group 42001
Router(config-evpn-group)# core interface HundredGigE 0/0/0/24
Router(config-evpn-group)# core interface HundredGigE 0/0/0/25
Router(config-evpn-group)#exit
```

b) Assign core interfaces for group 43001.

Example:

```
Router(config-evpn)# group 43001
Router(config-evpn-group)# core interface HundredGigE 0/0/0/26
Router(config-evpn-group)# core interface HundredGigE 0/0/0/27
Router(config-evpn-group)#exit
```

Step 2 Associate the EVPN groups with attachment circuits.

a) Assign the core-isolation-group 42001 to interface bundle-Ether 42001.

Example:

```
Router# configure
Router(config)# evpn
```

```
Router(config-evpn)# interface bundle-Ether 42001
Router(config-evpn-ac)# core-isolation-group 42001
Router(config-evpn-ac)# exit
```

b) Assign the core-isolation-group 43001 to interface bundle-Ether 43001.

Example:

```
Router(config-evpn)# interface bundle-Ether 43001
Router(config-evpn-ac)# core-isolation-group 43001
Router(config-evpn-ac)# commit
```

Step 3 Running configuration of EVPN core isolation through peer failure detection.

Example:

```
configure
 evpn
  group 42001
   core interface HundredGigE 0/0/0/24
   core interface HundredGigE 0/0/0/25
   !
  group 43001
   core interface HundredGigE 0/0/0/26
   core interface HundredGigE 0/0/0/27
   !
 !
configure
 evpn
  interface bundle-Ether 42001
   core-isolation-group 42001
   !
  interface bundle-Ether 43001
   core-isolation-group 43001
   !
  !
```

Step 4 Use the **show evpn group** command to display the complete list of EVPN groups, their associated core interfaces and access interfaces.

The status, up or down, of each interface is displayed. For the access interface to be up, at least one of the core interfaces must be up. The output shows that the core is isolated because the core interfaces are down, bringing down the access interfaces connected to this core.

Example:

```
Router# show evpn group
EVPN Group: 42001
  State: Isolated
  Core Interfaces:
     HundredGigE 0/0/0/24: down
     HundredGigE 0/0/0/25: down

  Access Interfaces:
     Bundle-Ether42001: down
```

The output shows that the core is in the Ready state because both the core and access interfaces are UP.

Example:

```
Router# show evpn group
EVPN Group: 43001
  State: Ready
```

```
Core Interfaces:
   HundredGigE 0/0/0/26: up
   HundredGigE 0/0/0/27: up

Access Interfaces:
   Bundle-Ether43001: up
```

The EVPN core interfaces are configured under their respective groups, and these groups are linked to the corresponding attachment circuits. The access interface status depends on the core interfaces' status; at least one core interface must be up for the access interface to be up.

EVPN cost-out

An EVPN cost-out is a network capability that

- controls the operational state of bundle interfaces within an Ethernet segment configured with Link Aggregation Control Protocol (LACP)
- enables placing a node out of service (OOS) without manually shutting down all bundle interfaces on the PE device, and
- facilitates preparation of the node for reload or software upgrade.

Table 22: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Cost-Out	Release 26.2.1	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 Cost-Out	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN Cost-Out	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN Cost-Out	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The EVPN Cost-Out functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
EVPN Cost-Out	Release 24.3.1	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*) *The EVPN Cost-Out functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN Cost-Out	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The cost-out node brings down the bundle interfaces on the PE to prepare the node for reload or software upgrade. By costing out a node, the traffic is steered away from the PE without any traffic disruption. This allows you to manage the network traffic effectively while reloading or upgrading a node. *This feature is supported only on routers with the 88-LC1-36EH line cards.

Traffic redirection using EVPN cost-out for Ethernet segments

The EVPN cost-out process enables seamless traffic redirection in multihomed EVPN environments by gracefully withdrawing a node's bundle interfaces from service.

- The router withdraws Ethernet A-D Ethernet Segment (ES-EAD) routes before shutting down the associated bundle interfaces.
- The PE device notifies the connected CE device to bring down the corresponding bundle member interface.
- Traffic is automatically steered away from the affected PE node, ensuring no disruption to ongoing services.
- In a multihoming scenario, any traffic from the CE destined for the Ethernet segment is redirected to a peer PE device.

EVPN cost-out is supported only on Ethernet segments that have manually configured Ethernet Segment Identifiers (ESIs).

How EVPN cost-out works

Summary

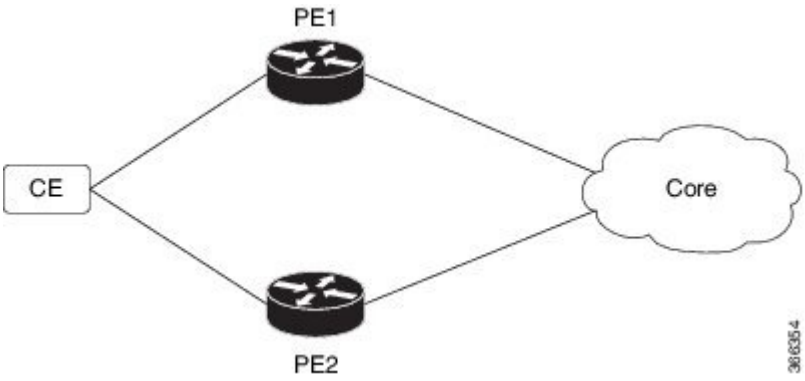
The key components involved in the EVPN cost-out process are:

- CE device: Connects to multiple PE devices and forwards traffic accordingly.
- PE device: Hosts bundle interfaces on Ethernet segments and controls their operational state.

- Cost-out command: Controls the operational state of bundle interfaces on PE and CE devices.
- Startup-cost-in command: Manages timed transition of the node from cost-out to cost-in state after reload.

Workflow

Figure 12: EVPN cost-out



These are the stages of EVPN cost-out.

1. When the **cost-out** command is configured on PE1, all bundle interfaces on the Ethernet segment are brought down on PE1. Correspondingly, the related bundle member on the CE is also brought down. This causes the CE to send all traffic for that Ethernet segment to PE2.
2. To bring the node back into service, the **no cost-out**command is used. This command brings up all bundle interfaces on the EVPN Ethernet segment on the PE and the corresponding bundle members on the CE.
3. While the node is in the cost-out state:
   - Adding a new bundle Ethernet segment causes that bundle to be brought down.
   - Removing a bundle Ethernet segment causes that bundle to be brought up.
4. The **startup-cost-in** command allows the node to come into service automatically after a specified time following a reload. Initially, the node enters the cost-out state when EVPN initializes and remains so until the timer expires.
5. If the **no startup-cost-in** command is executed while the timer is running, the timer stops and the node transitions to the cost-in state.
6. The **cost-out** configuration always takes precedence over the **startup-cost-in** timer. Therefore:
   - If both configurations are present during a reload, the cost-out state is controlled solely by the **cost-out** command, and the timer is ignored.
   - If the startup timer is running and the **cost-out** command is configured, the timer stops, and the out-of-service state is controlled exclusively by the **cost-out** configuration.
7. If a process restart occurs while the **startup-cost-in** timer is running, the node remains in the cost-out state and the timer restarts.

Result

This process ensures controlled failover and recovery of traffic between PE devices in an EVPN environment by managing the operational state of bundle interfaces on both PE and CE devices.

Configure EVPN cost-out and startup cost-in timer

Configure EVPN cost-out to bring down a node on a PE device and set the startup cost-in timer to control the node's startup delay after reload.

Use this task when you need to administratively bring down an EVPN node or delay its startup after a reload to manage network stability and convergence.

Procedure

Step 1 Configure cost-out to bring down the EVPN node on the PE.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# cost-out
Router(config-evpn)commit
```

Step 2 Bring the EVPN node back into service.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# no cost-out
Router(config-evpn)commit
```

Step 3 Configure the startup cost-in timer to delay node startup after reload.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# startup-cost-in 6000
Router(config-evpn)commit
```

Step 4 Running configuration of EVPN cost-out and startup cost-in timer.

Example:

```
configure
evpn
 cost-out
!

configure
evpn
 startup-cost-in 6000
!
```

Step 5 Use the **show evpn summary** command to verify the cost-out and startup cost-in timer configurations.

Example:

Verify the node cost-out configuration.

```
Router# show evpn summary
Fri Apr  7 07:45:22.311 IST
Global Information
----------------------------
Number of EVIs                : 2
Number of Local EAD Entries   : 0
Number of Remote EAD Entries  : 0
Number of Local MAC Routes    : 0
Number of Local MAC Routes    : 5
          MAC                 : 5
          MAC-IPv4            : 0
          MAC-IPv6            : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes   : 7
          MAC                 : 7
          MAC-IPv4            : 0
          MAC-IPv6            : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels     : 5
Number of ES Entries          : 9
Number of Neighbor Entries    : 1
EVPN Router ID                : 192.168.0.1
BGP Router ID                 : ::
BGP ASN                       : 100
PBB BSA MAC address           : 0207.1fee.be00
Global peering timer          :      3 seconds
Global recovery timer         :     30 seconds
EVPN cost-out                 : TRUE
      startup-cost-in timer   : Not configured
```

Verify the no cost-out configuration.

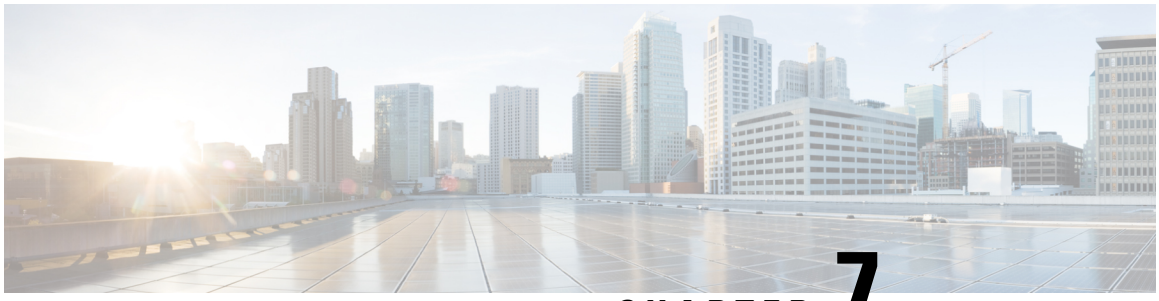
```
Router# show evpn summary
Fri Apr  7 07:45:22.311 IST
Global Information
----------------------------
Number of EVIs                : 2
Number of Local EAD Entries   : 0
Number of Remote EAD Entries  : 0
Number of Local MAC Routes    : 0
Number of Local MAC Routes    : 5
          MAC                 : 5
          MAC-IPv4            : 0
          MAC-IPv6            : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes   : 7
          MAC                 : 7
          MAC-IPv4            : 0
          MAC-IPv6            : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels     : 5
Number of ES Entries          : 9
Number of Neighbor Entries    : 1
EVPN Router ID                : 192.168.0.1
BGP Router ID                 : ::
BGP ASN                       : 100
PBB BSA MAC address           : 0207.1fee.be00
Global peering timer          :      3 seconds
Global recovery timer         :     30 seconds
```

```
EVPN cost-out                 : FALSE
      startup-cost-in timer   : Not configured
```

Verify the startup-cost-in timer configuration.

```
Router# show evpn summary
Fri Apr  7 07:45:22.311 IST
Global Information
------------------------------
Number of EVIs                : 2
Number of Local EAD Entries   : 0
Number of Remote EAD Entries  : 0
Number of Local MAC Routes    : 0
Number of Local MAC Routes    : 5
          MAC                 : 5
          MAC-IPv4            : 0
          MAC-IPv6            : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes   : 7
          MAC                 : 7
          MAC-IPv4            : 0
          MAC-IPv6            : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels     : 5
Number of ES Entries          : 9
Number of Neighbor Entries    : 1
EVPN Router ID                : 192.168.0.1
BGP Router ID                 : ::
BGP ASN                       : 100
PBB BSA MAC address           : 0207.1fee.be00
Global peering timer          :      3 seconds
Global recovery timer         :     30 seconds
EVPN node cost-out            : TRUE
        startup-cost-in timer : 6000
```

The EVPN node is administratively brought down or brought up as needed, and the startup cost-in timer controls the delay before the node becomes active after a reload.



CHAPTER 7

Advanced EVPN Configurations and Extensions

VRF leaking for EVPN E-LAN

A virtual routing and forwarding (VRF) instance is a network virtualization technology that

- provides logical separation of network resources by creating multiple isolated virtual networks
- operates independently with its own routing table, forwarding behavior, and network policies, and
- enables communication among devices within the same VRF while isolating them from devices in other VRFs.

Table 23: Feature History Table

Feature Name	Release Information	Feature Description
VRF Leaking for EVPN Single-Homing	Release 26.2.1	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.
VRF Leaking for EVPN Single-Homing	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
VRF Leaking for EVPN Single-Homing	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Feature Name	Release Information	Feature Description
VRF Leaking for EVPN Single-Homing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
VRF Leaking for EVPN Single-Homing	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The VRF leaking functionality is now extended to the Cisco 8712-MOD-M routers.
VRF Leaking for EVPN Single-Homing	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The VRF leaking functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
VRF Leaking for EVPN Single-Homing	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The VRF leaking functionality is now extended to routers with the 88-LC1-36EH line cards.
VRF Leaking for EVPN Single-Homing	Release 7.11.1	We now allow for seamless intercommunication between different VRF instances in an EVPN domain, thus enabling controlled inter-VRF communication and resource-sharing, which is helpful in multi-tenancy environments, data center deployments, and hybrid cloud scenarios. This feature is supported only on Q200-based line cards.

Features of Layer 2 interconnection using VRF leaking

Layer 2 interconnection using VRF leaking in an EVPN network provides these features:

- Enables controlled Layer 2 communication between different VRF instances by selectively sharing routes.
- Maintains VRF isolation and segmentation while allowing traffic interconnection through EVPN Route type 2 (MAC+IP) import.
- Permits interconnection of VRFs at Layer 2 using gateways or bridges that forward traffic between VRFs.
- Allows definition of traffic policies to control flow, including filtering based on EVPN EVI and MAC addresses.
- Forwards Layer 2 frames between VRFs while preserving Layer 3 isolation.

Configure VRF leaking for EVPN E-LAN

This procedure enables controlled route leaking between the global routing table and a VRF routing table within an EVPN E-LAN environment. VRF leaking allows selective sharing of routes between VRFs and the global routing table, facilitating communication across different routing domains while maintaining segmentation and policy control. This is essential in multi-tenant or segmented network architectures where certain routes need to be shared securely and efficiently.

Procedure

Step 1 Configure BGP where the router performs the route leak.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 10.10.10.10
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# network 172.16.20.0/24
Router(config-bgp-af)# exit
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# vrf ORANGE
Router(config-bgp-vrf)# rd 100:100
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# network 192.168.10.0/24
Router(config-bgp-vrf-af)# commit
```

Step 2 Configure the route policies.

These policies help you filter which prefixes are permitted to be leaked. In this example, the *route-policy GLOBAL-2-VRF* and *route-policy VRF-2-GLOBAL* are used.

Example:

```
Router(config)# route-policy GLOBAL-2-VRF
Router(config-rpl)# if destination in (172.16.20.0/24) then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
Router(config)# route-policy VRF-2-GLOBAL
Router(config-rpl)# if destination in (192.168.10.0/24 le 32) then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
```

Step 3 Configure the VRF and apply the route-policy.

Example:

```
Router(config)# vrf ORANGE
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import from default-vrf route-policy GLOBAL-2-VRF
Router(config-vrf-af)# import route-target
Router(config-vrf-import-rt)# 100:100
Router(config-vrf-import-rt)# exit
Router(config-vrf-af)# export to default-vrf route-policy VRF-2-GLOBAL
Router(config-vrf-af)# export route-target
```

```
Router(config-vrf-export-rt)# 100:100
Router(config-vrf-export-rt)# commit
```

Step 4 Running configuration of VRF leaking.

Example:

```
router bgp 100
 bgp router-id 10.10.10.10
 address-family ipv4 unicast
  network 172.16.20.0/24
 !
 address-family vpnv4 unicast
 !
 vrf ORANGE
  rd 100:100
  address-family ipv4 unicast
   network 192.168.10.0/24
  !
 !
!
route-policy GLOBAL-2-VRF
  if destination in (172.16.20.0/24) then
    pass
  endif
end-policy
!
route-policy VRF-2-GLOBAL
  if destination in (192.168.10.0/24 le 32) then
    pass
  endif
end-policy
!
vrf ORANGE
 address-family ipv4 unicast
  import from default-vrf route-policy GLOBAL-2-VRF
  import route-target
   100:100
  !
  export to default-vrf route-policy VRF-2-GLOBAL
  export route-target
   100:100
  !
 !
!
```

Step 5 Use the **show route** command to verify the prefixes appear in the RIB and BGP tables.

Example:

```
Router# show route
Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path

Gateway of last resort is not set

C    10.88.174.0/24 is directly connected, 1d20h, MgmtEth0/RSP0/CPU0/0
```

```
L    10.88.174.223/32 is directly connected, 1d20h, MgmtEth0/RSP0/CPU0/0
L    10.10.10.10/32 is directly connected, 04:33:44, Loopback100
C    172.16.20.0/24 is directly connected, 07:03:18, HundredGigE0/0/0/24
L    172.16.20.1/32 is directly connected, 07:03:18, HundredGigE0/0/0/24
B    192.168.10.0/24 is directly connected, 03:02:21, HundredGigE0/0/0/0 (nexthop in vrf ORANGE)

Router# show ip bgp
BGP router identifier 10.10.10.10, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 5
BGP main routing table version 5
BGP NSR Initial initsync version 3 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 172.16.20.0/24     0.0.0.0                  0         32768 i
*> 192.168.10.0/24    0.0.0.0                  0         32768 i

Processed 2 prefixes, 2 paths
```

This show output displays the information for the VRF ORANGE:

```
Router# show route vrf ORANGE
Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path

Gateway of last resort is not set

B    172.16.20.0/24 is directly connected, 01:43:49, HundredGigE0/0/0/24 (nexthop in vrf default)
   C    192.168.10.0/24 is directly connected, 07:06:38, HundredGigE0/0/0/24
   L    192.168.10.2/32 is directly connected, 07:06:38, HundredGigE0/0/0/0


Router# show bgp vrf ORANGE
BGP VRF ORANGE, state: Active
BGP Route Distinguisher: 100:100
VRF ID: 0x60000003
BGP router identifier 10.10.10.10, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000012   RD version: 9
BGP main routing table version 9
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:100 (default for vrf ORANGE)
*> 172.16.20.0/24     0.0.0.0                  0         32768 i
```

```
*> 192.168.10.0/24    0.0.0.0                  0         32768 i

Processed 2 prefixes, 2 paths
```

MAC mobility for EVPN E-LAN

MAC mobility is a network capability that

- allows devices or virtual machines to move between different physical hosts or locations within a network
- enables efficient resource utilization through dynamic traffic distribution and optimized routing based on MAC address location, and
- maintains uninterrupted network connectivity during such moves.

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
MAC Mobility for EVPN Single-Homing	Release 26.2.1	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.
MAC Mobility for EVPN Single-Homing	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
MAC Mobility for EVPN Single-Homing	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
MAC Mobility for EVPN Single-Homing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
MAC Mobility for EVPN Single-Homing	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The MAC mobility functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
MAC Mobility for EVPN Single-Homing	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The MAC mobility functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
MAC Mobility for EVPN Single-Homing	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The MAC mobility functionality is now extended to routers with the 88-LC1-36EH line cards.
MAC Mobility for EVPN Single-Homing	Release 7.11.1	Now, it is now possible to seamlessly move MAC addresses between various network devices or locations while preserving their connectivity and associated network services. This ensures uninterrupted communication for devices or virtual machines frequently changing their physical or virtual location within the network. The L2 gateway dynamically updates its forwarding table when a MAC address moves from one device to another within the EVPN E-LAN network, guaranteeing that packets destined for that MAC address are correctly forwarded to its new location. This feature is supported only on Q200-based line cards.

Feature highlights of MAC mobility for EVPN E-LAN

- Facilitates seamless movement of MAC addresses among different devices or network locations, maintaining uninterrupted connectivity. This agility allows devices or virtual machines to be flexible and mobile, accommodating dynamic workloads and efficient resource allocation.
- Manages a substantial volume of mobile devices or virtual machines, permitting seamless movement across different network segments without causing disruptions or requiring manual reconfiguration.
- Ensures that packets are appropriately forwarded to the updated MAC address locations, optimizing routing decisions, curbing unnecessary traffic, and enhancing overall network performance.
- Eliminates the need for manual configuration changes when devices or virtual machines move within the network. This simplifies network management and reduces the likelihood of human errors or misconfigurations.

MAC Mobility includes the capability to detect and block duplicate MAC addresses, which enhances network stability and security. This function supports seamless mobility by preventing address conflicts that could disrupt connectivity or degrade performance, thereby maintaining reliable and efficient network operations.

Detect and block duplicate MAC addresses

Duplicate MAC address detection is a network capability that

- identifies hosts with duplicate MAC addresses
- blocks all routes associated with these duplicate addresses, and
- prevents network instability caused by address conflicts.

Table 25: Feature History Table

Feature Name	Release Information	Feature Description
Detect and Block Duplicate MAC Addresses	Release 26.2.1	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.
Detect and Block Duplicate MAC Addresses	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) * This feature is now supported on Cisco 8404-SYS-D routers.
Detect and Block Duplicate MAC Addresses	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Detect and Block Duplicate MAC Addresses	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
Detect and Block Duplicate MAC Addresses	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The Detect and Block Duplicate MAC Addresses funtionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
Detect and Block Duplicate MAC Addresses	Release 24.3.1	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*) * The Detect and Block Duplicate MAC Addresses funtionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Detect and Block Duplicate MAC Addresses	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) * The Detect and Block Duplicate MAC Addresses funtionality is now extended to routers with the 88-LC1-36EH line cards.
Detect and Block Duplicate MAC Addresses	Release 7.11.1	You can now effectively mitigate traffic disruptions, packet loss, and potential network outages in your network operations by detecting and freezing duplicate MAC addresses and blocking all associated routes. This feature is supported only on Q200-based line cards. The feature introduces the **evpn mac secure** command.

Handling duplicate MAC addresses in network devices

Multiple devices using the same MAC address cause MAC address flapping. This issue occurs when different devices intermittently claim the identical MAC address, forcing network devices to repeatedly update their forwarding tables. Duplicate MAC addresses can result in:

- Excessive network traffic
- Increased CPU load on network devices
- Overall network instability

To overcome such issues, routers detect duplicate MAC addresses based on predefined parameters and freeze the offending MAC address to prevent further disruptions. However, a configurable option allows you to avoid permanently freezing the MAC address, providing greater flexibility in network management.

Handling MAC address mobility and duplicate detection in EVPN hosts

The router tracks MAC addresses as they move between hosts to manage EVPN host mobility. When two hosts share the same MAC address, the router learns and relearns the MAC routes from each host. Each new learning of a MAC route from a different host counts as one move, superseding the previous route. This back-and-forth learning continues until the router marks the MAC address as a duplicate based on configured parameters.

Use the **evpn mac secure** command to configure when the router marks a MAC address as duplicate and whether to freeze or unfreeze it during movement between hosts. The key configurable parameters are

- **move-count**: Number of times a MAC address changes location between hosts within a specified period to be considered duplicate.
- **move-interval**: Time period during which the MAC address must move the specified number of times to trigger duplicate detection.
- **freeze-time**: Duration the MAC address remains locked after being detected as duplicate. After this, it unlocks and can be relearned.
- **retry-count**: Number of times a MAC address can be unlocked after duplicate detection before it is frozen permanently.

When a MAC address is frozen, a syslog message notifies the user. While frozen, the router ignores new or updated MAC routes for that address. After the freeze-time expires, the MAC routes are unfrozen, and the move-count resets to zero. For unfrozen local MAC routes, the router initiates an ARP probe and flush, while remote MAC routes enter probe mode, restarting duplicate detection.

The router also tracks how many times a MAC address has been frozen and unfrozen. If a MAC address is marked duplicate after being unfrozen the configured retry-count times, it is frozen permanently. To clear permanently frozen hosts, you can:

- Shut down the host causing duplicate traffic.
- Use the **clear l2route evpn frozen-mac frozen-flag** command to clear frozen hosts.

This mechanism helps maintain network stability by managing MAC address mobility and preventing persistent duplicate MAC address issues.

How to prevent MAC address freezing

A MAC address is permanently frozen when it undergoes three duplicate detection and recovery events within a 24-hour period. Freezing disables the MAC address, potentially disrupting network connectivity. If duplicate detection events occur outside this 24-hour window, the count resets, and the MAC address is not permanently frozen.

Procedure

Step 1 Enable infinite duplicate detection.

To prevent permanent freezing, configure the MAC address to allow infinite duplicate detection and recovery cycles by entering this command:

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# mac secure retry-count infinity
```

This setting ensures the MAC address will not be frozen permanently despite repeated duplicate detection events.

Step 2 Configure reset interval for retry count (Optional).

By default, the router uses a 24-hour interval to track duplicate detection events. To customize this interval, use

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# mac secure reset-freeze-count-interval 30
```

This interval defines the period after which the retry count resets, preventing permanent freezing if events are spaced out.

The MAC address remains active and is not permanently frozen, allowing ongoing duplicate detection and recovery without network disruption.

What to do next

Monitor network behavior to verify MAC address stability and adjust the reset interval as needed to suit your environment.

Guidelines for managing host duplication and route advertisement

These guidelines ensure controlled handling of frequent host movements, preventing route flapping and promoting network stability.

- Duplication threshold: Allow up to five host movements (duplications) within a 180-second window before marking the host as a duplicate.
- Duplicate marking: When a host moves five times within 180 seconds, mark it as a duplicate for 30 seconds.
- Route advertisement suppression: During the 30-second duplicate period, suppress all route advertisements for the affected host.
- Duplicate status removal: After 30 seconds, remove the duplicate status to resume normal route advertisements.
- Permanent freeze: If a host is detected as a duplicate for the fourth time, permanently freeze the host and suppress all its route advertisements indefinitely.

Configure duplicate MAC address detection and prevent MAC address freezing

Detect duplicate MAC addresses moving between hosts and control MAC address freezing to maintain network stability.

Use this task to enable duplicate MAC address detection on EVPN routers and configure parameters that determine when a MAC address is marked as duplicate and how freezing is handled.

Procedure

Step 1 Enable duplicate MAC address detection on host MAC addresses.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# mac secure
```

Step 2 Set detection parameters to control duplicate MAC handling.

Example:

```
Router(config-evpn-mac-secure)# move-count 2
Router(config-evpn-mac-secure)# freeze-time 10
Router(config-evpn-mac-secure)# retry-count 2
Router(config-evpn-mac-secure)# commit
```

Step 3 To prevent MAC address freezing permanently, configure infinite retries.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# mac secure
Router(config-evpn-mac-secure)# retry-count infinite
Router(config-evpn-mac-secure)# commit
```

Step 4 Running configuration of duplicate MAC address detection and preventing MAC address freezing.

Example:

```
evpn
 mac secure
  move-count 2
  freeze-time 10
  retry-count 2
!
evpn
 mac secure
  retry-count infinite
 !
```

Step 5 Use the **show l2route evpn mac-ip 10.47.177.225 detail** command to verify duplicate MAC detection and freezing status.

In this example, the *0011.0000.0001* MAC address is identified as duplicate and subsequently frozen. The *DmZm* flag denotes that the MAC address has been marked as duplicate and frozen.

Example:

```
Router# show l2route evpn mac-ip 10.47.177.225 detail
Topo ID  Mac Address    IP Address      Producer    Next Hop(s)                                Seq No
   Flags
Opaque Data Type        Opaque Data Len
Opaque Data Value
Opaque NH Type          Opaque NH Len
Opaque NH Value
-------- -------------- --------------- ----------- ---------------------------------------- --------
 --------
161      0011.0000.0001 10.47.177.225   LOCAL       Bundle-Ether8.1212, N/A                  43
   BLDmZm
N/A                     N/A
N/A
N/A                     N/A
N/A
    Last Update: Fri Nov 03 17:42:09.426 CET
161      0011.0000.0001 10.47.177.225   L2VPN       25000/I/ME, N/A                          42
   DmZm
0                       12
0x06000000 0x3b010080 0x00000000
```

The router detects duplicate MAC addresses based on configured parameters and controls freezing behavior to prevent network disruption.

EVPN designated forwarder election

A designated forwarder election is a network control method that

- defines the backup path well before a link failure occurs
- enables the access network to manage EVPN provider edge (PE) devices proactively, and
- ensures that, during a link failure, the PE node knows the next PE to assume the active role, reducing traffic loss.

Table 26: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Designated Forwarder Election	Release 26.2.1	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 Designated Forwarder Election	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D

EVPN Designated Forwarder Election	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN Designated Forwarder Election	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The EVPN designated forwarder election functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN Designated Forwarder Election	Release 24.3.1	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*) * The EVPN designated forwarder election functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN Designated Forwarder Election	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) Designated Forwarder (DF) election enables the access network to control EVPN PE devices by defining the backup path much before the event of a link failure. During the link failure, the PE node is aware of the next PE that will take over the active role and this reduces the traffic loss. EVPN designated forwarder election supports preference-based and access-driven mechanism. * This feature is supported only on routers with the 88-LC1-36EH line cards.

DF election methods

You can configure the designated forwarder (DF) election using two methods: preference-based or access-driven.

- Preference-based DF election uses weight to determine which PE device acts as the DF at any time. This method suits topologies where interface failures are revertive.
- Access-driven DF election is recommended for topologies where an access PE connects directly to the core PE. When access PEs operate in non-revertive mode, this mechanism allows the access PE to select the DF.

EVPN designated forwarder election and backup path

In an access network, an interface connects PE nodes to the EVPN PE in the core network. When this interface fails, traffic loss can last longer because the backup PE is not preselected before the failure.

The EVPN DF election feature enables the EVPN PE to pre-program a backup PE before any interface failure occurs. This preselection allows the PE node to immediately know which PE will take over if the interface fails, significantly reducing convergence time.

To configure the backup path, use the preference DF weight option for the ESI. By assigning a weight to a PE, you control the DF election process and define the backup path accordingly.

This mechanism improves network resilience by minimizing traffic disruption during interface failures.

Restrictions for EVPN DF election

- This feature is supported only on EVPN PEs operating in port-active mode.
- The bundle attached to the Ethernet segment must be configured with the **lacp mode active** command.
- The **lacp mode on** command is not supported and must not be used.

How backup path precomputation works in EVPN

Summary

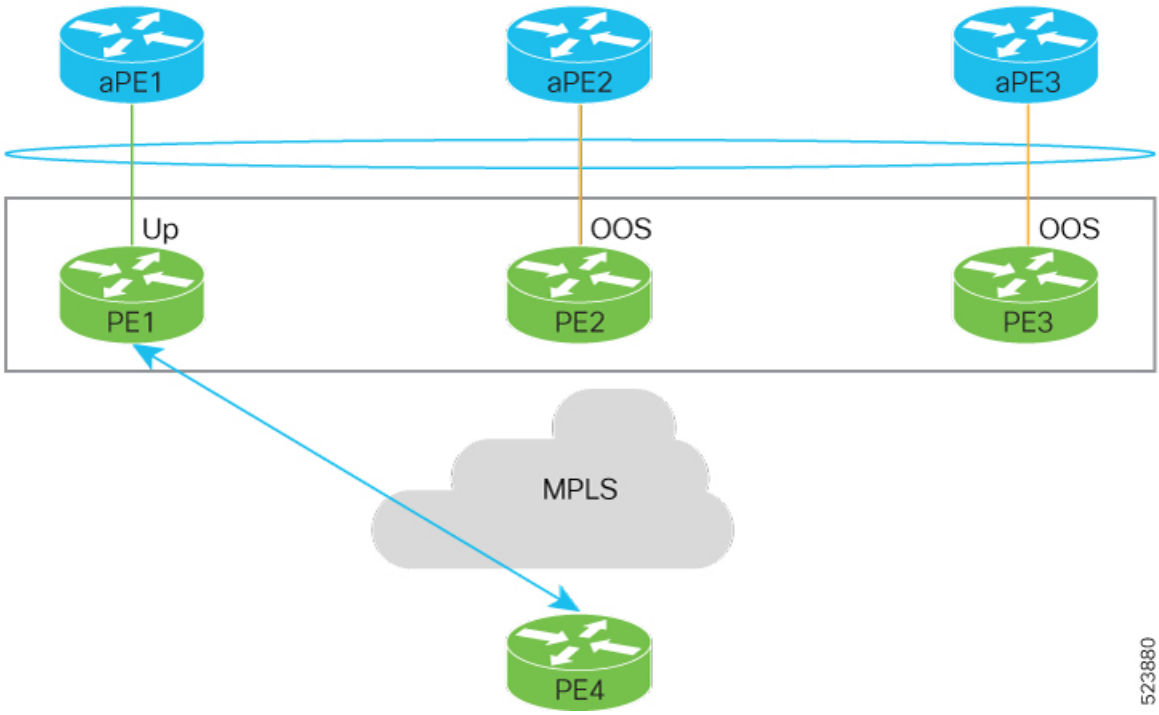
The key components involved in the backup path precomputation process in an EVPN network are:

- PE devices (PE1, PE2, PE3): Core PE routers configured with different weights to determine primary and backup paths.
- Access PEs (aPE1, aPE2, aPE3): Access PE routers configured in a multichassis link aggregation group (MCLAG) redundancy group operating in non-revertive mode.
- Ethernet segment: A bundle shared by all PE devices with a common Ethernet Segment Identifier (ESI).
- Traffic sources (for example, PE4): Hosts or devices sending traffic through the EVPN network.

The backup path precomputation process in an EVPN network involves core PE devices assigned different weights to determine primary and backup paths, access PE devices grouped in a multichassis link aggregation group operating in non-revertive mode, and a shared Ethernet segment identified by a common ESI. These components collaborate to provide redundancy and efficient path selection for traffic sources within the network.

Workflow

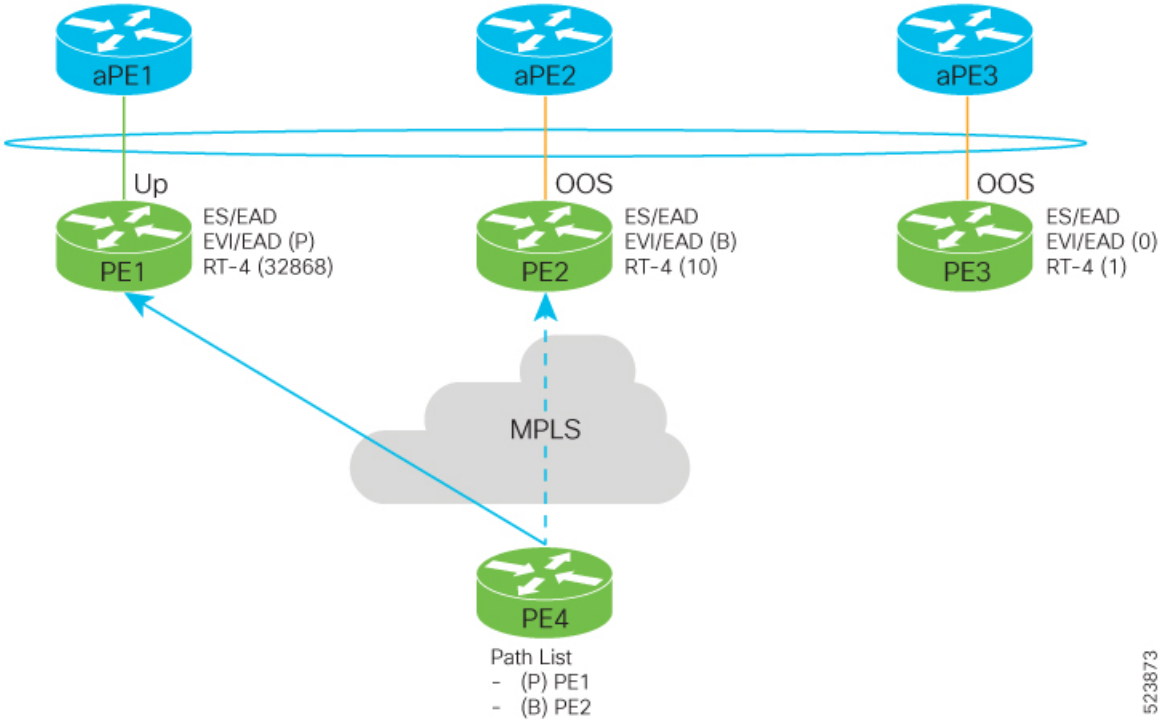
Figure 13: EVPN DF election



These are the stages of EVPN DF election.

1. Initial traffic flow.

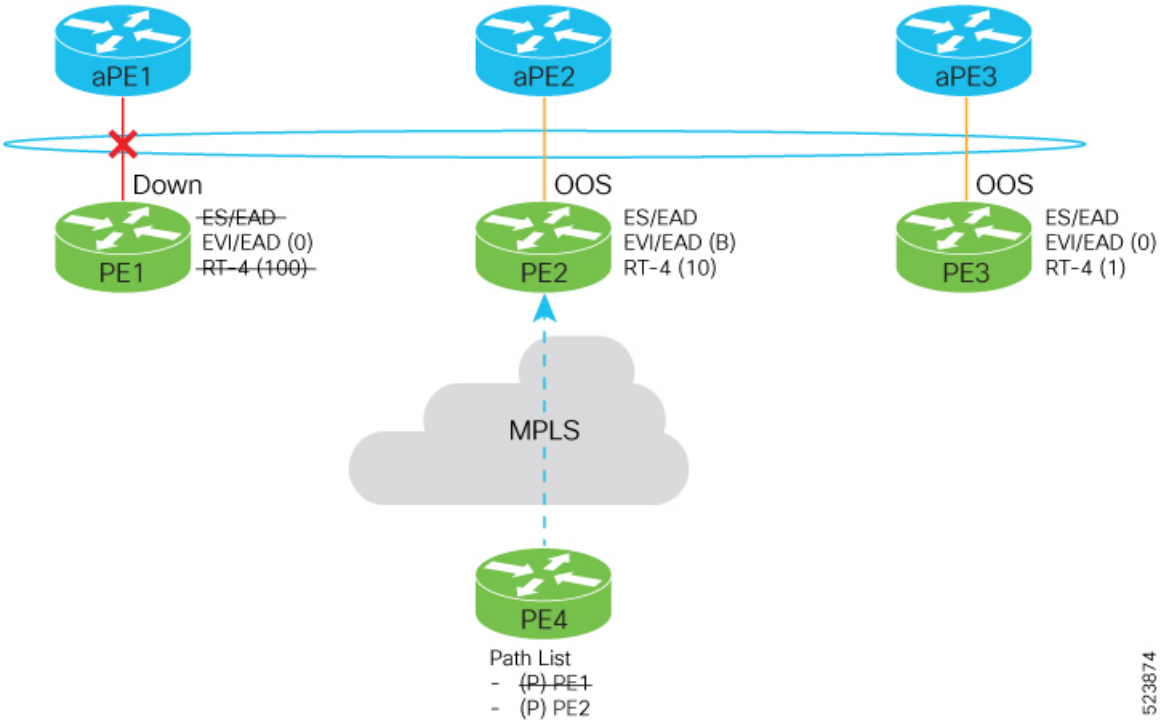
Figure 14: Traffic flow



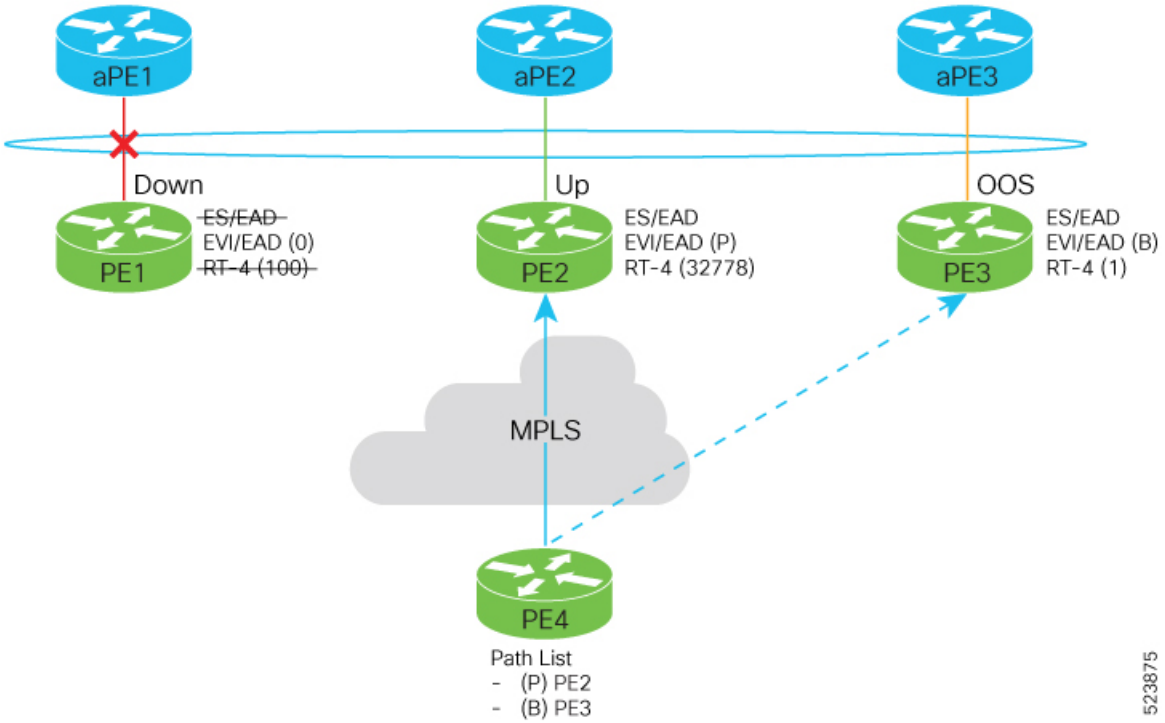
- The aPE1-PE1 interface is active (up), while aPE2-PE2 and aPE3-PE3 interfaces remain out of service (OOS).
- Traffic from PE4 is forwarded to aPE1 through PE1 because PE1 has the highest configured weight of 100.
- The highest weight is adjusted by adding 32,768 to the configured value (e.g., PE1's weight 100 becomes 32,868).
- The highest weight is advertised with the P-bit (primary), the next highest with the B-bit (backup), and the lowest weight as non-designated forwarder (NDF).
- When EVPN PE devices have the same weight, traffic forwarding preference is determined by the lowest IP address.
- Only one PE signals that the Ethernet Segment bundle state is up; all other PEs mark the Ethernet Segment as standby and their bundles as OOS.
- All PE devices maintain awareness of their peers' next hops and weights.

2. Failure and recovery scenarios.
   - The weights configured on EVPN PE devices follow the same order as the protection mechanism on the access side PEs: aPE1, aPE2, then aPE3.
   - The weight hierarchy is PE1 > PE2 > PE3.
   - If this ordering is not maintained, the network will eventually converge, but efficiency will be reduced.

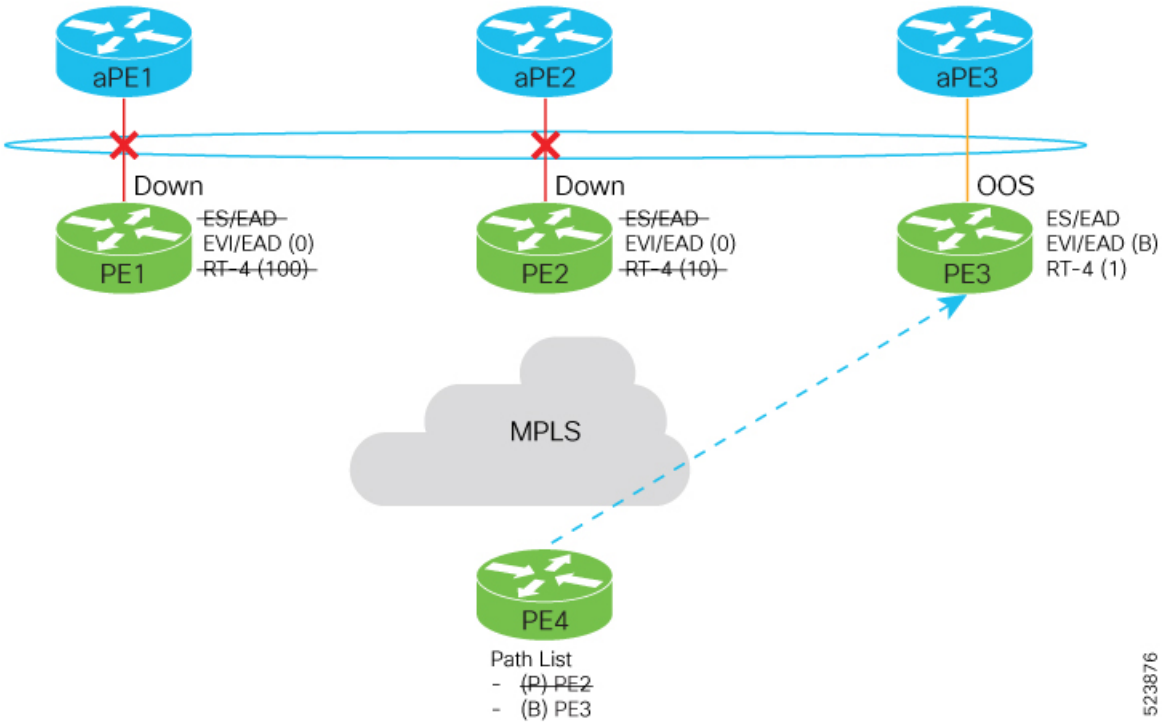
3. Scenario 1 – aPE1-PE1 interface down



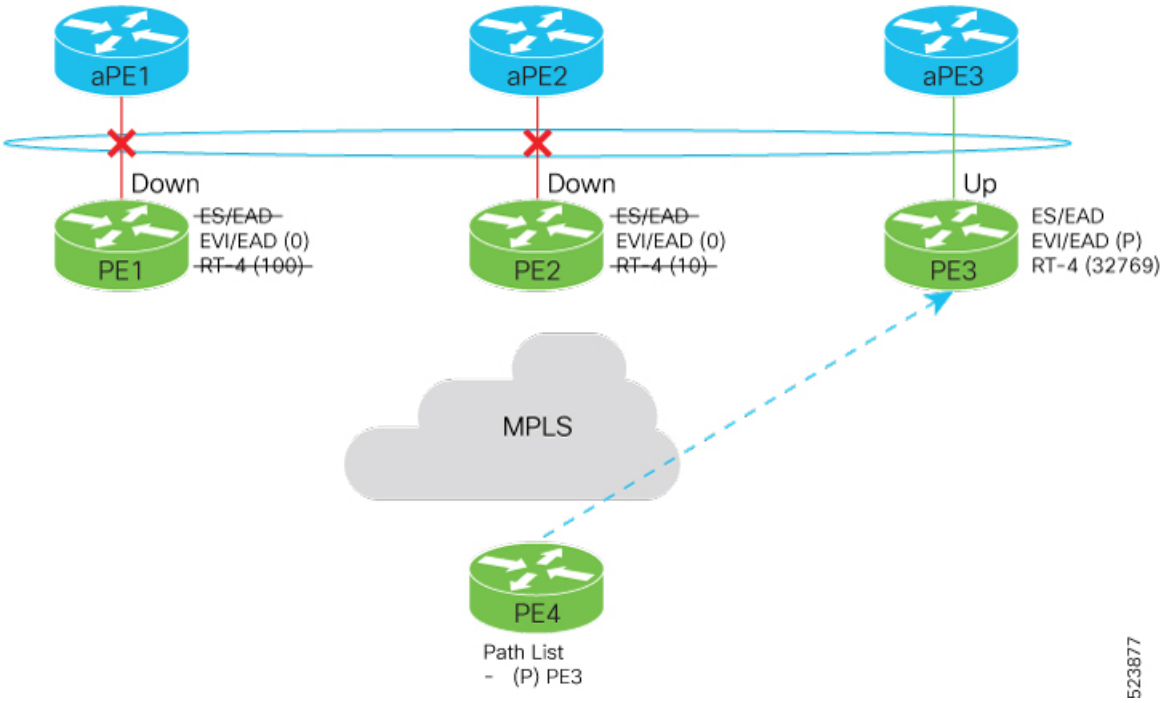
- PE1 withdraws the EAD/ES route.
- Traffic is sent through the backup path through PE2.
- aPE2-PE2 becomes primary with a weight of 32,778 and advertises the P-bit.
- aPE3-PE3 becomes backup and advertises the B-bit.



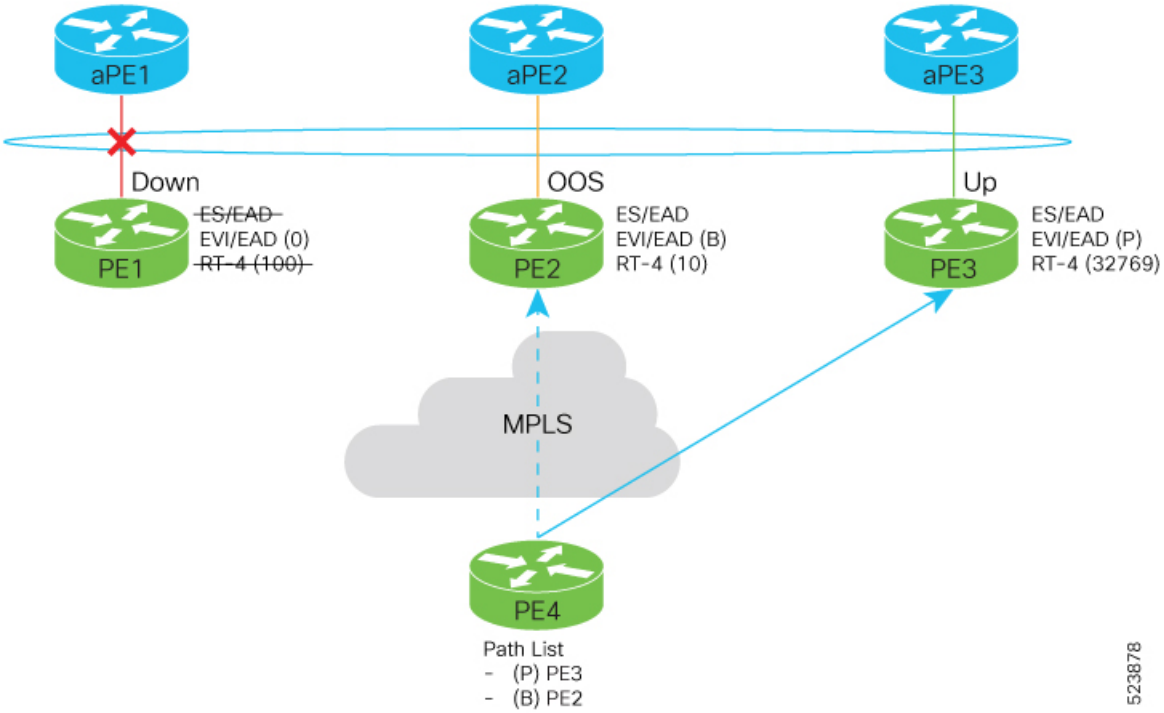
4. Scenario 2 – aPE2-PE2 interface also down



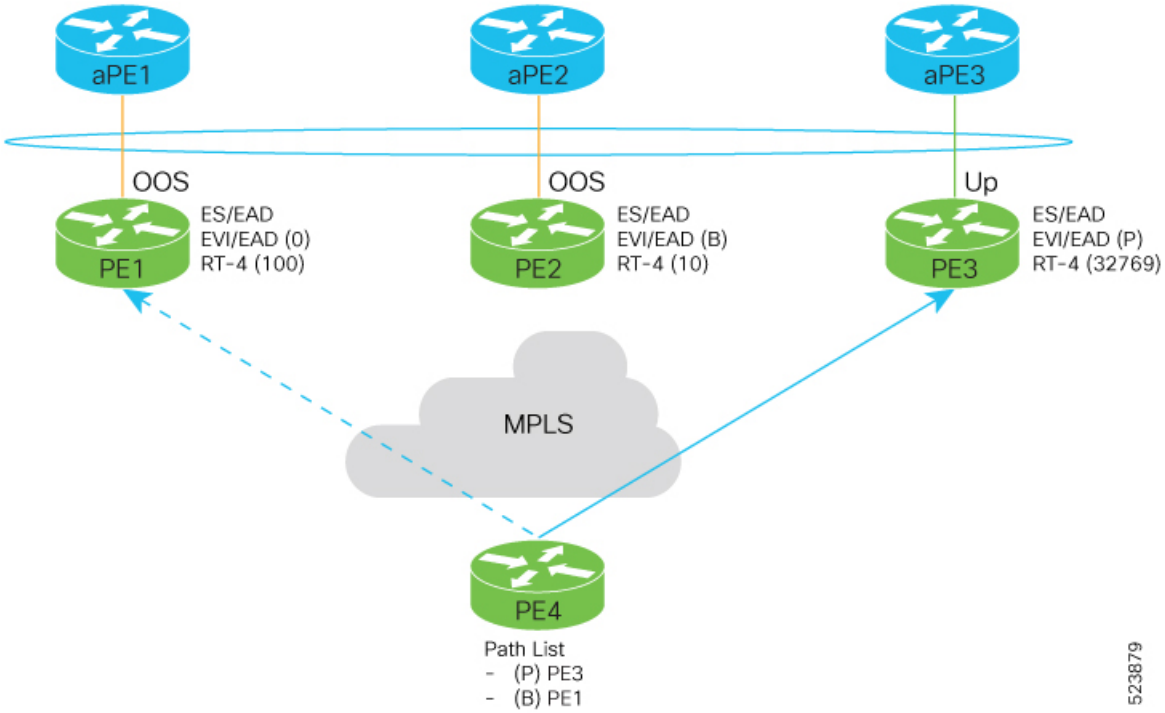
- Traffic is sent through aPE3-PE3 link.
- aPE3-PE3 becomes the primary path with a weight of 32,769.



5. Scenario 3 – aPE2-PE2 interface recovers
   - aPE3-PE3 remains the primary path.
   - aPE2-PE2 becomes the backup path with a weight of 10.



6. Scenario 4 – aPE1-PE1 interface recovers
   - aPE3-PE3 remains the primary path with a weight of 32,769.
   - aPE1-PE1 becomes the backup path with a weight of 100.
   - aPE2-PE2 becomes non-designated forwarder (NDF) with a weight of 10.



Result

This process ensures efficient traffic forwarding and redundancy in the EVPN network by precomputing backup paths based on configured weights and interface states, enabling fast failover and recovery while maintaining optimal traffic flow.

Configure EVPN DF election

Configure EVPN DF election using preference-based and access-driven service carving modes, and enable LACP with non-revertive mode on associated aPE devices.

Use this task to set up stable and efficient DF election across multiple PE devices with differentiated weights and access-driven behavior, while supporting link aggregation on aPE devices.

Before you begin

- Ensure all PE devices are reachable and have consistent Ethernet Segment Identifiers (ESI).
- Confirm LACP support on bundle interfaces.

Perform these tasks to configure access-driven and preference-based EVPN DF election:

- Configure EVPN DF election on PE1, PE2, and PE3, with the service carving mode as preference-based and access-driven.
- Configure LACP on aPE1, aPE2, and aPE3

Procedure

Step 1 Configure EVPN DF election on PE1, PE2, and PE3

Example:

on PE1

```
Router#configure
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 100
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit
```

Example:

on PE2

Example:

```
Router#configure
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 10
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit
```

Example:

on PE3

Example:

```
Router#configure
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 1
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit
```

Step 2 Configure LACP on aPE1, aPE2, and aPE3.

Example:

on aPE1

Example:

```
Router#configure
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/40
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 10000
Router(config-if)description Connection to PE1
Router(config-if)commit
```

Example:

on aPE2

```
Router#configure
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/39
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 20000
Router(config-if)description Connection to PE2
Router(config-if)commit
```

Example:

on aPE2

```
Router#configure
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/38
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 30000
Router(config-if)description Connection to PE3
Router(config-if)commit
```

Step 3 Running configuration of EVPN DF election.

Example:

```
/* PE1 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 100
    access-driven
   !
 !

/* PE2 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
```

```
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 10
    access-driven
   !
 !

/* PE3 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 1
    access-driven
   !
 !

/* aPE1 Configuration */

interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/40
 bundle id 10 mode active
 bundle port-priority 10000
 description Connection to PE1
!

/* aPE2 Configuration */

interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/39
 bundle id 10 mode active
 bundle port-priority 20000
 description Connection to PE2
!

/* aPE3 Configuration */

interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/40
 bundle id 10 mode active
 bundle port-priority 30000
 description Connection to PE3
!
```

Step 4 Use the **show evpn ethernet-segment detail** to verify that you have successfully configured EVPN DF election.

Example:

```
Router#show evpn ethernet-segment detail
Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0001.0001.0001.1b01.001b BE1                               192.168.0.1
                                                           192.168.0.3
  ES to BGP Gates   : Ready
```

```
ES to L2FIB Gates : Ready
Main port         :
   Interface name : Bundle-Ether1
   Interface MAC  : 02ef.af8d.8008
   IfHandle       : 0x00004190
   State          : Up
   Redundancy     : Active
ESI type          : 0
   Value          : 01.0001.0001.1b01.001b
ES Import RT      : 0100.0100.011b (from ESI)
Source MAC        : 0000.0000.0000 (N/A)
Topology          :
   Operational    : MH
   Configured     : Port-Active
Service Carving   : Preferential
   Multicast      : Disabled
Convergence       :
Peering Details   : 2 Nexthops
   192.168.0.1 [PREF:P:d6ce:T] >> Weight in hexadecimal
   192.168.0.3 [PREF:P:457]
Service Carving Synchronization:
   Mode           : NONE
   Peer Updates   :
Service Carving Results:
   Forwarders     : 3
   Elected        : 3
   Not Elected    : 0
EVPN-VPWS Service Carving Results:
   Primary        : 1
   Backup         : 0
   Non-DF         : 0
MAC Flushing mode : STP-TCN
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Local SHG label   : 28384
Remote SHG labels : 0
Access signal mode: Bundle OOS (Default)
```

Highest random weight mode for EVPN DF election

A Highest Random Weight (HRW) mode for EVPN DF election is a capability that

- provides optimal load distribution for DF election
- ensures redundancy, enables fast access, and
- guarantees nondisruptive service for an Ethernet Segment (ES) regardless of the state of a peer DF.

DF election weight calculation

The DF election is determined by calculating weights for each router on the Ethernet segment. The router with the highest weight is selected as the DF, while the router with the next highest weight becomes the backup designated forwarder (BDF).

The weight is computed using the following formula:

$$Wrand(v, Si) = (1103515245 \times ((1103515245 \times Si + 12345) \text{ XOR } D(v)) + 12345) \bmod 2^{31}$$

where:

- Si: IP address of router *i*
- v: Ethernet Virtual Instance (EVI)
- D(v): 31-bit digest, specifically the CRC-32 checksum of *v*

This calculation ensures a unique and deterministic weight assignment based on the router's IP address and the EVI, facilitating the election of the DF and BDF roles.

Designated forwarder election algorithm

The DF election algorithm determines which PE device forwards traffic for a multihomed Ethernet segment. It uses a mathematical function called "service carving," which combines the ESI and EVI, as described in RFC 7432.

Key points about the algorithm:

- The election is based on an ordinal value derived from a modulus calculation involving the number of peers and the EVI.
- This modulus calculation mode can perform poorly when Ethernet tags are all even or all odd.
- In scenarios where an Ethernet Segment (ES) is multihomed to two PEs, all VLANs may select the same PE as the DF.
- Consequently, one PE may never be elected as the DF, resulting in a non-optimal distribution of forwarding responsibilities.

Benefits of highest random weight mode over modulus mode in DF election

The HRW mode of DF election offers several advantages compared to the modulus mode:

- Equal distribution of DF election across PEs: The DF election for each VLAN is evenly distributed among the PEs, ensuring balanced load sharing.
- Stability during PE failures: If a PE that is neither a DF nor a BDF hosting VLANs on a given ES goes down or loses connection to the ES, it does not trigger a reassignment of DF and BDF roles to other PEs. This behavior reduces unnecessary computation during connection flaps.
- Minimized service disruption: HRW mode avoids the service interruptions that are common with the modulus-based DF election algorithm.
- Redundant connectivity through BDF: The BDF provides backup connectivity, ensuring no traffic disruption occurs when a DF fails. In such cases, the BDF automatically assumes the DF role.

Configure highest random weight mode for EVPN DF election

Enable the HRW mode for DF election in EVPN to optimize forwarding decisions based on a hashing algorithm, improving load distribution and redundancy.

Procedure

Step 1 Configure HRW mode for EVPN DF election.

Example:

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

Step 2 Running configuration of HRW mode for DF election.

Example:

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

Step 3 Use the **show evpn ethernet-segment interface bundleEther 23 carving detail** command to verify that you have configured HRW mode of DF election.

Example:

```
Router#show evpn ethernet-segment interface bundleEther 23 carving detail
Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0011.1111.1111.1111.1111 Gi0/2/0/0                         192.168.0.2
                                                           192.168.0.3
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : GigabitEthernet0/2/0/0
     Interface MAC  : 02db.c740.ca4e
     IfHandle       : 0x01000060
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 11.1111.1111.1111.1111
  ES Import RT      : 0011.0011.0011 (Local)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, Single-active
     Configured     : Single-active (AApS) (default)
  Service Carving   : HRW    -> Operation mode of carving
  Peering Details   : 192.168.0.2[HRW:P:00] 192.168.0.3[HRW:P:00] -> Carving capability as advertised
 by peers
  Service Carving Results:
     Forwarders     : 1
     Permanent      : 0
     Elected        : 0
     Not Elected    : 1
```

```
MAC Flushing mode : STP-TCN
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Local SHG label   : 28109
Remote SHG labels : 1
            24016 : nexthop 192.168.0.3
```

EVPN non-revertive DF election

A non-revertive mode of DF election is a network configuration mode that

- adjusts the weight of PE devices so that the PE that became the designated forwarder during a link failure remains the designated forwarder after the link recovers
- prevents traffic disruption during DF re-election and service re-carving, and
- avoids triggering service re-carving after recovery.

Table 27: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Non-Revertive Designated Forwarder (DF) Election	Release 26.2.1	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 Non-Revertive Designated Forwarder (DF) Election	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN Non-Revertive Designated Forwarder (DF) Election	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN Non-Revertive Designated Forwarder (DF) Election	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The EVPN Non-Revertive Designated Forwarder Election functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
EVPN Non-Revertive Designated Forwarder (DF) Election	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The EVPN Non-Revertive Designated Forwarder Election functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN Non-Revertive Designated Forwarder (DF) Election	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) In a preference-based Designated Forwarder (DF) election, non-revertive mode prevents the traffic disruption that occurs during the recovery of a node in a port-active multihoming network. While recovering from a link failure, an EVPN ethernet-segment (ES) performs DF re-election and re-carves the services among the multihomed nodes, which causes traffic interruption and interface flapping, leading to traffic loss. In the non-revertive mode, the EVPN ES does not re-carve the services after the recovery, thus avoiding the traffic disruption. The feature introduces these changes: **CLI**: • **non-revertive** • **revert** • The **ethernet-segment interface** *interface-name* **revert** keyword is introduced in the **l2vpn evpn** command. **YANG Data Model**: • `Cisco-IOS-XR-evpn-oper.yang` • `Cisco-IOS-XR-l2vpn-cfg.yang` (see GitHub, YANG Data Models Navigator) *This feature is supported only on routers with the 88-LC1-36EH line cards.

Non-revertive mode to minimize traffic interruption during DF re-election

During link recovery, the re-election of the DF and the re-carving of services cause traffic interruptions and loss. This problem worsens when an Ethernet Segment hosts multiple services because the time required for service re-carving and transferring the DF role to the Provider Edge (PE) with the highest weight prolongs traffic disruption.

The non-revertive mode addresses this issue by preventing the DF role from reverting to the original PE with the highest weight after failure recovery. Instead, it maintains the current DF, which reduces service interruptions and traffic loss during recovery. This approach enhances network stability and ensures better service continuity.

Key points:

- DF election is preference-based, selecting the PE with the highest weight as DF.
- Link failure triggers DF as DF.
- Link failure triggers DF re-election, causing traffic interruption.
- Recovery triggers another re-election and service re-carving, increasing the risk of traffic loss.
- Non-revertive mode avoids reverting the DF role to the original PE after recovery.
- This mode reduces traffic interruption and improves service continuity during failover and recovery.

To return to revertive mode, the network resumes the DF election and service carving processes. This transition allows the DF role to revert to the PE with the highest preference, restoring the default behavior.

- Using the revert timer: Configure a timer with the **revert** command that starts during node recovery. When this timer expires, revertive mode activates, triggering the DF election. This delay helps prevent immediate traffic disruption by postponing the election, allowing the PE with the highest preference to become the DF smoothly.
- Disabling non-revertive mode: Use the **l2vpn evpn ethernet-segment interface revert** command to disable non-revertive mode. This action immediately triggers DF election and service carving. If a revert timer was previously set, it is canceled upon disabling non-revertive mode.

Restrictions for EVPN non-revertive DF election

Use non-reverting mode of EVPN DF election only in these scenarios:

- When performing preference-based DF election.
- On physical and bundle interfaces.
- In EVPN port-active multihoming mode.

Do not use non-reverting mode of EVPN DF election in these scenarios:

- When using access-driven DF election.
- On virtual interfaces such as virtual Ethernet segment (vES), network virtualization endpoint (NVE), and pseudowire headend (PWHE).
- When employing segment routing over IPv6 (SRv6).

Configure EVPN non-revertive DF election

Enable non-revertive DF election to prevent traffic disruption during link recovery in EVPN.

In EVPN networks, the DF election determines which PE device forwards traffic for a multi-homed Ethernet segment. By default, DF election reverts to the highest-weight PE after link recovery, which can cause traffic disruption. Configuring non-revertive mode maintains the current DF to avoid this disruption.

Before you begin

It is recommended to configure the non-revertive mode of DF election on all the nodes in the network.

1. Configure Ethernet segment in port-active load-balancing mode on peering PEs for a specific interface, using the **load-balancing-mode port-active** command.
2. Configure the service carving mode as preference-based using the **service-carving preference-based** command. The DF election happens based on the highest preference, that is the weight of the PE.
3. Configure the non-revertive mode of DF election using the **non-revertive** command, to enable the non-revertive mode on the PEs.
4. Configure the PE devices with different weights, using the **weight** command.

Procedure

Step 1 Configure non-revertive mode.

In this example, PE1 and PE2 are configured with a weight of 100 and 10 respectively.

- After the DF election, PE1 is selected as the DF.
- When there is a link failure, PE1 goes down, and the next PE with the highest weight, PE2, becomes the DF.
- By default, the DF election happens during the recovery, and PEl becomes the DF again. Transferring the DF role from PE2 to PE1 leads to traffic disruption.
- When the non-revertive mode is enabled, the weight of the PE1 is adjusted so that PE2 remains the DF. This prevents the traffic disruption incurred due to the DF election.

a) Configure non-revertive mode on PE1.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode port-active
Router(config-evpn-ac-es)# service-carving preference-based
Router(config-evpn-ac-es-sc-pref)# non-revertive
Router(config-evpn-ac-es-sc-pref)# weight 100
Router(config-evpn-ac-es-sc-pref)# commit
```

b) Configure non-revertive mode on PE2.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode port-active
Router(config-evpn-ac-es)# service-carving preference-based
Router(config-evpn-ac-es-sc-pref)# non-revertive
Router(config-evpn-ac-es-sc-pref)# weight 10
Router(config-evpn-ac-es-sc-pref)# commit
```

Step 2 Running configuration of EVPN non-revertive mode DF election.

Example:

```
evpn
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
   load-balancing-mode port-active
   service-carving preference-based
    non-revertive
    weight 100
   !
evpn
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
   load-balancing-mode port-active
   service-carving preference-based
    non-revertive
    weight 10
```

Step 3 Use the **show evpn ethernet-segment interface Bundle-Ether 1 private** command to verify that non-revertive mode is enabled.

Example:

```
Router# show evpn ethernet-segment interface Bundle-Ether 1 private

…
  Topology          :
     Operational    : SH
     Configured     : Port-Active
  Service Carving   : Preferential
     Config Weight  : 100
     Oper Weight    : 100
     Non-Revertive  : Enabled, Active
     Access Driven  : Disabled
     Multicast      : Disabled
  Convergence       :
  Peering Details   : 2 Nexthops
     192.168.0.1 [PREF:DP:7fff:T][1]
     192.168.0.3 [PREF:DP:7fff:T][2]
```

Configure to return to revertive mode

Restore the default revertive mode behavior in EVPN after a link failure recovery.

In non-revertive mode, the DF election does not occur during recovery from a link failure. To revert to the default behavior where DF election happens, configure the revert timer.

Procedure

Step 1 Configure non-revertive mode on an interface and configure revert timer on the interface.

When you configure a revert timer on the PEs enabled with non-revertive mode, the timer starts after the nodes have recovered from link failure. After the timer expires, the PEs return to the revertive mode and DF election happens in the network. The timer is configured in seconds.

Example:

```
outer# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode port-active
Router(config-evpn-ac-es)# service-carving preference-based
Router(config-evpn-ac-es-sc-pref)# non-revertive
Router(config-evpn-ac-es-sc-pref)# weight 100
Router(config-evpn-ac-es-sc-pref)# exit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# timers
Router(config-evpn-ac-timers)# revert 300
Router(config-evpn-ac-es)# commit
```

You can also configure the revert timer globally.

```
Router(config)# evpn
Router(config-evpn)# timers
Router(config-evpn-timers)# revert 300
Router(config-evpn-timers)# commit
```

Step 2 Running configuration of return to revertive mode.

Example:

Revert timer configuration on an interface.

```
evpn
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
   load-balancing-mode port-active
   service-carving preference-based
    non-revertive
  !
   timers
    revert 300
```

Global configuration of revert timer.

```
evpn
  timers
   revert 300
```

Step 3 Use the **show evpn ethernet-segment interface Bundle-Ether 1 private** command to verify that non-revertive mode is enabled along with the configured revert timer.

Example:

```
Router# show evpn ethernet-segment interface Bundle-Ether 1 private

…
 Topology          :
    Operational    : SH
    Configured     : Port-Active
 Service Carving   : Preferential
    Config Weight  : 100
    Oper Weight    : 100
    Non-Revertive  : Enabled, Active
    Access Driven  : Disabled
    SRG Driven     : Disabled
    Multicast      : Disabled
 Convergence       :
 Peering Details   : 0 Nexthops
 Service Carving Synchronization:
    Mode           : NONE
    Peer Updates   :
 Service Carving Results:
    Forwarders     : 0
    Elected        : 0
    Not Elected    : 0
 EVPN-VPWS Service Carving Results:
    Primary        : 0
    Backup         : 0
    Non-DF         : 0
 MAC Flush msg     : STP-TCN
 Peering timer     : 3 sec [not running]
 Recovery timer    : 30 sec [not running]
 Carving timer     : 0 sec [not running]
 Revert timer      : 300 sec [not running]
 HRW Reset timer   : 5 sec [not running]
 AC Debounce timer : 3000 msec [not running]
```

In this example, the revert timer has expired and the non-revertive mode is inactive.

```
Router# show evpn ethernet-segment interface Bundle-Ether 1 private

…
 Topology          :
    Operational    : SH
    Configured     : Port-Active
 Service Carving   : Preferential
    Config Weight  : 100
    Oper Weight    : 100
    Non-Revertive  : Enabled, Inactive
    Access Driven  : Disabled
    SRG Driven     : Disabled
    Multicast      : Disabled
 Convergence       :
 Peering Details   : 0 Nexthops
 Service Carving Synchronization:
    Mode           : NONE
    Peer Updates   :
 Service Carving Results:
    Forwarders     : 0
    Elected        : 0
    Not Elected    : 0
 EVPN-VPWS Service Carving Results:
    Primary        : 0
```

```
   Backup         : 0
   Non-DF         : 0
MAC Flush msg     : STP-TCN
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Revert timer      : 0 sec [not running]
HRW Reset timer   : 5 sec [not running]
AC Debounce timer : 3000 msec [not running]
```

The PE devices return to revertive mode after the revert timer expires, restoring the default DF election behavior during link recovery.

Disable non-revertive mode

Disable the non-revertive behavior on an Ethernet segment interface to cancel the revert timer and trigger a new DF election in the network.

Use this task when you need to stop the non-revertive mode on an Ethernet segment interface, causing the network to perform a fresh DF election.

Procedure

Disable non-revertive mode on the specified Ethernet segment interface.

Example:

```
Router# l2vpn evpn ethernet-segment interface Bundle-Ether1 revert
```

This command cancels the revert timer if it is configured and initiates a new DF election in the network.

The non-revertive mode is disabled, the revert timer is cancelled, and the network performs a new DF election.

CHAPTER 8

Enhanced EVPN Services and Failover Techniques

EVPN multiple services per Ethernet segment

An Ethernet segment is a network infrastructure component that

- supports multiple services on the same physical hardware resource
- provides traffic segregation among these services, and
- enables users to manage traffic configurations effectively.

Table 28: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Multiple Services per Ethernet Segment	Release 26.2.1	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 Multiple Services per Ethernet Segment	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.

Feature Name	Release Information	Feature Description
EVPN Multiple Services per Ethernet Segment	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN Multiple Services per Ethernet Segment	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN Multiple Services per Ethernet Segment	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: P100])(select variants only*) *The EVPN multiple services per Ethernet segment functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN Multiple Services per Ethernet Segment	Release 24.3.1	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*) *The EVPN multiple services per Ethernet segment functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN Multiple Services per Ethernet Segment	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: Q200, P100]) (select variants only*) You can configure EVPN to run multiple services on a single Ethernet Segment (ES), which enables the efficient use of network resources. While the services run on the same physical hardware resource, each service can be associated with a different EVPN instance and separated from each other. This allows traffic segregation, which enables users to employ their own traffic management configurations. *This feature is supported only on routers with the Q200 and 88-LC1-36EH line cards.

Highlights and benefits of EVPN multiple services per Ethernet segment

- Enables consolidation of diverse services on a shared Ethernet Segment without compromising service isolation.
- Supports independent traffic policies and configurations for each service, enhancing operational control.
- Facilitates efficient use of physical infrastructure by allowing multiple services to coexist on the same hardware.
- Improves network scalability and flexibility by reducing the need for separate physical segments.
- Simplifies maintenance and upgrades by centralizing service management on a single Ethernet segment.

These capabilities help network operators optimize resource utilization while maintaining clear separation and control of service traffic, leading to streamlined operations and reduced costs.

Services supported on a single Ethernet bundle

You can configure multiple services on a single Ethernet bundle, with one service assigned to each sub-interface. The supported services include:

- EVPN E-Line xconnect service
- Native EVPN E-LAN service

These services are supported only on all-active multihoming mode.

Configure EVPN multiple services per Ethernet segment

Configure multiple EVPN services on bundle-Ethernet sub-interfaces to support diverse customer services over a single Ethernet segment.

Consider a CE device connected to two PE devices through bundle-Ethernet interface 22001. Configure multiple services on bundle Ethernet sub-interfaces.

Procedure

Step 1 Configure attachment circuits.

Consider bundle-Ether22001 ES, and configure multiple services on sub-interface.

Example:

```
Router# configure
Router(config)# interface Bundle-Ether22001.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 12
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.13 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 13
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.14 l2transport
```

```
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 14
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 1
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.2 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 2
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.3 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 3
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.4 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 4
Router(config-l2vpn-subif)# commit
```

Step 2 Configure EVPN E-Line xconnect service.

Example:

```
Router# configure
Router(config)# interface Bundle-Ether22001.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 11
Router(config-l2vpn-subif)# rewrite ingress tag pop 2 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router# configure
Route(config)# l2vpn
Router(config-l2vpn)# xconnect group xg22001
Router(config-l2vpn-xc)# p2p evpn-vpws-mclag-22001
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether22001.11
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 22101 target 220101 source 220301
Router(config-l2vpn-xc-p2p-pw)# commit
```

Step 3 Configure native EVPN.

Example:

```
Router # configure
Router (config)# evpn
Router (config-evpn)# interface Bundle-Ether22001
Router (config-evpn-ac)# ethernet-segment identifier type 0 ff.ff.ff.ff.ff.ff.ff.ff.ee
Router (config-evpn-ac-es)# bgp route-target 2200.0001.0001
Router (config-evpn-ac-es)# exit
Router (config-evpn)# evi 24001
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64:24001
Router (config-evpn-evi-bgp)# route-target export 64:24001
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 21006
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target 64:10000
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22101
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64:22101
Router (config-evpn-evi-bgp)# route-target export 64:22101
Router (config-evpn-evi-bgp)# exit
```

```
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22021
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64: 22021
Router (config-evpn-evi-bgp)# route-target export 64: 22021
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn-evi)# advertise-mac
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22022
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64: 22022
Router (config-evpn-evi-bgp)# route-target export 64: 22022
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# advertise-mac
Router (config-evpn-evi)# commit
```

Step 4 Running configuration of EVPN multiple services per Ethernet segment.

Example:

```
/* Configure attachment circuits */
interface Bundle-Ether22001.12 l2transport
encapsulation dot1q 1 second-dot1q 12
!
interface Bundle-Ether22001.13 l2transport
encapsulation dot1q 1 second-dot1q 13
!
interface Bundle-Ether22001.14 l2transport
encapsulation dot1q 1 second-dot1q 14
!
interface Bundle-Ether22001.1 l2transport
 encapsulation dot1q 1 second-dot1q 1
!
interface Bundle-Ether22001.2 l2transport
encapsulation dot1q 1 second-dot1q 2
!
interface Bundle-Ether22001.3 l2transport
encapsulation dot1q 1 second-dot1q 3
!
interface Bundle-Ether22001.4 l2transport
encapsulation dot1q 1 second-dot1q 4


/* Configure EVPN E-Line xconnect service */
interface Bundle-Ether22001.11 l2transport
  encapsulation dot1q 1 second-dot1q 11
  rewrite ingress tag pop 2 symmetric
 !
l2vpn
xconnect group xg22001
p2p evpn-vpws-mclag-22001
 interface Bundle-Ether22001.11
 neighbor evpn evi 22101 target 220101 source 220301
!
  !
/* Configure Native EVPN */
Evpn
 interface Bundle-Ether22001
  ethernet-segment   identifier type 0 ff.ff.ff.ff.ff.ff.ff.ff.ee
  bgp route-target 2200.0001.0001
  !
  evi 24001
```

```
 bgp
  route-target import 64:24001
  route-target export 64:24001
 !
 evi 21006
  bgp
    route-target 64:100006
 !
  evi 22101
   bgp
     route-target import 64:22101
     route-target export 64:22101
   !
 evi 22021
  bgp
    route-target import 64:22021
    route-target export 64:22021
  !
  advertise-mac
!
evi 22022
 bgp
  route-target import 64:22022
  route-target export 64:22022
 !
  advertise-mac
!
```

Step 5 Use **show l2vpn xconnect summary** and **show l2vpn xconnect group xg22001 xc-name evpn-vpws-mclag-22001** commands to verify if each of the services is configured on the sub-interface.

Example:

```
Router# show l2vpn xconnect summary
Number of groups: 6
Number of xconnects: 505  Up: 505  Down: 0  Unresolved: 0 Partially-programmed: 0
AC-PW: 505  AC-AC: 0  PW-PW: 0 Monitor-Session-PW: 0
Number of Admin Down segments: 0
Number of MP2MP xconnects: 0
 Up 0 Down 0
Advertised: 0 Non-Advertised: 0


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

Hierarchical EVPN access pseudowire

A hierarchical EVPN access pseudowire is a network capability that

- reduces the number of pseudowires between network provider edge (N-PE) devices

- connects user provider edge (U-PE) devices to N-PE devices using EVPN access pseudowires for each VPN instance, and
- links customer edge (CE) devices to U-PE devices through attachment circuits.

This capability optimizes network scalability by minimizing the pseudowire count on the provider edge, simplifying management and improving efficiency.

- Pseudowire (PW): A virtual point-to-point connection that emulates a physical wire over a packet-switched network.
- Network Provider Edge (N-PE): The provider's edge device that terminates pseudowires.
- User Provider Edge (U-PE): The device connecting customer edge devices to the provider network through pseudowires.
- Attachment Circuit: The physical or logical link connecting a CE device to a U-PE device.

Table 29: Feature History Table

Feature Name	Release Information	Feature Description
Hierarchical EVPN access pseudowire	Release 26.2.1	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.
Hierarchical EVPN access pseudowire	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Hierarchical EVPN access pseudowire	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Hierarchical EVPN access pseudowire	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
Hierarchical EVPN access pseudowire	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The Hierarchical EVPN Access Pseudowire functionality is now extended to the Cisco 8712-MOD-M routers.

Hierarchical EVPN access pseudowire	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The Hierarchical EVPN Access Pseudowire functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Hierarchical EVPN access pseudowire	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) This feature enables you to configure EVPN VPWS in the access node under the same bridge domain as EVPN in the core and helps to build a PW to the nearest high-end PE that stitches those access circuits using EVPN. Therefore, the access nodes can leverage the benefits of EVPN. This feature also allows you to reduce the number of pseudowires (PWs) between the network provider edge (N-PE) devices by replacing PE devices with user provider edge (U-PE) and network provider edge (N-PE) devices. This feature prevents signaling overhead and packet replication. *This feature is supported only on routers with 88-LC1-36EH line cards.

Hierarchical EVPN access pseudowire model

The hierarchical EVPN access pseudowire is a network feature that reduces the number of PWs between N-PE devices. This capability is accomplished by introducing a two-tier provider edge architecture where:

- U-PE devices connect to CE devices through attachment circuits and establish EVPN access pseudowires for each VPN instance to the N-PE devices.
- N-PE devices communicate with other N-PE devices in the core network, handling the aggregation of pseudowires from multiple U-PE devices.

How hierarchical EVPN access pseudowire works

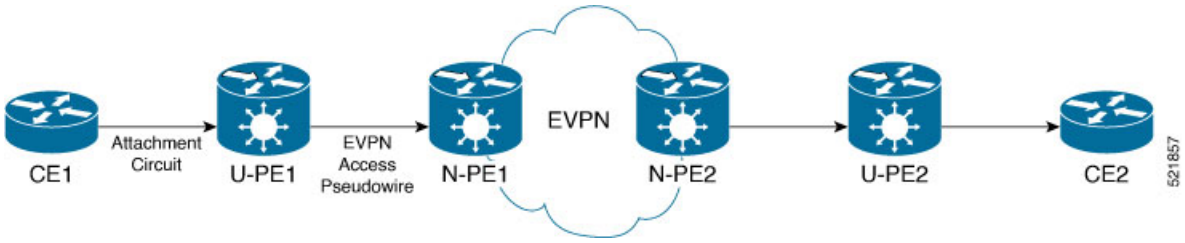
Summary

The key components involved in the hierarchical EVPN access pseudowire are:

- U-PE1: Connects to the CE1 device through an attachment circuit and transports CE1 traffic over an EVPN access PW.
- N-PE1: Receives the access PW from U-PE1 and connects to other N-PE devices (such as N-PE2) within the EVPN core.
- N-PE2: Part of the EVPN core, interconnected with N-PE1.
- AC: The physical or logical link between CE1 and U-PE1.

The hierarchical EVPN access pseudowire process connects the customer edge device to the EVPN core by transporting traffic from U-PE1 over an access pseudowire to N-PE1. N-PE1 then forwards this traffic within the EVPN core to other network provider edges, maintaining a clear separation between the access and core layers.

Workflow



These stages describe how hierarchical EVPN access pseudowire works.

1. The U-PE1 device establishes an attachment circuit connection to CE1.
2. U-PE1 transports CE1 traffic over an EVPN access pseudowire to N-PE1.
3. On N-PE1, the access pseudowire from U-PE1 is treated similarly to an attachment circuit.
4. U-PE1 operates outside the EVPN core and is not part of the core network with other N-PE devices.
5. N-PE1 forwards traffic received from the access pseudowire to core pseudowires within the EVPN core, connecting to other N-PE devices such as N-PE2.

Result

This process enables hierarchical EVPN access by separating the user provider edge from the core provider edge devices, allowing efficient transport of customer traffic from the attachment circuit through access pseudowires into the EVPN core.

Configure hierarchical EVPN access pseudowire

Configure the hierarchical EVPN access pseudowire feature on U-PE and N-PE devices to enable efficient Layer 2 VPN connectivity.

This task applies when setting up hierarchical EVPN access pseudowires to interconnect U-PE and N-PE routers in an EVPN environment.

Procedure

Step 1 Configure the U-PE device.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XG1
Router(config-l2vpn-xc)# p2p P1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/31
```

```
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 target 33 source 33
Router(config-l2vpn-xc-p2p-pw)# commit
```

Step 2 Configure the N-PE device.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group evpn
Router(config-l2vpn-bg)# bridge-domain evpn1
Router(config-l2vpn-bg-bd)# neighbor evpn evi 4 target 33
Router(config-l2vpn-bg-bd)# evi 1
Router(config-l2vpn-bg-bd-evi)# commit
```

Step 3 Use the **show l2vpn bridge-domain bd-name evpn1** command to verify the EVPN state, and the list of access PWs.

The sample output on N-PE1 shows it processing EVPN access pseudowire traffic from U-PE1 like an attachment circuit and forwarding it into the EVPN core to connect with other N-PE devices.

Example:

```
Router:N-PE1# show l2vpn bridge-domain bd-name evpn1
Wed Jun 16 09:22:30.328 EDT
Legend: pp = Partially Programmed.
Bridge group: evpn, bridge-domain: evpn1, id: 1, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 0 (0 up), VFIs: 0, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
  List of ACs:
  List of Access PWs:
      EVPN 4,33,192.168.0.4, state: up, Static MAC addresses: 0
  List of VFIs:
  List of Access VFIs:
```

The hierarchical EVPN access pseudowire is configured successfully, enabling Layer 2 connectivity between U-PE and N-PE devices.

Layer 2 fast reroute

A layer 2 fast reroute (FRR) is a network capability that

- redirects traffic during link or node failures in a layer 2 network
- establishes backup paths to enable rapid switchover and minimize disruption, and
- prevents traffic loss when a PE-CE link fails before the remote PE receives the mass withdrawal message.

Supported Layer 2 fast reroute services

Layer 2 fast reroute is supported on these services:

- E-LAN service—is a multipoint-to-multipoint Layer 2 connection, requiring FRR to handle traffic rerouting across multiple endpoints and maintain seamless connectivity within the LAN segment.
- E-Line service—is a point-to-point Layer 2 connection, so FRR focuses on rerouting traffic between two endpoints.

Layer 2 fast reroute for E-LAN services

A Layer 2 fast reroute (FRR) for E-LAN services is a network capability that

- provides rapid traffic rerouting in the event of a link or node failure within a multipoint Layer 2 network
- improves network reliability by maintaining connectivity among multiple endpoints, and
- ensures service continuity with minimal disruption by pre-establishing backup paths that accommodate the multipoint topology.

Table 30: Feature History Table

Feature Name	Release Information	Feature Description
Layer 2 fast reroute for E-LAN services	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.
Layer 2 fast reroute for E-LAN services	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D • 8711-48Z-M
Layer 2 fast reroute for E-LAN services	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
Layer 2 fast reroute for E-LAN services	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
Layer 2 fast reroute for E-LAN services	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) Fast reroute minimizes traffic loss by quickly redirecting traffic to a backup path in the event of a link failure, ensuring fast convergence and maintaining the service continuity. This feature introduces the **convergence reroute** command.

MAC address handling during AC failure in Layer 2 fast reroute for E-LAN service

In an E-LAN service, local hosts are associated with a specific bridge port based on their MAC addresses, independent of the AC status. When Layer 2 FRR is enabled and an AC fails, the MAC addresses remain linked to the L2 FRR-enabled AC and are not flushed. This mechanism ensures continuous MAC address association despite changes in the AC state, maintaining network stability and connectivity.

Layer 2 fast reroute in all-active multihoming mode

In all-active redundancy mode or single-active mode, configure the AC-backup function to enable fast traffic redirection by using the service label of the all-active peer. This configuration ensures that hosts or MAC addresses remain permanently associated with the AC, maintaining stable and continuous network connectivity during failover events.

Benefits of Layer 2 fast reroute for E-LAN service

L2 FRR for E-LAN service provides several key benefits that enhance network performance and reliability:

- delivers fast and predictable convergence, ensuring minimal disruption during failover events.
- enables rapid failure notification even in large ring topologies with many nodes.
- supports manual configuration to achieve predictable failover behavior tailored to network requirements.
- requires no changes to the existing network topology, simplifying deployment and maintenance.

How Layer 2 fast reroute for EVPN multihomed E-LAN services work

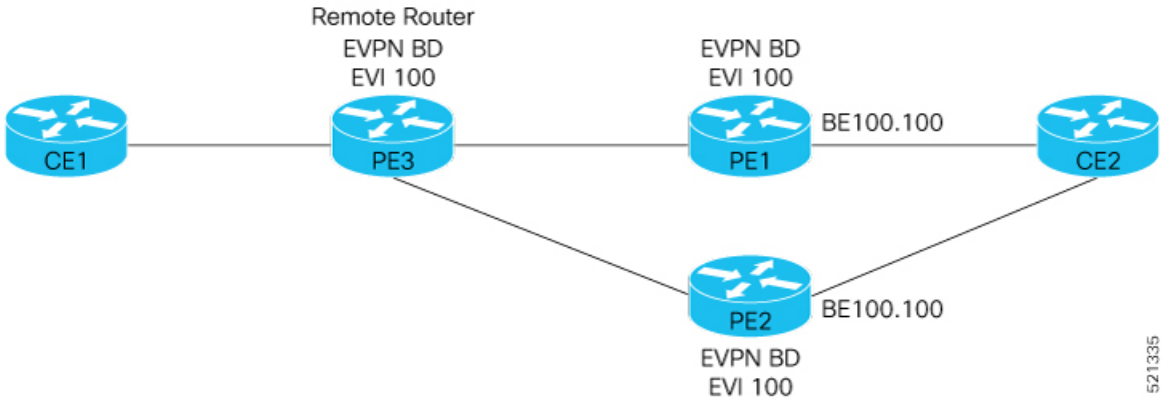
Summary

The key components involved in the L2 FRR process for EVPN multihomed E-LAN services are:

- CE2: Customer edge device multihomed to PE1 and PE2.
- PE1 and PE2: Provider edge routers operating in EVPN active-active or single-active mode, enabled with L2 FRR, and connected to PE3 over the MPLS core.
- PE3: Remote Provider edge router connected to CE1.
- FRR label: A backup path label allocated per Ethernet VPN Instance (EVI) for traffic protection.

The L2 FRR process for EVPN multihomed E-LAN services enables fast traffic redirection by pre-programming backup paths and using FRR labels to ensure seamless failover when a link to a customer edge device fails, minimizing service disruption. This process involves multihomed provider edge routers distributing and forwarding traffic with rapid failover triggered upon link failure.

Workflow



These stages describe how the L2 FRR for EVPN multihomed E-LAN services work.

1. CE1 sends traffic to PE3.
2. PE3 distributes the traffic across PE1 and PE2.
3. PE1 and PE2 forward the traffic to CE2.
4. If the PE1-CE2 link fails, PE1 triggers L2 FRR and redirects traffic to PE2 until network convergence completes.
5. When L2 FRR is enabled on PE1, the backup path to PE2 is pre-programmed in hardware. Upon detecting the failure on the CE2 link, PE1 uses this pre-programmed backup path.
6. PE2 allocates and advertises an FRR label for the protected traffic.
7. All incoming traffic to PE1 is encapsulated with PE2's FRR label and forwarded to PE2.
8. PE2 receives the traffic with the FRR label and forwards it to CE2.

Result

This process ensures fast reroute of traffic in EVPN multihoming modes, minimizing traffic disruption during link failures by pre-establishing backup paths and labels for seamless failover.

Restrictions for Layer 2 fast reroute for E-LAN service

- This feature is supported on EVPN all-active or single-active mode.
- This feature applies only to unicast traffic.
- This feature is not supported for BUM traffic.

Configure Layer 2 fast reroute for E-LAN service

Enable L2 FRR on a PE router to provide fast convergence in an E-LAN EVPN multihoming network.

Use this task to configure L2 FRR on both PE routers in an E-LAN EVPN multihoming setup to ensure rapid failover and maintain service continuity.

Procedure

Step 1 Associate the Ethernet segment with the bundle interface.

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 Bundle-Ether4.1
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)# evi 1
Router(config-l2vpn-bg-bd-evi)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface Bundle-Ether4.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)#
```

Step 2 Enable L2 FRR.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# exit
Router(config-evpn-instance)# exit
Router(config-evpn)# evi 2
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# exit
Router(config-evpn-instance)# exit
Router(config-evpn)# interface Bundle-Ether4
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# convergence
```

```
Router(config-evpn-ac-es-conv)# reroute
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac# exit
Router(config-evpn)# exit
Router(config)# exit
```

Step 3 Use **show evpn ethernet-segment carving detail** and show **l2vpn forwarding interface BE4.1 private location 0/RP0/CPU0** commands to verify L2 FRR configuration.

Example:

```
Router# show evpn ethernet-segment carving detail
...Ethernet Segment Id      Interface                          Nexthops
----------------------- ---------------------------------- --------------------
0040.0000.0000.0000.0001 BE4                                4.5.6.7
                                                            5.6.7.8
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether4
     Interface MAC  : 00c9.c654.9a04
     IfHandle       : 0x7800008c
     State          : Up
     Redundancy     : Not Defined
  ESI ID            : 1
  ESI type          : 0
     Value          : 0040.0000.0000.0000.0001
  ES Import RT      : 4000.0000.0000 (from ESI)
  Topology          :
     Operational    : MH, Single-active
     Configured     : Single-active (AApS)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence    : Reroute
  Peering Details   : 2 Nexthops
     4.5.6.7 [MOD:P:00:T]
     5.6.7.8 [MOD:P:00:T]
  Service Carving Synchronization:
     Mode           : NTP_SCT
     Peer Updates   :
                 4.5.6.7 [SCT: 2025-01-22 17:01:01.1737583]
                 5.6.7.8 [SCT: 2025-01-22 17:00:36.1737583]
  Service Carving Results:
     Forwarders     : 2
     Elected        : 1
            EVI E   :        2
     Not Elected    : 1
            EVI NE  :        1
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flush msg     : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Revert timer      : 0 sec [not running]
  HRW Reset timer   : 5 sec [not running]
  Local SHG label   : 24008
     IPv6_Filtering_ID : 1:16
  Remote SHG labels : 1
              24007 : nexthop 5.6.7.8
  Access signal mode: Bundle OOS
  ...
```

```
Router# show l2vpn forwarding interface BE4.1 private location 0/RP0/CPU0
Wed Jan 22 17:02:01.387 EST

Xconnect ID 0xc0000002

  Xconnect info:
    xcon_status=Up, xcon_bound=TRUE, switching_type=0, data_type=12
    xcon_name=

    Object: XCON
    Base info: version=0xaabbcc13, flags=0x3110, type=2, object_id=UNSPECIFIED, reserved=0


  AC info:
    xcon_id=0xc0000002, ifh=0x7800008c, subifh=0x78000096, ac_id=0, ac_type=21, status=Bound
    ac_mtu=1500, iw_mode=1, adj=150.0.0.120+Bundle-Ether4,
    r_aps_channel=FALSE, prot_exclusion=FALSE
    rg_id=0, ro_id=0x0000000000000000
    evpn internal label = None
    E-Tree = Root
    FXC local-switch AC xcid = 0x0 (Invalid)
    FXC local-switch PW xcid = 0xffffffff (Invalid)
    EVPN MP route flags = 0x0
    Statistics:
      packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 0
      bytes: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 0
      MAC move: 0
      packets dropped: PLU 0, tail 0
      bytes dropped: PLU 0, tail 0

    Object: AC
    Base info: version=0xaabbcc11, flags=0x0, type=3, object_id=0x10001000000002d8|v9, reserved=0

    AC Backup info:
      VC label: 24004
      Local VC label: 0
      Backup Pseudowire XC ID: 0x0
      Statistics:
        packets: received 0, sent 0
        bytes: received 0, sent 0
        packets dropped: PLU 0, tail 0, out of order 0
        bytes dropped: PLU 0, tail 0, out of order 0

      Object: AC_BACKUP
      Base info: version=0xaabbcc39, flags=0x0, type=43, object_id=0x1000100000000300|v1, reserved=0

        Time (~200ms)     Event                     Flags
        =============     =====                     =====
        Jan 22 17:00:58.4 Create                      0x0  -  -
        ------------------------------------------------

      Nexthop info:
        nh_addr=5.6.7.8,
        ecd_plat_data_valid=TRUE, ecd_plat_data_len=104, plat_data_size=232
        child_count=0, child_evpn_ole_count=2, child_mac_count=0, child_pwhe_mp_count=0,
child_ac_backup_count=2,
        child_vni_count=0, child_ifl_count=0, child_sg_count=0

        Object: NHOP
       Base info: version=0xaabbcc14, flags=0x4010, type=7, object_id=0x10001000000002f4|v5, reserved=0


    bp_seg1_type=0x3, mtu=1500
    is_flooding_disabled=FALSE, is_mac_learning_disabled=FALSE, is_mac_port_down_flush_disabled=FALSE,
```

```
EVPN ESI ID: 0
EVPN SHG Local Label: None
EVPN SHG Remote Labels: 0
 MAC learning: enabled
 Software MAC learning: enabled
 MAC port down flush: enabled
 Flooding:
   Broadcast & Multicast: enabled
   Unknown unicast: enabled
 MAC aging time: 300 s, Type: inactivity
 MAC limit: none
 MAC Secure: disabled, Logging: disabled, Accept-Shutdown: enabled
 DHCPv4 snooping: profile not known on this node, disabled
 Dynamic ARP Inspection: disabled, Logging: disabled
 IP Source Guard: disabled, Logging: disabled
 IGMP snooping profile: profile not known on this node
 MLD snooping profile: profile not known on this node
 Router guard disabled
 vES:disabled
 Etree Leaf:disabled
 STP participating: disabled
 Storm control: disabled
 Main port: Bundle-Ether4, MSTI: 2

 Object: BRIDGE_PORT
 Base info: version=0xaabbcc1a, flags=0x0, type=12, object_id=0x10001000000002d9|v6, reserved=0
```

L2 FRR is enabled on the PE routers, providing fast failover in the E-LAN EVPN multihoming network. The Ethernet segment is associated with the bundle interface, and convergence reroute is active, ensuring rapid recovery from failures.

Layer 2 fast reroute for E-Line service

A Layer 2 fast reroute (FRR) for E-Line services is a network capability that

- provides rapid traffic rerouting in the event of a link or node failure
- improves network reliability, and
- ensures service continuity with minimal disruption by pre-establishing backup paths.

Table 31: Feature History Table

Feature Name	Release Information	Feature Description
Layer 2 fast reroute for E-Line services	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
Layer 2 fast reroute for E-Line services	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D • 8711-48Z-M
Layer 2 fast reroute for E-Line services	Release 25.1.1	Introduced in this release on: Fixed Systems 8010 [ASIC: A100]) (select variants only*) You can now ensure faster convergence and uninterrupted service by redirecting the traffic using the EVPN pseudowire (PW) in an E-Line configuration when a dual-homing link fails. *This feature is now supported on: • 8011-4G24Y4H-I

Minimize traffic loss with Layer 2 fast reroute for E-Line service

Layer 2 fast reroute protects the provider edge–customer edge (PE-CE) connection by quickly redirecting traffic through a backup path when a primary link fails. This feature minimizes traffic loss and ensures rapid network convergence. If a local link failure occurs, traffic is rerouted to a peer PE, which then forwards it to the CE. In an E-Line service, an EVPN pseudowire establishes a point-to-point Layer 2 connection over an IP/MPLS network using EVPN. All traffic is redirected to the CE without involving MAC address learning or forwarding.

Benefits of Layer 2 fast reroute for E-Line service

- Achieves fast convergence with a 50 ms target failover time.
- Protects PE-CE links by rerouting traffic to a peer PE in case of local link failure.
- Maintains the same topology, requiring no additional network changes.
- Ensures minimal traffic disruption and rapid recovery in E-Line services.

How Layer 2 fast reroute for EVPN multihomed E-Line services work

Summary

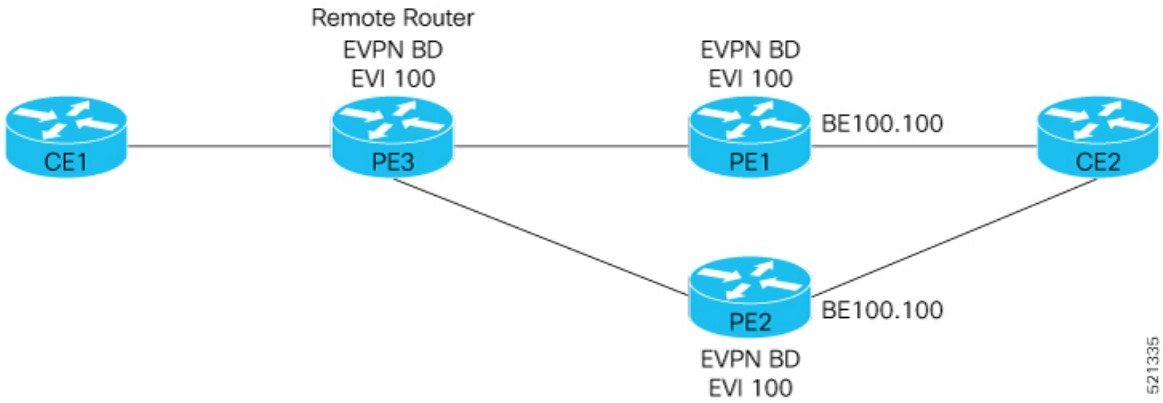
The key components involved in the Layer 2 fast reroute process for EVPN multihomed E-Line services are:

- CE2: A customer edge device connected to both PE1 and PE2 as a multihomed device.

- PE1 and PE2: Provider edge routers operating in EVPN active-active or single-active mode, enabled with L2 FRR and assigned FRR labels per EVI for backup paths.
- PE3: A remote provider edge router connected to CE1 and the MPLS core network.

The Layer 2 FRR process for EVPN multihomed E-Line services rapidly switches traffic from a failed PE to a backup PE using pre-assigned FRR labels.

Workflow



These stages describe how Layer 2 FRR for EVPN multihomed E-Line services work.

1. Normal traffic flow: CE1 sends traffic to PE3. PE3 distributes the traffic to PE1 and PE2, with PE1 acting as the Designated Forwarder (DF). PE1 and PE2 forward the traffic to CE2.
2. Failover scenario: When the link between PE1 and CE2 fails, PE1 detects the failure on the access side.
3. Traffic redirection: PE1 redirects incoming traffic by encapsulating it with PE2's FRR label and forwards it to PE2 over the pre-programmed backup path.
4. Traffic forwarding by PE2: Upon receiving the FRR-labeled traffic, PE2 forwards it to CE2 through the attachment circuit (AC), even if the AC is in a blocking state.
5. Route update: Meanwhile, PE3 updates its routes to send traffic directly to PE2 until the primary link is restored.

Result

This process ensures rapid failover and minimal traffic disruption in EVPN multihomed E-Line services by pre-establishing backup paths and enabling seamless traffic redirection upon link failure.

Restrictions for Layer 2 fast reroute for E-Line service

- This feature is supported on EVPN all-active or single-active mode.
- This feature applies only to unicast traffic.
- This feature is not supported for BUM traffic.

Configure layer 2 fast reroute for E-Line service

Enable L2 FRR on a PE router to enhance network resilience in an E-LINE EVPN multihoming environment.

This task applies to configuring L2 FRR in an EVPN E-Line service where multihoming is deployed with single-active load balancing mode.

Procedure

Step 1 Configure L2 FRR on a PE router in the E-LINE EVPN multihoming network.

Example:

```
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-instance)# exit
Router(config-evpn)# evi 2
Router(config-evpn-instance)# exit
Router(config-evpn)# interface Bundle-Ether4
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# convergence
Router(config-evpn-ac-es-conv)# reroute
Router(config-evpn-ac-es-conv)# exit
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# exit
Router(config)#
```

Step 2 Use **show evpn ethernet-segment carving detail** and **show l2vpn forwarding interface BE4.1 private location 0/RP0/CPU0** commands to verify the L2 FRR EVPN E-LINE configuration and status.

Example:

```
Router# show evpn ethernet-segment carving detail
Wed Jan 22 17:15:05.606 EST
...

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0040.0000.0000.0000.0001 BE4                               4.5.6.7
                                                           5.6.7.8
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether4
     Interface MAC  : 00c9.c654.9a04
     IfHandle       : 0x7800008c
     State          : Up
     Redundancy     : Not Defined
  ESI ID            : 1
  ESI type          : 0
     Value          : 0040.0000.0000.0000.0001
  ES Import RT      : 4000.0000.0000 (from ESI)
  Topology          :
     Operational    : MH, Single-active
     Configured     : Single-active (AApS)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence     : Reroute
  Peering Details   : 2 Nexthops
     4.5.6.7 [MOD:P:00:T]
     5.6.7.8 [MOD:P:00:T]
  Service Carving Synchronization:
```

```
    Mode           : NTP_SCT
    Peer Updates   :
                4.5.6.7 [SCT: 2025-01-22 17:13:55.1737584]
                5.6.7.8 [SCT: 2025-01-22 17:06:30.1737583]
  Service Carving Results:
    Forwarders     : 2
    Elected        : 0
    Not Elected    : 0
  EVPN-VPWS Service Carving Results:
    Primary        : 2
      EVI:ETag P   :      1:2,     2:4
    Backup         : 0
    Non-DF         : 0
  MAC Flush msg    : STP-TCN
  Peering timer    : 3 sec [not running]
  Recovery timer   : 30 sec [not running]
  Carving timer    : 0 sec [not running]
  Revert timer     : 0 sec [not running]
  HRW Reset timer  : 5 sec [not running]
  Local SHG label  : 24008
    IPv6_Filtering_ID : 1:16
  Remote SHG labels : 1
              24007 : nexthop 5.6.7.8
  Access signal mode: Bundle OOS
```

```
Router# show l2vpn forwarding interface BE4.1 private location 0/RP0/CPU0
Wed Jan 22 17:15:29.510 EST

Xconnect ID 0xc0000002

  Xconnect info:
    xcon_status=Up, xcon_bound=TRUE, switching_type=0, data_type=4
    xcon_name=xg1:xc1

    Object: XCON
    Base info: version=0xaabbcc13, flags=0x110, type=2, object_id=UNSPECIFIED, reserved=0


  AC info:
    xcon_id=0xc0000002, ifh=0x7800008c, subifh=0x78000096, ac_id=0, ac_type=21, status=Bound
    ac_mtu=1500, iw_mode=0, adj=150.0.0.120+Bundle-Ether4,
    r_aps_channel=FALSE, prot_exclusion=FALSE
    rg_id=0, ro_id=0x0000000000000000
    evpn internal label = None
    E-Tree = Root
    FXC local-switch AC xcid = 0x0 (Invalid)
    FXC local-switch PW xcid = 0x0 (Invalid)
    EVPN MP route flags = 0x4
    Main port: Bundle-Ether4, MSTI: 3
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      packets dropped: MTU exceeded 0, other 0

    Object: AC
    Base info: version=0xaabbcc11, flags=0x0, type=3, object_id=0x100010000000032a|v5, reserved=0

    AC Backup info:
      VC label: 24012
      Local VC label: 24012
      Backup Pseudowire XC ID: 0x20000005
      Statistics:
        packets: received 0, sent 0
        bytes: received 0, sent 0
        packets dropped: PLU 0, tail 0, out of order 0
```

```
        bytes dropped: PLU 0, tail 0, out of order 0

      Object: AC_BACKUP
      Base info: version=0xaabbcc39, flags=0x0, type=43, object_id=0x100010000000032b|v1, reserved=0



      Nexthop info:
        nh_addr=5.6.7.8,
        ecd_plat_data_valid=TRUE, ecd_plat_data_len=104, plat_data_size=232
        child_count=0, child_evpn_ole_count=0, child_mac_count=0, child_pwhe_mp_count=0,
child_ac_backup_count=2,
        child_vni_count=0, child_ifl_count=0, child_sg_count=0

        Object: NHOP
       Base info: version=0xaabbcc14, flags=0x4010, type=7, object_id=0x100010000000032c|v3, reserved=0



  PW info:
    pw_id=1, 1, nh_valid=TRUE, sig_cap_flags=0x1, context=0x0,
    MPLS, Destination address: 1.2.3.4, evi: 1, ac-id: 1, status: Bound
    Local Pseudowire label: 24013
    Remote Pseudowire label: 24007
    Control word enabled
    EVPN Virtual ES PW: 0
    VFI PW: 0
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      packets dropped: PLU 0, tail 0, out of order 0
      bytes dropped: PLU 0, tail 0, out of order 0

    Object: ATOM
    Base info: version=0xaabbcc12, flags=0x0, type=4, object_id=0x100010000000032d|v3, reserved=0

    Nexthop info:
      nh_addr=1.2.3.4,
      ecd_plat_data_valid=TRUE, ecd_plat_data_len=104, plat_data_size=232
      child_count=2, child_evpn_ole_count=0, child_mac_count=0, child_pwhe_mp_count=0,
child_ac_backup_count=0,
      child_vni_count=0, child_ifl_count=0, child_sg_count=0

      Object: NHOP
     Base info: version=0xaabbcc14, flags=0x4010, type=7, object_id=0x100010000000032e|v3, reserved=0


    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      packets dropped: MTU 0, tail 0, out of order 0
      bytes dropped: MTU 0, tail 0, out of order 0

    PD System Data: Learn key: 0
```

The PE router is configured for Layer 2 FRR in an EVPN E-Line multihoming setup, enabling rapid failover and improved service availability.

EVPN and L3VPN using route type-5 over BGP-LU with SR

EVPN and L3VPN using Route Type-5 over BGP-LU with SR is a network architecture that

- combines EVPN and Layer 3 VPN
- utilizes route type-5 over BGP Layer-3 Unicast (BGP-LU), and
- leverages segment routing (SR) for advanced traffic engineering.

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
EVPN and L3VPN using Route Type-5 over BGP-LU with SR	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.
EVPN and L3VPN using Route Type-5 over BGP-LU with SR	Release 25.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100])(select variants only*) Enhance your network infrastructure with our advanced architecture that seamlessly integrates EVPN, L3VPN, and Route Type-5 over BGP-LU over segment routing, offering a scalable, flexible, and resilient solution for service providers and large enterprises. This design combines the versatility of EVPN for Layer 2 services with the robust scalability of L3VPN, ensuring seamless IP connectivity across multiple sites using route type-5. *This feature is supported on 88-LC1-12TH24FH-E and 88-LC1-52Y8H-EM.

Key concepts of EVPN and L3VPN using route type-5 over BGP-LU with SR

These concepts outline the foundational principles and functionalities of EVPN, L3VPN, route type-5, BGP Layer-3 Unicast, and Segment Routing:

- EVPN is a scalable solution for extending Layer 2 connectivity across geographically dispersed sites, supporting mobility, MAC learning, and multihoming.
- L3VPN provides IP routing services over shared infrastructure, enabling isolated customer traffic across a provider's network.
- Route Type-5 allows the advertisement of Layer 3 prefixes (IPv4 or IPv6) in EVPN, bridging Layer 2 and Layer 3 services.
- BGP Layer-3 Unicast (BGP-LU) distributes unicast IP routes, acting as the control plane to carry both EVPN and L3VPN prefixes.

- Segment routing (SR) simplifies traffic engineering by encoding the path through the network into the packet headers using segments.

Benefits of EVPN and L3VPN using route type-5 over BGP-LU with SR

EVPN and L3VPN leveraging route type-5 over BGP-LU with SR offer these benefits:

- Seamless Layer 2 and Layer 3 integration—EVPN offers efficient Layer 2 extensions with MAC address learning through control-plane, eliminating flooding. L3VPN provides scalable VRF-based IP routing, enhancing tenant or service segregation.
- Service multiplexing—Supports multiple VRFs with unique mappings to BD, BVI, and EVI for granular traffic segmentation.
- Scalable core with BGP-LU—BGP-LU enables end-to-end LSPs across multiple IGP domains, facilitating inter-domain segment routing transport while decoupling core and service layers.
- Inter-domain SR transport—BGP-LU facilitates seamless SR between IGP domains.

References

For detailed information and configuration steps for EVPN, L3VPN, BGP-LU, and SR, refer to the configuration guides:

- *BGP Configuration Guide for Cisco 8000 Series Routers*
- *L2VPN Configuration Guide for Cisco 8000 Series Routers*
- *L3VPN Configuration Guide for Cisco 8000 Series Routers*
- *Segment Routing Configuration Guide for Cisco 8000 Series Routers*

Sub-second convergence for EVPN with BGP PIC-edge

Sub-second convergence for EVPN with BGP PIC-edge is a network functionality that

- maintains uninterrupted service during network failures
- delivers rapid convergence for active-active EVPN ELAN and E-Line services, and
- enables immediate switchover to backup nexthop path for user traffic when a preferred path becomes unavailable.

Table 33: Feature History Table

Feature Name	Release Information	Feature Description
Sub-second convergence for EVPN with BGP PIC-edge	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.

Feature Name	Release Information	Feature Description
Sub-second convergence for EVPN with BGP PIC-edge	Release 25.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8200 [ASIC: P100], 8700 [ASIC: P100, K100]); Modular Systems (8800 [LC ASIC: P100]) You can maintain continuous service in multi-homed EVPN deployments using sub-second convergence for EVPN with BGP PIC-edge. This functionality rapidly switches traffic to a backup nexthop path when the preferred nexthop fails, delivering fast convergence and high availability for active-active EVPN E-LAN and E-Line services.

EVPN network resiliency with sub-second BGP PIC-edge convergence

Sub-second convergence for EVPN with BGP PIC-edge enhances network availability and reliability by minimizing service disruptions in large-scale environments where even brief outages affect users and business operations.

Key features include:

- Preprogrammed primary and backup paths: Both active and standby routes are configured in advance to maintain service continuity during failures.
- Automatic traffic redirection: The system reroutes traffic seamlessly to backup paths upon primary path failure and restores normal operation quickly when the primary path is available.
- Rapid convergence for active-active EVPN E-LAN and E-Line services: Multi-homed deployments achieve sub-second recovery times independent of the underlying BGP and EVPN prefixes.

Supported deployment scenarios for sub-second EVPN convergence

The sub-second EVPN convergence supports these deployment scenarios:

When...	and remote PE is reachable through...	the feature provides...
the remote PE is within the same IGP domain	IGP	sub-second EVPN convergence for node failure in active-active Multihoming scenarios.
the remote PE is in a different IGP domain	BGP-LU	sub-second EVPN convergence in inter-IGP domain scenarios.

Benefits of sub-second convergence for EVPN with BGP PIC-edge

These are some of the benefits of sub-second convergence for EVPN with BGP PIC-edge:

- Rapid traffic recovery: Enables sub-second failover when a primary path or node fails, minimizing traffic disruption for end users.

- Improved network resiliency: Enhances the ability of the network to quickly recover from failures, supporting high-availability services.
- Prefix-Independent Convergence (PIC): Pre-programs backup paths in hardware, allowing instant traffic switching without software re-programming of each prefix.
- Optimized for active-active multihoming: Provides seamless traffic redirection in active-active multihoming scenarios.
- Scales efficiently for large EVPN deployments.

Limitations of sub-second convergence for EVPN with BGP PIC-edge

- Sub-second convergence applies only to unicast EVPN traffic. Backup paths can be pre-programmed in hardware only for unicast prefixes.
- Fast convergence for BUM traffic is not supported.
- Sub-second convergence is supported only when a single EVPN nexthop path goes down at a time. Multiple simultaneous failures or mass flaps may not achieve sub-second recovery.
- Use the **preferred-nexthop {[highest-ip] [lowest-ip]}** command to enable sub-second convergence.
- Backup path pre-programming relies on hardware capabilities.

How sub-second convergence for EVPN with BGP PIC-edge works

Sub-second EVPN convergence features minimize traffic interruption during network failures by switching to a pre-programmed backup nexthop path without control-plane delay.

Summary

The key components involved in the process are:

- Primary path: The main route used for forwarding EVPN service traffic.
- Backup path: An alternate route pre-programmed in hardware to take over if the primary path fails.
- Forwarding Information Base (FIB): A table that manages path selection and failover.

Workflow

These stages describe how the sub-second EVPN convergence works.

1. When the preferred path is operational, all traffic is sent using the primary path.
2. If the primary path fails because of a node failure, link failure, or IGP event, the hardware immediately switches forwarding to the backup path without waiting for the control plane to reconverge.
3. The system continues forwarding over the backup path until the primary path is restored or a new preferred path is configured.
4. If the primary path is not restored, the control plane later reconverges and programs the backup path as the sole forwarding path.

Result

Service interruption is minimized, and traffic switchover occurs in less than one second in supported scenarios.

Configure sub-second convergence for EVPN with BGP PIC-edge

Achieve sub-second EVPN convergence by enabling rapid failover through preferred next-hop configuration.

This task applies when you want to optimize EVPN path selection to ensure fast failover between primary and backup paths.

Before you begin

Ensure you have access to the router CLI and necessary privileges to configure EVPN.

Procedure

Step 1 Configure the preferred next-hop for the EVPN instance to optimize path selection for rapid failover.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-instance)# preferred-nexthop highest-ip
Router(config-evpn-instance)# commit
```

Step 2 Run the **show running-config evpn** command to ensure the configuration is active.

Example:

```
Router#show running-config evpn
evpn
 evi 100
  preferred-nexthop highest-ip
 !
!
```

The router is configured for sub-second EVPN convergence, with both primary and backup paths pre-programmed to provide rapid and automatic failover.

Layer 3 EVPN IGMP and MLD state synchronization

Layer 3 EVPN IGMP and Multicast Listener Discovery (MLD) state synchronization is a network solution that

- synchronizes IPv4 IGMP and IPv6 MLD states across multiple PE devices
- ensures reliable and seamless multicast service delivery in residential fiber-to-the-home (FTTH) deployments, and
- removes the need for complex L2 and integrated routing and bridging (IRB) configurations by utilizing L3 subinterfaces.

Table 34: Feature History Table

Feature Name	Release Information	Feature Description
Layer 3 EVPN IGMP and MLD state synchronization	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Layer 3 EVPN IGMP and MLD state synchronization	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.
Layer 3 EVPN IGMP and MLD state synchronization	Release 25.3.1	Introduced in this release on: Fixed Systems(8200, 8700, 8011)(select variants only*); Modular Systems (8800 [LC ASIC: P100]) You can ensure seamless and reliable multicast delivery in residential FTTH networks with IGMP and MLD state synchronization for L3 using EVPN. This feature synchronizes IPv4 IGMP and IPv6 Multicast Listener Discovery (MLD) states across multiple PE devices using L3 sub-interfaces, eliminating the need for complex L2 or IRB configurations. It supports both VRF and global routing table deployments, providing flexibility for various network designs. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 8011-4G24Y4H-I

Simplified multicast delivery in residential FTTH networks

In many fiber-to-the-home (FTTH) deployments, multicast receivers are located within residential networks where hosts do not communicate with each other. To minimize costs, service providers often deploy basic Layer 2 switches in these networks and connect them to Provider Edge (PE) devices using Layer 2 port channels. However, when PE devices terminate traffic at Layer 2, Integrated Routing and Bridging (IRB) interfaces must be configured to bridge Layer 2 and Layer 3 domains. This adds unnecessary complexity because there is no host-to-host communication in these environments.

Including L2 switching and IRB interfaces in such scenarios increases operational overhead without providing tangible benefits. A simpler architecture can be achieved by terminating access connections as Layer 3

subinterfaces on the PE device. This eliminates the need for IRB interfaces, streamlines forwarding logic, and results in a more efficient and cost-effective design for residential multicast delivery.

Using EVPN for IGMP and MLD state synchronization at Layer 3 ensures consistent IPv4 and IPv6 multicast group membership across multiple PE devices. This approach supports resilient multicast services in residential FTTH networks by leveraging Layer 3 connectivity. The design aligns with RFC 9251, providing robust and scalable multihoming capabilities.

Multihomed topologies for IGMP and MLD state synchronization

A multihomed topology for IGMP and MLD state synchronization is a network design that

- connects CE devices to multiple PE devices
- uses L3 subinterfaces for direct connections, and
- supports multicast state synchronization to ensure uninterrupted service for both IPv4 and IPv6 multicast traffic.

Benefits of Layer 3 EVPN IGMP and MLD state synchronization

Layer 3 EVPN IGMP and MLD state synchronization offers several key benefits:

- Simplifies network architecture by removing unnecessary Layer 2 switching in residential deployments.
- Enhances scalability and lowers operational overhead through the adoption of L3 multihoming, as specified in RFC 9251.
- Boosts multicast performance and reliability by streamlining forwarding processes, improving efficiency across different IP versions.

Guidelines for Layer 3 EVPN IGMP and MLD state synchronization

- Do not change the standard multicast routing configuration when enabling Layer 3 EVPN IGMP and MLD state synchronization.
- Use only bundle subinterfaces to support Layer 3 EVPN IGMP and MLD state synchronization.
- Deploy this feature in both VRF and global routing table environments.
- Configure all IGMP (IPv4) and MLD (IPv6) parameters—such as timers, versions, and querier settings—identically across all redundancy groups.
- If static joins are necessary on multihomed ports, configure them identically across redundancy groups because static joins are not synchronized automatically.
- When using PIM or PIMv6 with IGMP or MLD state synchronization, configure multicast redundancy in all-active mode only; single-active and port-active modes are not supported.

How Layer 3 EVPN IGMP and MLD state synchronization works

In EVPN multihoming, the designated forwarder (DF) exclusively forwards multicast traffic, including IGMP and MLD queries, to the customer edge, while the non-designated forwarder (NDF) blocks duplicates to prevent loops. This IGMP and MLD state synchronization ensures efficient, loop-free multicast delivery by coordinating forwarding roles across Layer 2 and Layer 3 components.

Summary

The key components involved in IGMP and MLD state synchronization are:

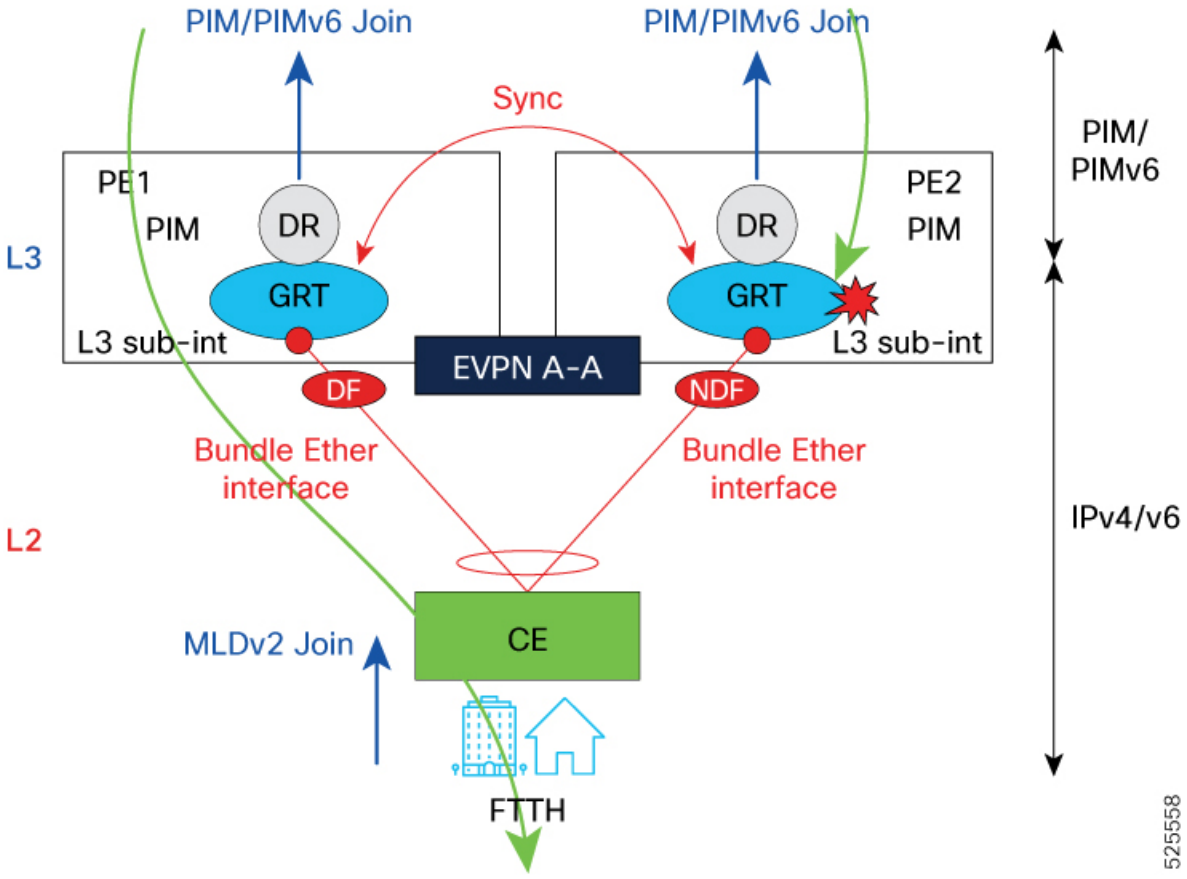
- L3
  - PE1 and PE2: Both act as PE routers running PIM/PIMv6 (Protocol Independent Multicast for IPv4/IPv6) and connect to the customer edge (CE) via Bundle Ethernet interfaces.
  - Global routing table (GRT): Handles unicast and multicast routing.
  - Designated router (DR): Manages multicast group membership.
  - Designated forwarder (DF): Forwards multicast traffic on the shared segment as elected per EVPN rules.

    In EVPN multihoming scenarios, the DF is responsible for forwarding multicast traffic, including IGMP and MLD queries, to the CE. Only the DF PE forwards such traffic, while the Non-Designated Forwarder (NDF) blocks it to prevent duplication and loops. This mechanism ensures that the CE receives a single, loop-free copy of multicast queries and traffic.
  - EVPN A-A (All-active): Enables both PEs to be active and participate in forwarding.
- L2
  - CE: Device that receives L2 multicast (IGMP/MLD) joins from FTTH subscribers.

IGMP and MLD state synchronization ensures efficient and loop-free multicast delivery in EVPN active-active multihoming environments by coordinating state and forwarding roles among L2 and L3 network components.

Workflow

Figure 15: A sample toplogy for L3 EVPN IGMP and MLD state synchronization



This diagram illustrates the operation of L3 EVPN for synchronizing IGMP and MLD state across PE devices in a dual-homed all-active topology. The goal is to ensure consistent multicast group state between PE1 and PE2 for seamless multicast forwarding, even in the event of a link or device failure.

This example focuses on multicast running in the global routing table (GRT), but the process is the same when multicast operates in a VRF (MVPN).

These stages describe multicast state synchronization:

1. Reception of IGMP/MLD joins from the customer edge (CE):

   The CE device sends IGMP (for IPv4) or MLD (for IPv6) membership reports upstream to either PE1 or PE2 over the Layer 2 Bundle Ethernet interface to indicate interest in specific multicast groups.

2. State learning and actions:

   The PE that receives the IGMP or MLD join sends a PIM or PIMv6 join message to the core to request the multicast stream. Because both PEs act as PIM or PIMv6 DRs for the Bundle-Ether interface, either PE that receives the IGMP or MLD join will trigger a PIM or PIMv6 join to the core.

3. State synchronization between PEs:

   IGMP and MLD state information is synchronized between PE1 and PE2 using EVPN mechanisms. The PEs exchange synchronization messages containing IGMP and MLD states, including group membershipand

source details, over the EVPN all-active control plane. This ensures both PEs maintain an identical view of multicast group membership, regardless of which PE received the original join.

4. Live-live redundancy:

   IGMP/MLD state synchronization allows the second PE to also send PIM or PIMv6 join towards the core to request the multicast stream. Both PEs receive the multicast stream from the core, ensuring live-live redundancy.

5. Multicast traffic forwarding:

   Both PEs receive the multicast traffic, but only the DF sends it through the Bundle-Ether interface. The non-designated forwarder (NDF) does not forward the traffic.

6. Failure scenario:

   If the Bundle-Ether interface on the DF fails, the second PE automatically becomes the DF and immediately forwards the multicast stream. This ensures uninterrupted multicast delivery.

Example scenario

1. A subscriber sends an IGMP join message for a TV multicast group from behind the CE device.
2. The CE forwards the IGMP join on one of the two bundle links; for example, the join is received by PE1.
3. Both PE1 and PE2 are designated routers (DRs). Regardless of which PE receives the IGMP join, a Protocol Independent Multicast (PIM) join is sent upstream to request multicast traffic.
4. PE1 exchanges its IGMP state with PE2 over EVPN active-active synchronization.
5. When PE2 receives the IGMP state over EVPN A-A, it also sends a PIM join upstream.
6. PE1 forwards the multicast stream through the Bundle-Ether interface because it is the DF.
7. If PE1 fails, PE2 already maintains the multicast group state and receives the stream from the core, allowing it to immediately take over forwarding multicast traffic to the subscriber.

Note Because of IGMP/MLD state synchronization, the process operates the same way if the join is received on PE2, the non-designated forwarder, in step 2.

Configure Layer 3 EVPN IGMP and MLD state synchronization

Enable and maintain synchronized IGMP and MLD multicast group state across dual-homed PE routers in an EVPN all-active multihoming environment to ensure efficient, loop-free multicast delivery.

This task applies to Layer 3 EVPN deployments where PE routers connect to customer edge devices via Bundle Ethernet interfaces, supporting multicast traffic with redundancy and active-active forwarding.

Before you begin

- Enable and activate the **l2vpn evpn** address family on peering with route reflectors to allow EVPN control plane messages to be exchanged between PEs.

- Ensure that the same ESI value is configured under the EVPN settings on both PEs for the Bundle-Ether interface.

  For more details on EVPN configuration, see EVPN MPLS Multihoming.

- Layer 3 EVPN IGMP and MLD state synchronization requires the standard multicast configuration. For more details, see the *Multicast Configuration Guide for Cisco 8000 Series Routers*.

Procedure

Step 1 Configure the Ethernet Virtual Instance (EVI) to be used for EVPN synchronization.

a) Run the **evpn route-sync <evi>** command if multicast operates in GRT.

Example:

```
Router#config
Router(config)#evpn route-sync 10
Router(config-evpn-instance)#commit
```

The **evpn route-sync <evi>** command enables route synchronization for a specific EVI in the GRT. Use this command when multicast operates in the GRT and the ESI and associated bundle subinterfaces are not tied to any VRF. The command ensures that Layer 3 routes for the EVPN instance are synchronized across all PE devices within the GRT.

b) Run the **evpn route-sync <evi>** command if multicast operates in default VRF.

Example:

```
Router#config
Router(config)#evpn
Router(config-evpn)#route-sync 10
Router(config-evpn-instance)#vrf default
Router(config-evpn-instance)#exit
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 00.01.00.ac.00.00.01.0a.00
Router(config-evpn-ac-es)#commit
```

The **evpn route-sync <evi>** command enables route synchronization for a specific EVI in the default VRF context. This command synchronizes Layer 3 routes learned on bundle subinterfaces or Ethernet segments for that EVPN instance across all PE devices within the default VRF.

Note
The ethernet-segment **identifier type** must match the one configured on the remote dual-homed PE router.

c) Run the **vrf <name> evpn-route-sync <evi>** command if multicast operates in a private VRF.

Example:

```
Router#config
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 00.01.00.ac.00.00.01.0a.00
Router(config-evpn-ac-es)#root
Router(config)#vrf vrf-name evpn-route-sync 10
Router(config)#commit
```

The **vrf <name> evpn-route-sync <evi>** command enables route synchronization for a specific EVI within a non-default VRF context. Use this command when the Ethernet segment identifier (ESI) and related bundle

subinterfaces are part of a non-default VRF. The command ensures that Layer 3 routes for that EVPN instance are synchronized across all PE devices in the specified VRF.

Use a unique EVI number for this CLI; do not use the same EVI under **l2vpn** configuration for standard EVPN services.

Step 2 Use the **show mfib platform evpn bucket loc <location>** command to display EVPN bucket information for the specified location.

Example:

```
Router#show mfib platform evpn bucket loc 0/0/CPU0

------------------------------------------------------------------------------
ESI Interface        Handle     Bucket ID  State  Ref cnt    Stale  Del pending
------------------------------------------------------------------------------
BE1                  0x7800008c 8          NDF    0          F      F
BE1                  0x7800008c 9          DF     0          F      F
BE1                  0x7800008c 10         NDF    0          F      F
BE1                  0x7800008c 11         DF     0          F      F
BE1                  0x7800008c 4          NDF    0          F      F
BE1                  0x7800008c 5          DF     4          F      F
BE1                  0x7800008c 6          NDF    0          F      F
BE1                  0x7800008c 7          DF     0          F      F
BE1                  0x7800008c 0          NDF    0          F      F
BE1                  0x7800008c 1          DF     0          F      F
BE1                  0x7800008c 2          NDF    0          F      F
BE1                  0x7800008c 3          DF     0          F      F
BE2                  0x78000094 4          NDF    0          F      F
BE2                  0x78000094 5          DF     0          F      F
BE2                  0x78000094 6          NDF    0          F      F
BE2                  0x78000094 7          DF     0          F      F
BE2                  0x78000094 0          NDF    0          F      F
BE2                  0x78000094 1          DF     0          F      F
BE2                  0x78000094 2          NDF    0          F      F
BE2                  0x78000094 3          DF     0          F      F
BE2                  0x78000094 8          NDF    0          F      F
BE2                  0x78000094 9          DF     0          F      F
BE2                  0x78000094 10         NDF    0          F      F
BE2                  0x78000094 11         DF     0          F      F
------------------------------------------------------------------------------
Router#
```

This output shows that EVPN multi-homing is enabled with multiple buckets per ESI, and DF roles are distributed across different buckets and interfaces. The mix of DF and NDF states indicates active load balancing and redundancy, ensuring resiliency in multicast forwarding. No entries are stale or pending deletion.

Step 3 Use the **show mrib platform idb** command to display the multicast routing interface database (IDB) information for all interfaces on the platform.

Example:

```
Router#show mrib platform idb

------------------------------------------------------
IDB Hash Table (Total Count 5)
------------------------------------------------------
  ------------------------------------------------------------
  Bundle-Ether1 (0x7800008c)
  ------------------------------------------------------------
  ------------------------------------------------------------

  ------------------------------------------------------------
  Bundle-Ether2 (0x78000094)
```

```
------------------------------------------------------------
------------------------------------------------------------

------------------------------------------------------------
Bundle-Ether1.10 (0x7800009c)
------------------------------------------------------------
 Root Interface:     Bundle-Ether1 (0x7800008c)
 EVPN registered:    T
 ESI IFH:            0x7800008c
 MH count:           2
------------------------------------------------------------

------------------------------------------------------------
Bundle-Ether1.11 (0x780000a4)
------------------------------------------------------------
 Root Interface:     Bundle-Ether1 (0x7800008c)
 EVPN registered:    T
 ESI IFH:            0x7800008c
 MH count:           2
------------------------------------------------------------

------------------------------------------------------------
Bundle-Ether2.10 (0x780000d4)
------------------------------------------------------------
 Root Interface:     Bundle-Ether2 (0x78000094)
 EVPN registered:    T
 ESI IFH:            0x78000094
 MH count:           2
------------------------------------------------------------

------------------------------------------------------
```

This output shows that sub-interfaces Bundle-Ether1.10, Bundle-Ether1.11, and Bundle-Ether2.10 are registered for EVPN multi-homing, each associated with their root bundle and ESI. The MH count indicates that each sub-interface is participating in a multi-homing group with two members, confirming redundancy and active EVPN multi-homing configuration.

Step 4 Use the **show mfib vrf <vrf-name> platform route olist det location <location>** command to display multicast forwarding details, highlighting EVPN multi-homing information for each outgoing interface.

Example:

a) Use the **show mfib vrf vpn101 platform route olist det location 0/0/CPU0** command to verify that both BE1.10 and BE1.11 subinterfaces are multihomed under the same ESI, with both acting as designated forwarders for multicast traffic.

Example:

```
Router#show mfib vrf vpn101 platform route olist det location 0/0/CPU0

------------------------------------------------------------------------------
Legend:
Route Information
    MC GID:    Multicast Index     NPI:     NP Independent

Outgoing Interface Information
    UL_Intf: Underlying Interface  UL_IFH:  Underlying Interface Handle
    L:       Local Interface       B:       Bundle Interface
    O:       In NPI Layer          OT:      OLE TYPE
    MRID:    Multicast Group Index DF:      Designated Forwarder
    EI:      ESI IFH               BI:      Bucket ID
    SE:      Last sync error reported in OFA
    AE:      Last async error reported in OFA
------------------------------------------------------------------------------
```

```
VRF_ID: 0x0          Source: 40.10.1.2       Group: 232.0.0.1       Mask: 64

   SW Route Information
   ----------------------------
   Global MC GID: 317        Scale mode: Not-Set          Total OLE_CNT:2
   ------------------------------------------------------------------------------

   SW OLE Information
   ------------------------------------------------------------------------------
   Interface        IFH        UL_Intf     UL_IFH     L  B  O  SE   AE    NP
   ------------------------------------------------------------------------------
   BE1.11           0x780000a4 Hu0/0/0/0/1 0x368      F  T  T  0x0  0x0   0xff
   BE1.10           0x7800009c Hu0/0/0/0/1 0x368      F  T  T  0x0  0x0   0xff
   ------------------------------------------------------------------------------

   EVPN Multi-homing Information
   -------------------------------------
   Intf             EI         BI  DF
   -------------------------------------
   BE1.11           0x7800008c 5   DF
   BE1.10           0x7800008c 5   DF
   -------------------------------------

   HW OLE NPI Information
   ------------------------------------------------------------------------------
   Interface   IFH        UL_Intf     UL_IFH     NP ID  MRID   OT       IC
   ------------------------------------------------------------------------------
   BE1.11      0x780000a4 Hu0/0/0/0/1 0x368      0x0    0      BE-Sub   F
   BE1.10      0x7800009c Hu0/0/0/0/1 0x368      0x0    0      BE-Sub   F
   ------------------------------------------------------------------------------

VRF_ID: 0x0          Source: 40.10.1.2       Group: 232.0.0.2       Mask: 64

   SW Route Information
   ----------------------------
   Global MC GID: 318        Scale mode: Not-Set          Total OLE_CNT:2
   ------------------------------------------------------------------------------

   SW OLE Information
   ------------------------------------------------------------------------------
   Interface        IFH        UL_Intf     UL_IFH     L  B  O  SE   AE    NP
   ------------------------------------------------------------------------------
   BE1.11           0x780000a4 Hu0/0/0/0/0 0x2b0      F  T  T  0x0  0x0   0xff
   BE1.10           0x7800009c Hu0/0/0/0/0 0x2b0      F  T  T  0x0  0x0   0xff
   ------------------------------------------------------------------------------

   EVPN Multihoming Information
   -------------------------------------
   Intf             EI         BI  DF
   -------------------------------------
   BE1.11           0x7800008c 5   DF
   BE1.10           0x7800008c 5   DF
   -------------------------------------

   HW OLE NPI Information
   ------------------------------------------------------------------------------
   Interface   IFH        UL_Intf     UL_IFH     NP ID  MRID   OT       IC
   ------------------------------------------------------------------------------
   BE1.11      0x780000a4 Hu0/0/0/0/0 0x2b0      0x0    0      BE-Sub   F
   BE1.10      0x7800009c Hu0/0/0/0/0 0x2b0      0x0    0      BE-Sub   F
   ------------------------------------------------------------------------------
```

The output confirms that both BE1.10 and BE1.11 subinterfaces are multihomed under the same ESI, with both acting as designated forwarders for multicast traffic. This indicates a redundant and active-active multi-homing setup, providing resiliency for multicast forwarding in the EVPN environment.

b) Use the **show mfib vrf vpn101 platform route olist det location 0/RP0/CPU0** command to verify that the redundant EVPN multihoming setup is configured correctly, with multiple subinterfaces grouped into different ESIs.

Example:

```
Router# show mfib vrf vpn101 platform route olist det location 0/RP0/CPU0

------------------------------------------------------------------------------
Legend:
Route Information
    MC GID:    Multicast Index     NPI:     NP Independent

Outgoing Interface Information
    UL_Intf: Underlying Interface  UL_IFH:  Underlying Interface Handle
    L:       Local Interface       B:       Bundle Interface
    O:       In NPI Layer          OT:      OLE TYPE
    MRID:    Multicast Group Index DF:      Designated Forwarder
    EI:      ESI IFH               BI:      Bucket ID
    SE:      Last sync error reported in OFA
    AE:      Last async error reported in OFA
------------------------------------------------------------------------------


 VRF_ID: 0x0          Source: 40.10.1.2       Group: 232.0.0.1       Mask: 64

    SW Route Information
    ----------------------------
    Global MC GID: 318        Scale mode: Not-Set          Total OLE_CNT:3
    ------------------------------------------------------------------------------

    SW OLE Information
    ------------------------------------------------------------------------------
    Interface        IFH        UL_Intf     UL_IFH     L  B  O  SE   AE    NP
    ------------------------------------------------------------------------------
    BE1.11           0x780000a4 FH0/0/0/1   0x78000198 F  T  T  0x0  0x0   0xff
    BE1.10           0x7800009c FH0/0/0/1   0x78000198 F  T  T  0x0  0x0   0xff
    BE2.10           0x780000d4 FH0/0/0/2   0x780001a0 F  T  T  0x0  0x0   0xff
    ------------------------------------------------------------------------------

    EVPN Multihoming Information
    -------------------------------------
    Intf            EI         BI  DF
    -------------------------------------
    BE1.11          0x7800008c 5   NDF
    BE1.10          0x7800008c 5   NDF
    BE2.10          0x78000094 5   NDF
    -------------------------------------

    HW OLE NPI Information
    ------------------------------------------------------------------------------
    Interface   IFH        UL_Intf     UL_IFH     NP ID  MRID   OT       IC
    ------------------------------------------------------------------------------
    BE1.11      0x780000a4 FH0/0/0/1   0x78000198 0x0    0      BE-Sub   F
    BE1.10      0x7800009c FH0/0/0/1   0x78000198 0x0    0      BE-Sub   F
    BE2.10      0x780000d4 FH0/0/0/2   0x780001a0 0x0    0      BE-Sub   F
    ------------------------------------------------------------------------------

 VRF_ID: 0x0          Source: 40.10.1.2       Group: 232.0.0.2       Mask: 64

    SW Route Information
```

```
----------------------------
Global MC GID: 319        Scale mode: Not-Set          Total OLE_CNT:3
-------------------------------------------------------------------------------

SW OLE Information
-------------------------------------------------------------------------------
Interface        IFH        UL_Intf     UL_IFH     L  B  O  SE   AE    NP
-------------------------------------------------------------------------------
BE1.11           0x780000a4 FH0/0/0/0   0x78000190 F  T  T  0x0  0x0   0xff
BE1.10           0x7800009c FH0/0/0/0   0x78000190 F  T  T  0x0  0x0   0xff
BE2.10           0x780000d4 FH0/0/0/2   0x780001a0 F  T  T  0x0  0x0   0xff
-------------------------------------------------------------------------------

EVPN Multihoming Information
-----------------------------------
Intf            EI         BI  DF
-----------------------------------
BE1.11          0x7800008c 5   NDF
BE1.10          0x7800008c 5   NDF
BE2.10          0x78000094 5   NDF
-----------------------------------

HW OLE NPI Information
-------------------------------------------------------------------------------
Interface   IFH        UL_Intf     UL_IFH     NP ID  MRID   OT       IC
-------------------------------------------------------------------------------
BE1.11      0x780000a4 FH0/0/0/0   0x78000190 0x0    0      BE-Sub   F
BE1.10      0x7800009c FH0/0/0/0   0x78000190 0x0    0      BE-Sub   F
BE2.10      0x780000d4 FH0/0/0/2   0x780001a0 0x0    0      BE-Sub   F
-------------------------------------------------------------------------------
```

This output shows a redundant EVPN multihoming setup with multiple subinterfaces grouped into different ESIs. However, for the listed multicast groups, none of these local interfaces are acting as the designated forwarder, indicating that the DF role is handled elsewhere or is pending election.

Virtual Ethernet segment

A virtual Ethernet segment is a logical Ethernet segment that

- aggregates multiple physical Ethernet segments into a single common segment visible to the CE device
- enables multi-homing access to EVPN bridges through an MPLS network, and
- provides connectivity to PWs and AC sub-interfaces for redundancy and load balancing.

Table 35: Feature History Table

Feature Name	Release Information	Feature Description
Virtual Ethernet Segment	Release 26.2.1	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
Virtual Ethernet Segment	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Virtual Ethernet Segment	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
Virtual Ethernet Segment	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The Virtual Ethernet Segment functionality is now extended to the Cisco 8712-MOD-M routers.
Virtual Ethernet Segment	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The Virtual Ethernet Segment functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Virtual Ethernet Segment	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) A Virtual Ethernet Segment (VES) allows a Customer Edge (CE) device to connect to an EVPN service over an MPLS network, which can be used for redundancy and load balancing. *This feature is supported only on routers with the 88-LC1-36EH line cards.

Virtual Ethernet segment architecture for multihomed CE-PE connectivity in EVPN

A CE device connects to multiple PE devices, with each CE-PE connection forming an individual Ethernet segment (ES). When these multiple Ethernet segments are combined and presented as a single logical segment to the CE device, this combined entity is called a virtual Ethernet segment (VES). The VES uses a PW as the logical link between the CE and PE devices to facilitate access to EVPN bridges. This architecture supports network resilience and efficient traffic distribution by enabling access through both pseudowires and AC sub-interfaces.

How virtual Ethernet segment works

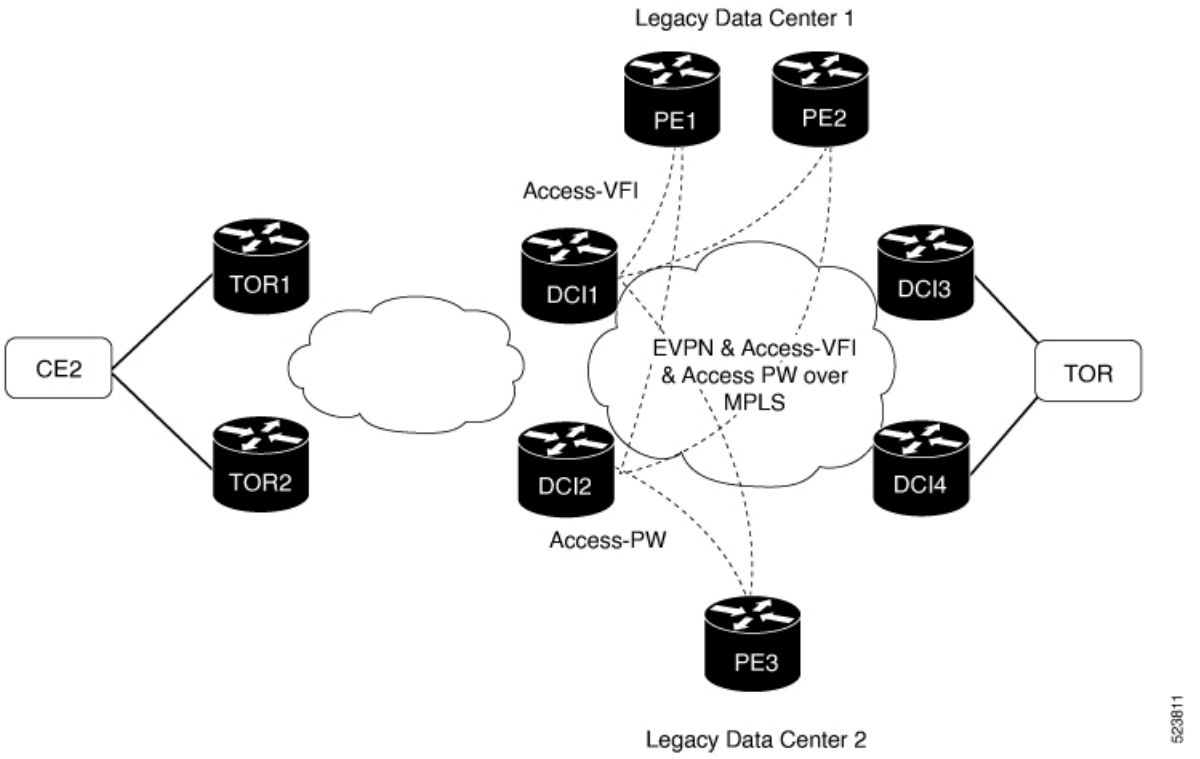
Summary

The key components involved in the VES traffic flow process are:

- CE2: Customer edge device that initiates traffic.
- DCI1 and DCI2: EVPN data centers connected to legacy data centers via access pseudowires on a single Ethernet segment.
- PE1, PE2, and PE3: Provider edge devices in legacy data centers receiving traffic from EVPN data centers.
- Designated Forwarder (DF) and non-DF: Roles elected by DCI1 and DCI2 to manage traffic forwarding and standby paths.

The VES process enables resilient traffic forwarding by having CE2 send traffic through EVPN data centers, which elect a DF to manage active paths while the non-DF remains on standby, ensuring efficient and redundant connectivity to legacy data center PEs.

Workflow



These stages describe how virtual Ethernet segment works.

1. CE2 sends traffic to either DCI1 or DCI2 through the EVPN network.
2. DCI1 and DCI2 advertise Type 4 routes and discover each other.
3. DCI1 and DCI2 perform a DF election; one becomes the DF, and the other becomes the non-DF.

4. For traffic destined to PE3 (Legacy Data Center 2), the DF forwards traffic through the access pseudowire on the single Ethernet segment; the non-DF path remains in standby.
5. For traffic destined to PE1 and PE2 (Legacy Data Center 1), the DF forwards traffic to PE1 and PE2; the non-DF path remains in standby.

Result

This process ensures loop-free and resilient traffic forwarding between EVPN data centers and legacy data centers over a virtual Ethernet segment by using Type 4 route advertisement and designated forwarder election, optimizing traffic flow and providing redundancy.

Configure virtual Ethernet segment

Configure access PWs to act as VES, enabling resilient and loop-free forwarding between EVPN data centers and legacy data centers.

Use this task to set up VES on DCI1, DCI2, and PE3 devices, connecting EVPN data centers to legacy data centers through access pseudowires on a single Ethernet segment.

Procedure

Step 1 Configure DCI1 with bridge domain and assign EVI 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-bg-bd)# neighbor 70.70.70.70 pw-id 17300001
Router(config-bg-bd-pw)# evi 1
Router(config-bg-bd-pw-evi)# member vni 10001
Router(config-bg-bd-pw-evi)# commit
Router# configure
Router(config)# evpn
Router(config-evpn)# virtual neighbor 70.70.70.70 pw-id 17300001
Router(config-evpn-ac-pw)# ethernet-segment
Router(config-evpn-ac-pw-es)# identifier type 0 12.12.00.00.00.01.00.00.03
Router(config-evpn-ac-pw-es)# bgp route-target 1212.8888.0003
Router(config-evpn-ac-pw-es)# exit
Router(config-evpn-ac-pw)# timers peering 15
Router(config-evpn-ac-pw-timers)# commit
```

Step 2 Configure DCI2 with bridge domain and assign EVI 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-bg-bd)# neighbor 70.70.70.70 pw-id 17300001
Router(config-bg-bd-pw)# evi 1
Router(config-bg-bd-pw-evi)# member vni 10001
Router(config-bg-bd-pw-evi)# commit
Router# configure
```

```
Router(config)# evpn
Router(config-evpn)# virtual neighbor 70.70.70.70 pw-id 27300001
Router(config-evpn-ac-pw)# ethernet-segment
Router(config-evpn-ac-pw-es)# identifier type 0 12.12.00.00.00.01.00.00.03
Router(config-evpn-ac-pw-es)# bgp route-target 1212.8888.0003
Router(config-evpn-ac-pw-es)# exit
Router(config-evpn-ac-pw)# timers peering 15
Router(config-evpn-ac-pw-timers)# commit
```

Step 3 Configure EVPN with virtual ethernet segment on both DCI1 and DCI2.

Example:

```
Router(config)# evpn
Router(config-evpn)# virtual neighbor 70.70.70.70 pw-id 27300001
Router(config-evpn-ac-pw)# ethernet-segment
Router(config-evpn-ac-pw-es)# identifier type 0 12.12.00.00.00.01.00.00.03
Router(config-evpn-ac-pw-es)# bgp route-target 1212.8888.0003
Router(config-evpn-ac-pw-es)# exit
Router(config-evpn-ac-pw)# timers peering 15
Router(config-evpn-ac-pw-timers)# commit
```

Step 4 Configure PE3 with bridge domain and assign the virtual ethernet segments of DC1 and DCI2 as neighbors to the bridge domain.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 73
Router(config-l2vpn-bg)# bridge-domain 73-1
Router(config-bg-bd)# neighbor 10.10.10.10 pw-id 17300001
Router(config-bg-bd-pw)# exit
Router(config-bg-bd)# neighbor 20.20.20.20 pw-id 27300001
Router(config-bg-bd)# commit
```

Step 5 Running configuration of virtual Ethernet segment.

Example:

```
/* On DCI1 */

l2vpn
 bridge group bg1
  bridge-domain bd1
   neighbor 70.70.70.70 pw-id 17300001
    evi 1
     member vni 10001
!

evpn
  virtual neighbor 70.70.70.70 pw-id 17300001
   ethernet-segment
    identifier type 0 12.12.00.00.00.01.00.00.03
    bgp route-target 1212.8888.0003
    !
  timers peering 15
!

/* On DCI2 */

l2vpn
 bridge group bg1
  bridge-domain bd1
   neighbor 70.70.70.70 pw-id 27300001
```

```
    evi 1
     member vni 10001
!

evpn
  virtual neighbor 70.70.70.70 pw-id 27300001
   ethernet-segment
    identifier type 0 12.12.00.00.00.01.00.00.03
    bgp route-target 1212.8888.0003
    !
  timers peering 15
!

/* On PE3 */
!
l2vpn
 bridge group bg73
  bridge-domain bd73-1
   neighbor 10.10.10.10 pw-id 17300001
   !
   neighbor 20.20.20.20 pw-id 27300001

!
```

Step 6 Use the **show evpn ethernet-segment** command to to verify the Ethernet segment ID and interface status.

Example:

```
Router# show evpn ethernet-segment
Thu Mar  7 10:56:37.662 UTC

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0012.1200.0000.0100.0003 PW:70.70.70.70,17300001           N/A
RP/0/RP0/CPU0:ios#show evpn ethernet-segment  detail
Thu Mar  7 10:56:53.806 UTC
Legend:
  B   - No Forwarders EVPN-enabled,
  C   - MAC missing (Backbone S-MAC PBB-EVPN / Grouping ES-MAC vES),
  RT  - ES-Import Route Target missing,
  E   - ESI missing,
  H   - Interface handle missing,
  I   - Name (Interface or Virtual Access) missing,
  M   - Interface in Down state,
  O   - BGP End of Download missing,
  P   - Interface already Access Protected,
  Pf  - Interface forced single-homed,
  R   - BGP RID not received,
  S   - Interface in redundancy standby state,
  X   - ESI-extracted MAC Conflict
  SHG - No local split-horizon-group label allocated
  Hp  - Interface blocked on peering complete during HA event
  Rc  - Recovery timer running during peering sequence

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0012.1200.0000.0100.0003 PW:70.70.70.70,17300001           N/A
  ES to BGP Gates   : R
  ES to L2FIB Gates : Ready
  Virtual Access    :
     Name           : PW_70.70.70.70_17300001
     State          : Peering
     Num PW Up      : 0
  ESI ID            : 1
  ESI type          : 0
```

```
   Value          : 0012.1200.0000.0100.0003
ES Import RT      : 1212.8888.0003 (Local)
Source MAC        : 0000.0000.0000 (N/A)
Topology          :
   Operational    : SH
   Configured     : Single-active (AApS) (default)
```

This configuration ensures resilient and loop-free forwarding over the virtual Ethernet segment by establishing access pseudowires on DCI1 and DCI2 and connecting them as neighbors to PE3's bridge domain.

EVPN E-Line with FXC service in VLAN unaware mode

An EVPN E-Line with flexible cross-connect (FXC) service in VLAN unaware mode is a network service that

- aggregates mulitple normalized attachment circuits (ACs) on a single Ethernet segment (ES) destined to a single endpoint or interface into a single EVPN E-Line tunnel represented by one E-Line service ID
- reduces the number of BGP states by advertising one EVI-EAD route per VLAN-unaware FXC instead of per AC, and
- does not signal VLAN failure over BGP, which means that in a multihoming scenario, ES-EAD routes are present, and the EVI can be shared with other VLAN-unaware FXCs or EVPN E-Line.

Table 36: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-Line with FXC service in VLAN unaware mode	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]) You can reduce BGP state complexity and improve scalability with EVPN E-Line in FXC VLAN unaware mode. This mode combines multiple normalized ACs on a single ES into one EVPN E-Line tunnel using a single service ID, advertising one EVI-EAD route per FXC instead of per AC. Additionally, VLAN failures are not signaled over BGP, reducing unnecessary state changes. This feature supports both single-homing and multihoming scenarios, providing flexible and efficient connectivity options.

Scalability challenges of pseudowire services

As service provider networks expand, supporting a growing customer base imposes specific scalability challenges for pseudowire services:

- The number of pseudowires increases proportionally with the customer base, requiring higher capacity from aggregation routers.

- Aggregation routers may become bottlenecks, leading to increased hardware investment to support additional pseudowires.
- Operational costs rise as managing a larger pseudowire infrastructure entails more equipment and ongoing maintenance.

FXC service for pseudowire aggregation

The FXC service mitigates pseudowire scalability challenges by enabling users to select L2 subinterfaces from multiple physical or bundled ports and group them into a single cross-connection group, where all AC members connect to a common pseudowire (PW). This approach reduces the total number of individual PWs required, thereby lowering the capacity demands on aggregation routers and simplifying network management.

Benefits of FXC service

- Allows selecting specific L2 subinterfaces from different physical or bundled ports.
- Consolidates multiple customer traffic streams into a single PW.
- Supports L3 services on individual PWHE subinterfaces.
- Optimizes resource utilization on access and service PE routers.

Limitations of EVPN E-Line with FXC service VLAN unaware mode

- Multihoming is supported on VLAN unaware FXC only if all ACs belong to the same main interface.
- When multiple ESIs exist, whether zero-ESI or non-zero ESI, only ESI 0 is signaled. Therefore, only single-home mode is supported in this scenario.

How traffic flows in FXC service-enabled networks

Service providers use the FXC service to manage complex customer traffic flows, ensuring each customer's data is properly tagged, routed, and delivered regardless of the underlying network architecture.

Summary

The key components involved in the process are:

- Access Provider Edge (A-PE): Runs FXC service, assigns and rewrites VLAN tags, and interfaces with customer devices.
- Service Provider Edge (S-PE): Receives and forwards MPLS labeled PW traffic between the core and A-PE.
- L2 subinterfaces: Serve as the entry and exit points for customer traffic, supporting VLAN tagging operations.
- Normalized VLAN IDs: Unique VLAN identifiers assigned to each customer's L2 sub-interface for consistent traffic mapping.

- PW tunnels: Carry normalized VLAN-tagged traffic across the core MPLS network.

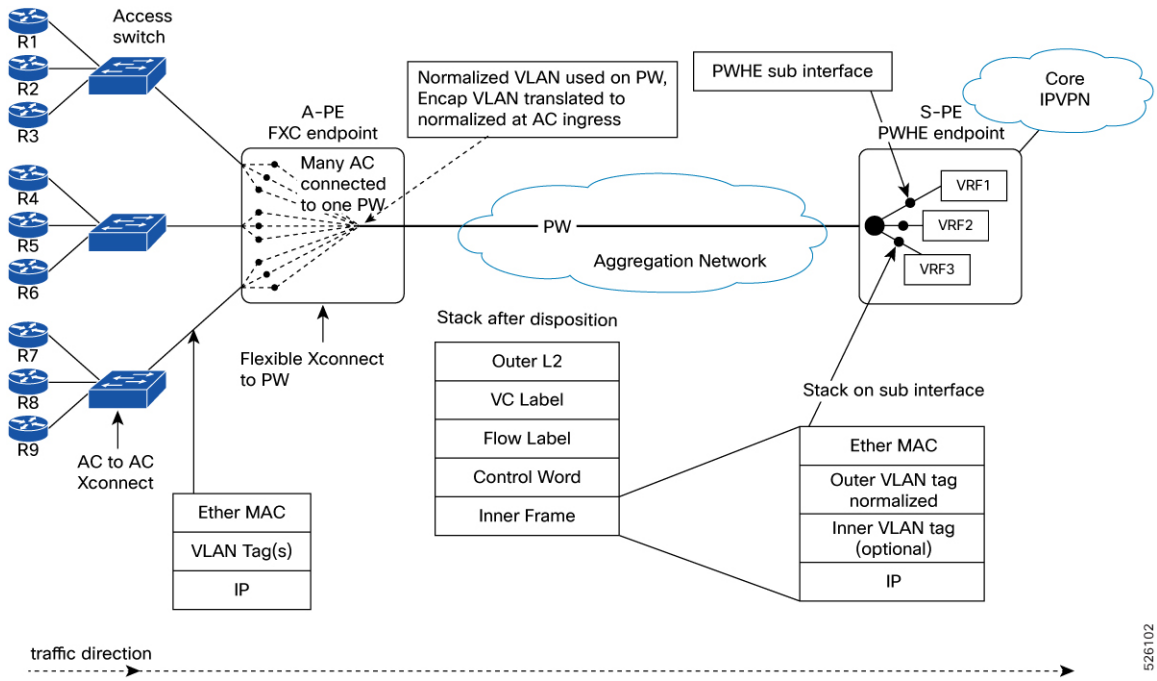
Only a single normalized VLAN tag is supported.

The FXC service manages VLAN tag normalization and mapping to ensure seamless traffic flow between customer-facing sub-interfaces and core MPLS networks, enabling accurate customer traffic identification and delivery.

Workflow

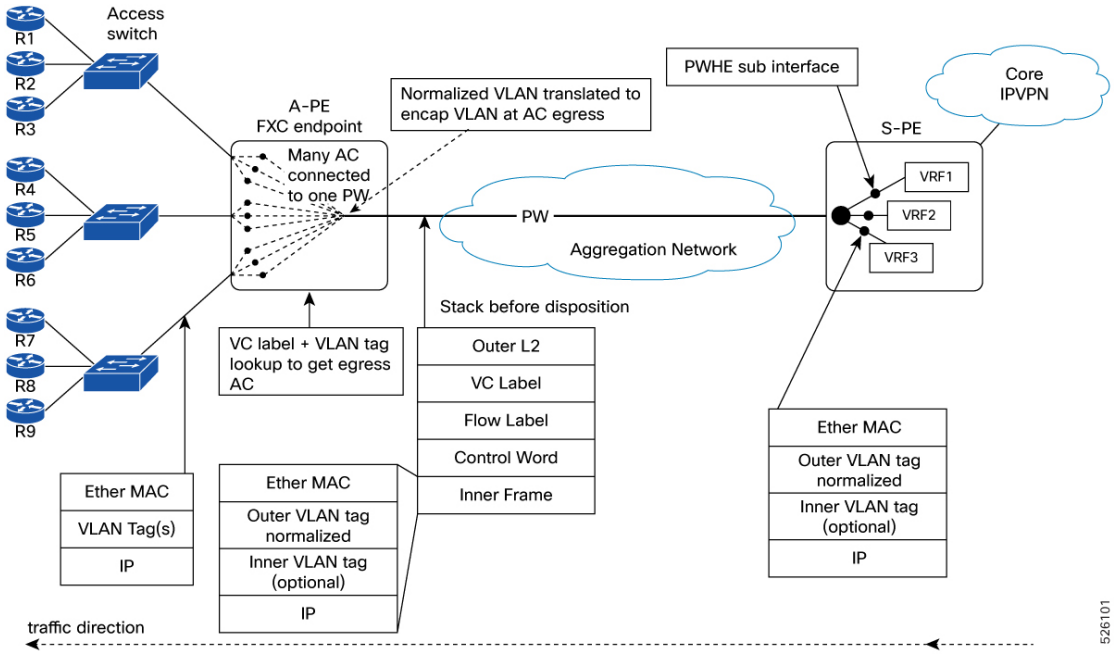
These stages describe how traffic flows in FXC service-enabled networks.

Figure 16: Traffic flow from A-PE to S-PE (access to core)



1. Customer traffic arrives at the access router (A-PE) on a dedicated L2 subinterface, tagged with a customer-specific VLAN.
2. The FXC service at the A-PE rewrites the incoming customer VLAN tag to a unique normalized VLAN ID at the sub-interface level.
3. The normalized VLAN-tagged traffic is forwarded into a PW tunnel towards the PW handling entity (PWHE) on the S-PE.
4. The S-PE receives the traffic and processes it according to the core MPLS network policies.

Figure 17: Traffic flow from S-PE to A-PE (core to access)



5. The S-PE sends MPLS-labeled PW traffic towards the A-PE, with each customer's traffic tagged using the corresponding normalized VLAN ID.
6. At the A-PE, the FXC service maps the normalized VLAN-tagged traffic back to the appropriate customer-facing L2 sub-interface during PW disposition.
7. On the egress L2 subinterface, the normalized VLAN tag is swapped back to the original customer VLAN tag.
8. The traffic is then delivered to the customer device with the correct VLAN tagging restored.

Configure flexible cross-connect service using VLAN-unaware mode

You can configure flexible cross-connect service in VLAN-unaware mode for both single-homed and multihomed EVPN scenarios:

- Configure single-homed flexible cross-connect service using VLAN-unaware mode
- Configure multihomed flexible cross-connect service using VLAN-unaware mode

Configure single-homed flexible cross-connect service using VLAN-unaware mode

Enable L2 connectivity between a single PE router and customer equipment using a single-homed FXC service in VLAN-unaware mode.

This task applies when you need to configure an FXC service that supports single-homing with EVPN control and VLAN-unaware operation on bundle interfaces and subinterfaces (See Figure 1 and Figure 2 for topology reference).

No ESI or ethernet-segment configuration is required for single-homed operation.

Procedure

Step 1 Configure the FXC service in VLAN-unaware mode on the A-PE.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxs1
Router(config-l2vpn-fxs)# interface FourHundredGigE0/0/0/0.1
Router(config-l2vpn-fxs)# interface FourHundredGigE0/0/0/0.2
Router(config-l2vpn-fxs)# interface FourHundredGigE0/0/0/1.1
Router(config-l2vpn-fxs)# neighbor evpn evi 1001 target 1
Router(config-l2vpn-fxs)# commit
```

Step 2 Configure the L2 transport and VLAN translation for each subinterface.

Example:

```
Router# configure
Router(config)# interface FourHundredGigE0/0/0/0.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 11
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 101 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface FourHundredGigE0/0/0/0.2 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 21
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 102 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface FourHundredGigE0/0/0/1.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 11
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 103 symmetric
Router(config-l2vpn-subif)# commit
```

Step 3 Configure the S-PE for pseudowire handling and service mapping.

Example:

```
Router# configure
Router(config)# generic-interface-list GI-LIST
Router(config-generic-if-list)# interface FourHundredGigE0/0/0/10
Router(config-generic-if-list)# exit
Router(config)# interface PW-Ether1
Router(config-if)# attach generic-interface-list GI-LIST
Router(config-if)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group PWHE
Router(config-l2vpn-xc)# p2p PWHE-1
Router(config-l2vpn-xc-p2p)# interface PW-Ether1
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1001 target 1
Router(config-l2vpn-xc-p2p)# commit
```

Step 4 Configure the S-PE subinterfaces for L3 or L2 handoff.

Example:

```
Router# configure
Router(config)# interface PW-Ether1.101
Router(config-subif)# vrf PRIV10
Router(config-subif)# ipv4 address 66.0.0.1 255.255.255.0
```

```
Router(config-subif)# encapsulation dot1q 101
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# interface PW-Ether1.102
Router(config-subif)# vrf PRIV20
Router(config-subif)# ipv4 address 192.1.1.1 255.255.255.0
Router(config-subif)# encapsulation dot1q 102
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# interface PW-Ether1.103
Router(config-subif)# vrf PRIV10
Router(config-subif)# ipv4 address 66.1.110.1 255.255.255.0
Router(config-subif)# encapsulation dot1q 103
Router(config-subif)# commit
```

Step 5 Running configuration of single-homed flexible cross-connect service using VLAN-unaware mode.

Example:

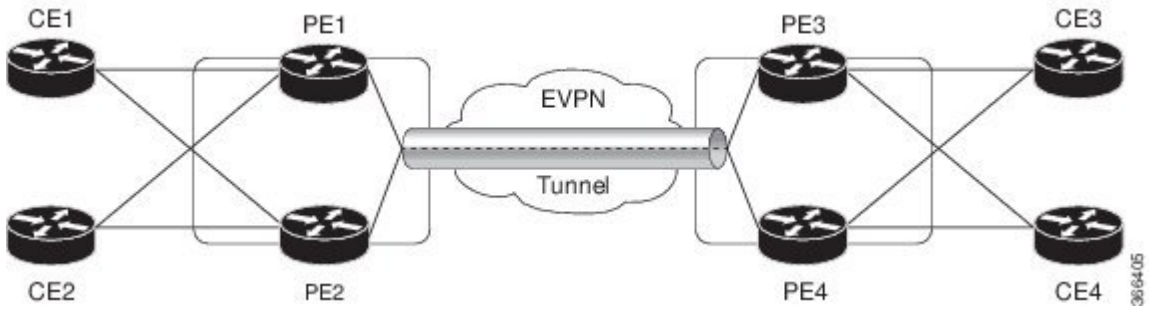
```
/* A-PE configuration */
interface FourHundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 11
 rewrite ingress tag translate 1-to-1 dot1q 101 symetric


interface FourHundredGigE0/0/0/0.2 l2transport
  encapsulation dot1q 21
  rewrite ingress tag translate 1-to-1 dot1q 102 symetric

interface FourHundredGigE0/0/0/1.1 l2transport
  encapsulation dot1q 11
  rewrite ingress tag translate 1-to-1 dot1q 103 symetric

l2vpn
  flexible-xconnect-service vlan-unaware fxs1
   interface FourHundredGigE0/0/0/0.1
   interface FourHundredGigE0/0/0/0.2
   interface FourHundredGigE0/0/0/1.1
   neighbor evpn evi 1001 target 1
!


/* S-PE configuration */
generic-interface-list GI-LIST
 FourHundredGigE0/0/0/10
!
interface PW-Ether1
 attach generic-interface-list GI-LIST

l2vpn
 xconnect group PWHE
  p2p PWHE-1
   interface PW-Ether1
   neighbor evpn evi 1001 target 1
  !

interface PW-Ether1.101
 vrf PRIV10
 ipv4 address 66.0.0.1 255.255.255.0
 encapsulation dot1q 101

interface PW-Ether1.102
 vrf PRIV20
 ipv4 address 192.1.1.1 255.255.255.0
```

```
 encapsulation dot1q 102

interface PW-Ether1.103
 vrf PRIV10
 ipv4 address 66.1.110.1 255.255.255.0
 encapsulation dot1q 103
```

Configure multihomed flexible cross-connect service using VLAN-unaware mode

Enable L2 connectivity across multiple PE routers using a multihomed FXC service in VLAN-unaware mode.

This task applies when you need to configure a FXC service that supports multihoming with EVPN control and VLAN unaware operation on bundle interfaces and subinterfaces.



Before you begin

- Ensure bundle interfaces and subinterfaces are operational.
- Confirm EVPN and L2VPN features are enabled on all PE routers.
- Obtain ESI for each PE router.

Procedure

Step 1 Configure the FXC service in VLAN unaware mode on all PE devices.

Set up each bundle subinterface for L2 transport and configure the EVPN Ethernet segment on the bundle interface.

a) Configure the FXC service in VLAN unaware mode on PE1.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc1_16
Router(config-l2vpn-fxs)# interface Bundle-Ether10.11
Router(config-l2vpn-fxs)# interface Bundle-Ether10.12
Router(config-l2vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether10.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
```

```
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether10.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# evpn
Router (config-evpn)# interface Bundle-Ether10
Router (config-evpn-ac)# ethernet-segment
Router (config-evpn-ac-es)# identifier type  0 00.01.00.ac.ce.55.00.0a.00
Router (config-evpn-ac-es)# commit
```

b) Configure the FXC service in VLAN unaware mode on PE2.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc1_16
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether10.11
Router(config-l2vpn-fxs)# interface Bundle-Ether10.12
Router(config-l2vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether10.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit
Router(config)# interface Bundle-Ether10.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# evpn
Router (config-evpn)# interface Bundle-Ether10
Router (config-evpn-ac)# ethernet-segment
Router (config-evpn-ac-es)# identifier type  0 00.01.00.ac.ce.55.00.0a.00
Router (config-evpn-ac-es)# commit
```

c) Configure the FXC service in VLAN unaware mode on PE3.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc1_16
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether20.11
Router(config-l2vpn-fxs)# interface Bundle-Ether20.12
Router(config-l2vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether20.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-subif)# exit
Router(config)# interface Bundle-Ether20.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
```

```
Router(config-subif)# exit
Router(config)# evpn
Router (config-evpn)# interface Bundle-Ether20
Router (config-evpn-ac)# ethernet-segment
Router (config-evpn-ac-es)# identifier type  0 00.01.00.ac.ce.55.00.14.00
Router (config-evpn-ac-es)# commit
```

d) Configure the FXC service in VLAN unaware mode on PE4.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc1_16
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether20.11
Router(config-l2vpn-fxs)# interface Bundle-Ether20.12
Router(config-l2vpn-fxs)# neighbor evpn evi 1 target 16
Router(config-l2vpn-fxs)# commit
Router(config-l2vpn-fxs)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether20.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 11 symmetric
Router(config-l2vpn-subif)# commit
Router(config-subif)# exit
Router(config)# interface Bundle-Ether20.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
Router(config-l2vpn-subif)# commit
Router(config-subif)# exit
Router(config)# evpn
Router (config-evpn)# interface Bundle-Ether20
Router (config-evpn-ac)# ethernet-segment
Router (config-evpn-ac-es)# identifier type  0 00.01.00.ac.ce.55.00.14.00
Router (config-evpn-ac-es)# commit
```

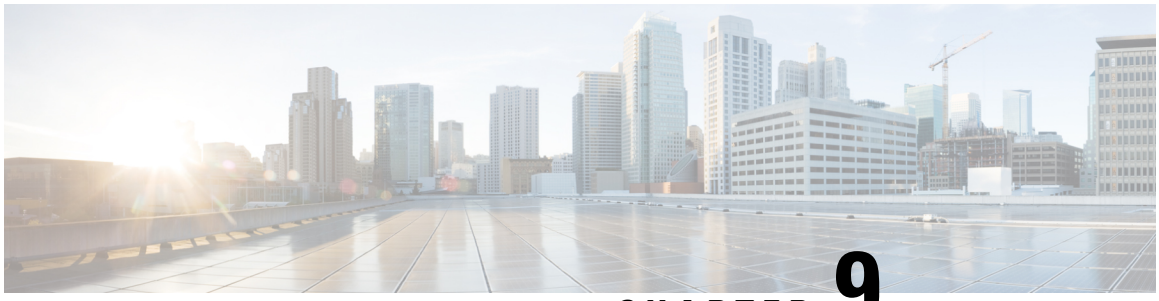
Step 2 Running configuration of multihomed flexible cross-connect service using VLAN-unaware mode.

Example:

```
/* On PE1 */

configure
l2vpn
 flexible-xconnect-service vlan-unaware fxc1_16
  interface Bundle-Ether10.11
  interface Bundle-Ether10.12
  neighbor evpn evi 1 target 16

!

configure
interface Bundle-Ether10.11 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!


configure
interface Bundle-Ether10.12 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric
```

```
!

evpn
  interface Bundle-Ether10
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.0a.00

!

/* On PE2 */

configure
l2vpn
 flexible-xconnect-service vlan-unaware fxc1_16
  interface Bundle-Ether10.11
  interface Bundle-Ether10.12
  neighbor evpn evi 1 target 16

!

configure
interface Bundle-Ether10.11 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!


configure
interface Bundle-Ether10.12 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

!

evpn
  interface Bundle-Ether10
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.0a.00

!

/* On PE3 */

configure
l2vpn
 flexible-xconnect-service vlan-unaware fxc1_16
  interface Bundle-Ether20.11
  interface Bundle-Ether20.12
  neighbor evpn evi 1 target 16

!

configure
interface Bundle-Ether20.11 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether20.12 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

!
```

```
evpn
  interface Bundle-Ether20
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.14.00

!

/* On PE4 */

configure
l2vpn
 flexible-xconnect-service vlan-unaware fxc1_16
  interface Bundle-Ether20.11
  interface Bundle-Ether20.12
  neighbor evpn evi 1 target 16

!

configure
interface Bundle-Ether20.11 l2transport
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 11 symmetric

!

configure
interface Bundle-Ether20.12 l2transport
  encapsulation dot1q 2
  rewrite ingress tag translate 1-to-1 dot1q 12 symmetric

!

evpn
  interface Bundle-Ether20
   ethernet-segment identifier type 0 00.01.00.ac.ce.55.00.14.00

!
```



CHAPTER 9

EVPN E-Tree for EVPN E-LAN

EVPN E-Tree

An EVPN E-Tree is a network architecture that

- segregates traffic to prevent direct communication between remote sites
- reduces network congestion, and
- minimizes the surface area vulnerable to attacks.

Table 37: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-Tree (Scenario 2)	Release 26.2.1	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 E-Tree (Scenario 2)	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) * his feature is now supported on Cisco 8404-SYS-D routers.
EVPN E-Tree (Scenario 1a)	Release 26.2.1	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 E-Tree (Scenario 1a)	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.

Feature Name	Release Information	Feature Description
EVPN E-Tree (Scenario 2)	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN E-Tree (Scenario 1a)	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
EVPN E-Tree (Scenario 2)	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN E-Tree (Scenario 1a)	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN E-Tree (Scenario 2)	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: P100])(select variants only*) *The EVPN E-Tree (Scenario 2) functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN E-Tree (Scenario 1a)	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: P100])(select variants only*) *The EVPN E-Tree (Scenario 1a) functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
EVPN E-Tree (Scenario 2)	Release 24.3.1	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*) ; Modular Systems (8800 [LC ASIC: P100])(select variants only*) *The EVPN E-Tree (Scenario 2) functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN E-Tree (Scenario 1a)	Release 24.3.1	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*) *The EVPN E-Tree (Scenario 1a) functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN E-Tree (Scenario 2)	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) We now enable a PE device to have both root and leaf sites for a given EVI, which increases the granularity of leaf designation from the entire bridge to AC bridge ports; ACs under a bridge may be root or leaf. *This feature is supported on routers with the 88-LC1-36EH line cards.
EVPN E-Tree (Scenario 1a)	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) We now support EVPN E-Tree with route-targets (RT) constraints using two RTs per EVI. This feature prevents L2 communication between the ACs of two or more leafs. *This functinality is now supported on routers with the 88-LC1-36EH line cards.
EVPN E-Tree	Release 7.11.1	We now enable efficient forwarding of ethernet traffic in a tree-like topology where a root PE router broadcasts or multicasts traffic to all the leaf PE routers while the leaf PE routers only forward traffic destined for the respective customer sites connected to them. This feature is supported only on Q200-based line cards. The feature introduces the **etree rt-leaf** command.

Benefits of EVPN E-Tree

EVPN E-Tree provides several benefits that enhance network segmentation and security. These benefits include:

- Enabling hierarchical traffic segregation by defining root and leaf nodes, which restricts communication between leaf nodes while allowing communication between root and leaf nodes.
- Reducing unnecessary traffic flooding across the EVPN E-LAN by limiting leaf-to-leaf communication.
- Enhancing security by preventing direct communication between remote sites or leaf nodes.
- Improving network efficiency and scalability by controlling traffic flows and minimizing broadcast domains.
- Supporting diverse enterprise and service provider networking scenarios that require controlled multi-point connectivity with traffic isolation.

How EVPN E-Tree service works

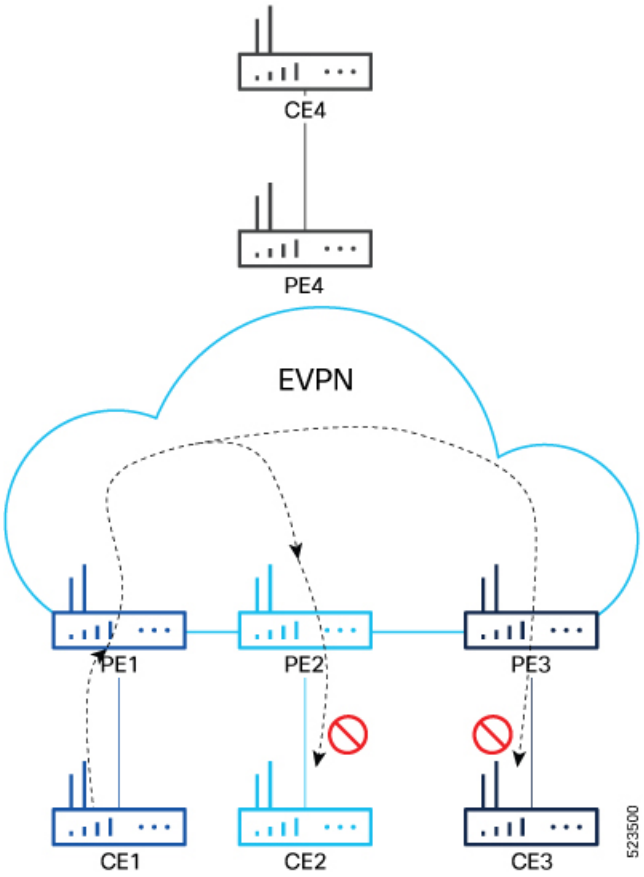
Summary

The key components involved in the EVPN E-Tree service are:

- Root node (PE4): Acts as the central communication hub within the service provider's network. It connects to all leaf nodes and manages traffic forwarding.
- Leaf nodes (PE1, PE2, PE3): Represent customer sites or remote locations. They communicate only with the root node and do not forward traffic to other leaf nodes.
- EVPN protocol: Enables dynamic learning of MAC addresses and supports efficient communication between root and leaf nodes.

The EVPN E-Tree process ensures that leaf nodes communicate only with the root node, preventing direct leaf-to-leaf traffic to enhance network isolation and reduce congestion. Dynamic MAC address learning through EVPN supports efficient traffic forwarding between the root and leaf nodes.

Workflow



These stages describe how EVPN E-Tree service works.

1. ACs are defined as either root or leaf nodes. If a PE is not configured as a leaf, it defaults to root.
2. The root node establishes connections with all leaf nodes, serving as the central point of communication.
3. Leaf nodes send and receive traffic exclusively to and from the root node.
4. The root node forwards traffic to all leaf nodes, ensuring communication is maintained without direct leaf-to-leaf traffic.
5. Direct communication between leaf nodes is restricted to prevent unnecessary traffic and enhance network isolation.
6. EVPN facilitates dynamic MAC address learning across the network, enabling flexible and efficient communication between root and leaf nodes.

Result

This process ensures hierarchical traffic segregation, maintaining communication between the root and leaf nodes while isolating leaf nodes from each other. It reduces unnecessary traffic, enhances security, and improves network efficiency.

E-Tree implementation scenarios

You can implement E-Tree using these scenarios; however, only scenario 1a and scenario 2 are supported:

- Scenario 1: All ACs at a particular PE device for a given Ethernet VPN instance (EVI) or bridge domain (BD) are either root or leaf sites. In this scenario, all traffic for an EVI from a PE in the network originates from either a root or a leaf site. You have two configuration options:
  - Scenario 1a: Configure E-Tree with route-target (RT) constraints using two RTs per EVI.
  - Scenario 1b: Configure E-Tree without RT constraints by using the E-Tree leaf label.
- Scenario 2: Configure a PE device to have both root and leaf sites for a given EVI.

E-Tree scenario 1a

EVPN E-Tree using route-target (RT) constraints allows you to configure BGP RT import and export policies for each AC. This configuration defines the communication pattern between root and leaf nodes within a BD. Layer 2 L2 traffic can flow only between root and leaf nodes—either from root to leaf or leaf to root—for a given BD. Direct L2 communication between leaf ACs is prevented, ensuring traffic isolation among leaf sites. Each EVI uses two distinct BGP RTs: one set associated with root ACs and another set associated with leaf ACs. This separation enforces the communication restrictions and maintains the E-Tree service topology.

BGP import and export policies for EVPN EVI instances

In EVPN EVI instances, BGP import and export policies govern the distribution of RTs to control Layer 2 communication between root and leaf PE devices.

- Root PE behavior:
  - Exports its ROOT-RT using the BGP export policy.
  - Imports other ROOT-RTs from corresponding root PEs for the same EVI.

    This aspect is essential in scenarios with multiple roots for a BD and EVPN EVI, such as multihome active-active, port-active, and single-active modes.
  - Imports LEAF-RT through the BGP import policy. This allows the root PE to learn all remote L2 MAC addresses advertised by leaf PEs through EVPN RT2.
- Leaf PE behavior:
  - Exports its LEAF-RT using the BGP export policy, and informs the root PE of the reachability of its directly connected L2 endpoints through EVPN RT2.
  - Imports ROOT-RT using the BGP import policy, which enables the leaf PE to discover L2 endpoints reachable through the root PE's AC under the EVPN EVI instance.
  - Must not import LEAF-RT to prevent L2 communication between leaf PEs.
- General policy application:
  - These BGP import and export policies apply to all EVPN RTs, including RT2 advertisements, ensuring proper isolation and communication paths within the E-Tree topology.

This policy framework enforces the E-Tree service model by allowing L2 traffic only between root and leaf nodes, preventing direct leaf-to-leaf communication, and supporting multi-root scenarios for enhanced redundancy and scalability.

MAC address learning in EVPN E-Tree scenario 1a

- L2 MAC addresses are learned on the AC of a specific BD on a leaf PE device as type LOCAL.
- The leaf PE advertises the learned MAC address to the root PE as an EVPN route target 2 (RT2).
- On the remote root PE, the MAC address entry is replicated in the MAC table with the learning type L2VPN.
- The root PE associates the MAC entry with the MPLS label of its BGP peer that advertises RT2 to the root PE node.
- Similarly, L2 MAC addresses learned on the AC of a BD on the root PE are marked as type LOCAL.
- The root PE advertises these MAC addresses to peer root or leaf PEs as EVPN RT2.
- On the remote root or leaf PE, the MAC table replicates the MAC entry with learning type L2VPN and associates it with the MPLS label of the BGP peer advertising RT2.
- For root PEs, the MAC table of a peer root node synchronizes the replicated MAC entry with learning type L2VPN for the same Ethernet Segment Identifier (ESI) and uses the same AC as the next hop.
- This synchronization prevents flooding and duplication of known unicast traffic, ensuring efficient MAC address learning and forwarding within the EVPN E-Tree topology.

Configure EVPN E-Tree scenario 1a

Configure EVPN E-Tree with route-target constraints on a PE device to enable root-to-leaf communication while isolating leaf-to-leaf traffic.

EVPN E-Tree Scenario 1a uses BGP route-target import and export policies to define ACs as root or leaf nodes within a BD and EVI. This configuration supports dynamic MAC learning and traffic filtering between root and leaf sites.

Procedure

Step 1 Configure 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 Bundle-Ether700.305
Router (config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface Bundle-Ether720.305
Router (config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit
```

Step 2 Configure attachment circuits.

Example:

```
Router# configure
Router(config)# interface Bundle-Ether700.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether720.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit
```

Step 3 Configure the EVPN EVI.

Example:

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target import 1001:305>>  Route target of leaf
Router(config-evpn-instance-bgp)# route-target export 1001:5305>> Route target of root
Router(config-evpn-instance-bgp)# exit
Router(config-evpn-instance)# exit
Router(config-evpn-instance)# etree
Router(config-evpn-instance-etree)# rt-leaf
Router(config-evpn-instance-etree)# exit
Router(config-evpn-instance)# control-word-disable
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit
```

Step 4 Configure bundle Ethernet interfaces.

Example:

```
Router# configure
Router(config)# interface Bundle-Ether700
Router(config-if)# lacp system mac 00aa.aabb.1010
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# exit
Router(config)# interface Bundle-Ether720
Router(config-if)# lacp system mac 00aa.aabb.1212
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit
```

Step 5 Configure EVPN interfaces.

Example:

```
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether700
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.00.00.00.00.00.00.00.00
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0001
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# exit
Router(config-evpn)# exit
Router(config-evpn)# interface Bundle-Ether720
```

```
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.00.00.00.00.00.00.00.00
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0020
Router(config-evpn-ac-es)# commit
```

Step 6 Running configuration of EVPN E-Tree scenario 1a.

Example:

```
l2vpn
 bridge group BG1
  bridge-domain BD1
   interface Bundle-Ether700.305
   !
   interface Bundle-Ether720.305

   !
   evi 305
   !
  !
 !
interface Bundle-Ether700.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether720.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
evpn
 evi 305
  bgp
   route-target import 1001:305
   route-target export 1001:5305
  !
  etree
   rt-leaf
  !
  control-word-disable
  advertise-mac
  !
 !
!
interface Bundle-Ether700
 lacp system mac 00aa.aabb.1010
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether720
 lacp system mac 00aa.aabb.1212
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether700
  ethernet-segment
   identifier type 0 00.00.00.00.00.00.00.00.00
   bgp route-target 0000.0000.0001
  !
 !
!
evpn
 interface Bundle-Ether720
```

```
  ethernet-segment
   identifier type 0 00.00.00.00.00.00.00.00.00
   bgp route-target 0000.0000.0020
  !
 !
!
```

Step 7 Use **show l2route evpn mac all** command to verify the EVPN E-Tree scenario 1a configuration.

The single-homing PE device only knows about its local L2 MAC addresses and the MAC addresses learned on the root node. The leaf single-homing PE device does not know any other MAC addresses learned on other leaf PE nodes. Each leaf is completely isolated from other leaf PEs in terms of their knowledge of MAC addresses learned from each other.

Example:

```
Router$ show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- ---------------------------------------
200      0011.0100.0001 L2VPN       30579/I/ME, N/A
200      0011.0100.0002 L2VPN       30579/I/ME, N/A
200      0011.0100.0003 L2VPN       30579/I/ME, N/A
200      0011.0100.0004 L2VPN       30579/I/ME, N/A
200      0011.0100.0005 L2VPN       30579/I/ME, N/A
200      0012.0100.0001 LOCAL       Bundle-Ether700.305, N/A
200      0012.0100.0002 LOCAL       Bundle-Ether700.305, N/A
200      0012.0100.0003 LOCAL       Bundle-Ether700.305, N/A
200      0012.0100.0004 LOCAL       Bundle-Ether700.305, N/A
200      0012.0100.0005 LOCAL       Bundle-Ether700.305, N/A
```

The PE device is configured for EVPN E-Tree scenario 1a, enabling root-to-leaf Layer 2 communication with isolation between leaf nodes. The device learns local MAC addresses and those learned on the root node, ensuring proper MAC address distribution and traffic filtering.

E-Tree scenario 2

Customer sites represented by ACs can be designated as either root or leaf nodes. For a given EVI, a PE device may receive traffic from both root and leaf ACs originating from a remote node. An EVI can be associated with both root and leaf sites simultaneously. If an AC is not explicitly configured as a leaf in the E-Tree topology, it defaults to being a root.

A PE device supports having both root and leaf sites within the same EVI. This scenario provides finer granularity by allowing root or leaf designation at the individual AC level, as opposed to scenario 1 where the designation is at the bridge domain level. Consequently, traffic for an EVI from a PE can originate from either root or leaf sites, enabling flexible and granular traffic segregation within the network.

Unicast traffic behavior in EVPN E-Tree scenario 2

- Remote PE devices perform ingress filtering to avoid unnecessary traffic traversing the core network only to be filtered at the egress PE.
- Each PE marks MAC addresses to indicate whether they are associated with a root or a leaf node.
- MAC address advertisements from leaf sites include a leaf-indication flag within an extended community attribute; routes lacking this flag originate from root sites.

- When remote PEs program MAC addresses with the leaf-indication flag, they cross-check the originating AC. If the AC is also a leaf, packets are not forwarded, preventing leaf-to-leaf traffic.
- The solution supports E-Tree extended community type 0x06 (EVPN) with sub-type 0x05 for leaf-indication on both known unicast and BUM traffic.
- Unknown unicast suppression should be enabled on EVIs connected to both root and leaf sites. This prevents unknown unicast traffic arriving at an EVI from being flooded to ACs, effectively eliminating leaf-to-leaf traffic during bridge domain MAC flush events.
- MAC addresses advertise the local ESI and do not include a leaf indicator when originating from root nodes.
- During processing of root synchronization routes, the root or leaf status is evaluated at the individual AC level rather than the entire bridge domain. If a root MAC with a matching local ESI is received but the corresponding AC is configured as a leaf, a syslog message is generated to indicate a misconfiguration.

BUM traffic behavior in EVPN E-Tree scenario 2

- PE devices perform egress filtering on BUM traffic. BUM traffic originating from leaf sites is filtered at the egress PE if the destination is also a leaf, preventing leaf-to-leaf BUM traffic.
- A PE with leaf sites assigns a leaf label and advertises this label to remote PEs through an Ethernet Segment/EVPN Auto-Discovery (ES/EAD) route with ESI set to 0, including the E-TREE extended community.
- BUM traffic handling based on AC type and homing:
  - Single-homed leaf AC: BUM traffic is tagged with the destination ETREE leaf label.
  - Single-homed root AC: BUM traffic is not tagged with any ESI or ETREE leaf label.
  - Multi-homed leaf AC: BUM traffic is tagged with the destination ETREE leaf label.
  - Multi-homed root AC: BUM traffic is tagged with the ESI label.
- The ingress PE tags MPLS frames originating from leaf sites with the ETREE leaf label. This label enables the egress PE to perform filtering based on native EVPN ESI label, ensuring proper BUM traffic segregation.
- To prevent intra-PE forwarding between leaf sites, all leaf ACs within a bridge domain are placed in a single split-horizon group, effectively isolating leaf-to-leaf BUM traffic within the same PE.

Configure EVPN E-Tree scenario 2

Configure EVPN E-Tree on a PE device to establish root and leaf bridge domains for Ethernet VPN services.

This task applies when you need to set up EVPN E-Tree scenario 2 on a PE device, involving bridge domain configuration, E-Tree interface roles, and EVI setup.

Procedure

Step 1 Configure the bridge domain.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd_1
```

Step 2 Configure E-Tree interfaces.

Example:

```
Router(config-l2vpn-bg-bd)# interface Bundle-Ether400
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# # interface Bundle-Ether401.1001
Router(config-l2vpn-bg-bd-ac)# etree
Router(config-l2vpn-bg-bd-ac-etree)# leaf
```

Step 3 Configure the EVI.

Example:

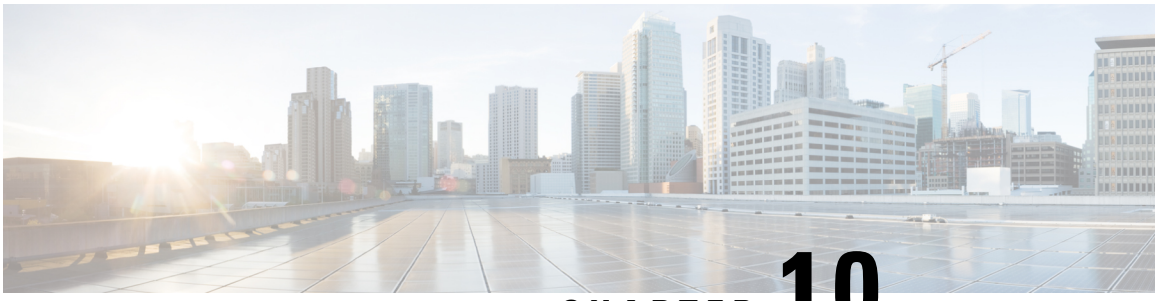
```
Router(config-l2vpn-bg-bd)# evi 200
Router(config-l2vpn-bg-bd-evi)# commit
```

Step 4 Running configuration of EVPN E-Tree scenario 2.

Example:

```
/* Configuration for root and leaf */

l2vpn
 bridge group bg1
  bridge-domain bd_1
   interface Bundle-Ether400.1
   !
   interface Bundle-Ether401.1001
   !
   interface Bundle-Ether4701.2001
   !
   etree
    leaf
   !
   evi 200
   !
  !
 !
```

The PE device is configured with EVPN E-Tree scenario 2, establishing root and leaf interfaces within the specified bridge domain and associating them with the EVI.



CHAPTER 10

CFM on EVPN

CFM on EVPN

An Ethernet Connectivity Fault Management (CFM) is an Ethernet layer operation, administration, and management (OAM) protocol that

- operates end-to-end per service instance
- provides proactive connectivity monitoring and fault verification, and
- enables fault isolation in large networks.

Table 38: Feature History Table

Feature Name	Release Information	Feature Description
Extended Hardware offload of Connectivity Check Messages (CCM) timers	Release 26.2.1	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 EVPN	Release 26.2.1	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 EVPN	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Feature Name	Release Information	Feature Description
CFM on EVPN	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
CFM on EVPN	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) *The CFM on EVPN functionality is now extended to the Cisco 8712-MOD-M routers.
CFM on EVPN	Release 24.3.1	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*) * The CFM on EVPN functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
CFM on EVPN	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: Q200, P100]) (select variants only*) You can now proactively monitor connectivity and verify faults and isolate them for EVPN services. This is because Ethernet Connectivity Fault Management (CFM) is now available for EVPN and provides end-to-end service level OAM (Operations, Administration, and Maintenance) for EVPN services. *This feature is supported only on routers with Q200 and 88-LC1-36EH line cards.

CFM on EVPN feature highlights and benefits

CFM on EVPN provides essential capabilities to monitor and maintain the health and connectivity of EVPN services. The key highlights and benefits include:

- Fault detection and isolation—enables rapid detection of connectivity faults within EVPN domains, helping to isolate and pinpoint issues quickly to minimize service disruption.
- Proactive monitoring—supports continuous monitoring of EVPN service health through periodic continuity checks, ensuring early identification of potential problems.
- Service assurance—verifies the operational status of EVPN instances, CFM helps maintain high service availability and reliability.
- Standardized protocols—uses IEEE 802.1ag and ITU-T Y.1731 standards, ensuring interoperability across multi-vendor environments.

- Scalability—designed to scale with large EVPN deployments, supporting multiple maintenance domains and endpoints.
- Simplified troubleshooting—provides detailed fault management tools such as loopback and linktrace messages, which assist network operators in diagnosing and resolving issues efficiently.
- Integration with EVPN— is tightly integrated with EVPN control plane mechanisms, enabling seamless operation and management within EVPN architectures.

Supported offload types and timer values

Continuity Check Messages (CCMs) are heartbeat messages exchanged periodically among all Maintenance End Points (MEPs) within a service. Each MEP sends multicast CCMs and receives CCMs from all other MEPs, enabling peer discovery and connectivity verification. The offload type is determined by where CCMs are processed.

Currently, only the Non-offload type is supported, where CCMs are generated and processed by the CPU. For a CFM session on a bundle interface or similar, CCM timers must be set to one second or greater.

CCM timers define the intervals at which CCMs are sent and received. If CCMs are not received within the configured interval, the CFM MEP is considered down. The supported CCM timer values for the Non-offload type are:

This table details the supported CCM times values for non-offload and hardware offload types:

Offload type	CCM timer value
Non-offload	1 second
	10 seconds
	1 minute
	10 minutes
1 second, applicable to these routers: • Cisco 8011-4G24Y4H-I • Cisco 8011-32Y8L2H2FH	

Restrictions for CFM on EVPN

To ensure accurate CFM operation on EVPN, observe these restrictions:

- Loopback and linktrace results may show artifacts such as multiple or varying responses for the same instance.
- Do not configure CFM on interfaces with untagged encapsulation for EVPN pseudowire, as this is unsupported by Cisco IOS XR software.

Supported services for CFM on EVPN

CFM on EVPN support these service types:

- EVPN E-LAN—CFM plays a crucial role in EVPN E-LAN services by enabling service providers and enterprises to maintain high availability and ensure the reliability of their distributed Ethernet services.
- EVPN E-Line—CFM facilitates monitoring and maintaining the health of point-to-point Ethernet services over packet-switched networks, which is essential for service providers managing EVPN E-Line services.

Note CFM on EVPN is supported only on single-homed devices.

CFM on EVPN E-LAN single-homing

CFM on EVPN E-LAN single-homing is a network monitoring feature that

- operates on networks running E-LAN services with single-homed devices
- monitors the health and performance of E-LAN services, and
- provides high-speed Layer 2 services with enhanced resiliency.

How CFM on EVPN E-LAN single-homing works

Summary

The key components involved in the CFM on EVPN E-LAN single-homing process are:

- Single-homed device: Connects to the EVPN E-LAN service and participates in connectivity monitoring.
- CFM: Monitors and verifies connectivity within the EVPN E-LAN domain.
- EVPN E-LAN domain: The network environment where devices communicate and connectivity is managed.

The CFM on EVPN E-LAN single-homing process enables continuous connectivity monitoring by the single-homed device within the EVPN E-LAN domain, and detects faults through continuity checks and alerts administrators to ensure network stability and simplify troubleshooting.

Workflow

These stages describe how CFM on EVPN E-LAN single-homing works.

1. The single-homed device connects to the EVPN E-LAN service.
2. CFM is enabled on the device to monitor connectivity within the EVPN E-LAN domain.
3. CFM initiates continuity checks between the single-homed device and other devices in the EVPN E-LAN.
4. CFM detects any connectivity faults and generates alarms or notifications.
5. Network administrators use CFM results to troubleshoot and resolve connectivity issues.

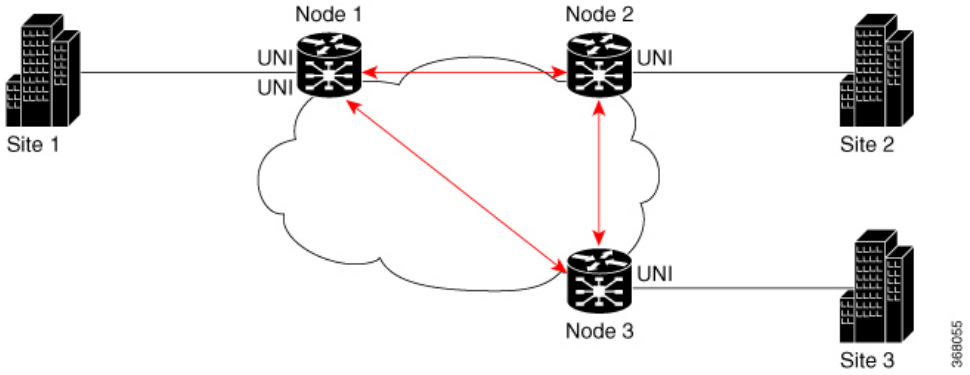
Result

This process ensures continuous monitoring and fault detection in single-homed EVPN E-LAN environments, improving network stability and simplifying troubleshooting.

Configure CFM on EVPN E-LAN full mesh topology

Enable and verify CFM on EVPN E-LAN full mesh topology to monitor network continuity and detect faults.

Use this task to configure CFM continuity checks, MEP cross-checks, and interface settings on Cisco routers participating in an EVPN E-LAN full mesh topology.



Procedure

Step 1 Enable Ethernet CFM and configure the domain and service.

Example:

```
Router# ethernet cfm
Router(config-cfm# domain bd-domain level 1 id null
Router(config-cfm-dmn)# service bd-domain bridge group bg-elan bridge-domain bd-elan id icc-based MC
 MCMC
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Step 2 Configure MEP cross-checks.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-dmn-svc)# mep-id 1112
Router(config-cfm-dmn-svc)# mep-id 1113
Router(config-cfm-dmn-svc)# commit
```

Step 3 Repeat steps 1 and 2 on nodes 2 and 3, adjusting MEP IDs accordingly.

For node 2, configure MEP cross-check with respective mep-id values of node 1 and node 3 (1111 and 1113 respectively, in this example). For node 3, configure MEP cross-check with respective mep-id values of node 1 and node 2 (1111 and 1112 respectively, in this example).

Step 4 Enable CFM on the interface.

Example:

```
Router(config)# interface TenGigE 0/0/0/2.100 l2transport
Router(config-subif)# description bg-elan
```

```
Router(config-subif)# encapsulation dot1q 100
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain bd-domain service bd-service mep-id 1111
Router(config-if-cfm-mep)# commit
```

Step 5 Repeat step 4 on nodes 2 and 3 with their respective MEP IDs.

You must repeat the above configurations for node 2 and node 3, with the respective *mep-id* values (that is, 1112 for node 2 and 1113 for node 3, in this example).

Step 6 Running configuration of CFM on EVPN E-LAN full mesh topology.

Example:

```
ethernet cfm
 domain bd-domain level 1 id null
  service bd-domain bridge group bg-elan bridge-domain bd-elan id icc-based MC MCMC
   continuity-check interval 10s
   mep crosscheck
    mep-id 1112
    mep-id 1113
   !
  !
 !
!

interface TenGigE 0/0/0/2.100 l2transport
 description bg-elan
 encapsulation dot1q 100
 rewrite ingress tag pop 1 symmetric
 ethernet cfm
  mep domain bd-domain service bd-service mep-id 1111
 !
```

CFM is enabled on all nodes in the EVPN E-LAN full mesh topology, providing continuous connectivity monitoring and fault detection.

Configure CFM on EVPN E-LAN hub and spoke topology

Configure CFM on an EVPN E-LAN hub and spoke topology with SLA profiles to monitor connectivity between the hub and spoke nodes.

This task extends the full mesh CFM configuration by adding SLA operation profiles on the hub node to ensure continuous network service monitoring.

Before you begin

Complete the full mesh CFM configuration steps on all nodes.

Procedure

Step 1 Enable Ethernet CFM with the MEP domain, service, and MEP ID, and configure SLA profiles for spoke nodes' MEP IDs.

The CFM configuration for the hub and spoke topology remains the same as that of full mesh topology mentioned above, except for these additional steps for SLA profile configuration to be done under the interface. You must configure the SLA profile between the hub and the spokes to ensure continuous network services.

In this example, 1112 and 1113 are the mep-id values of node 2 and node 3.

Example:

```
Router(config)#interface TenGigE 0/0/0/2.100 l2transport
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain bd-domain service bd-service mep-id 1111
Router(config-if-cfm-mep)# sla operation profile test-profile1 target mep-id 1112
Router(config-if-cfm-mep)# sla operation profile test-profile2 target mep-id 1112
Router(config-if-cfm-mep)# sla operation profile test-profile1 target mep-id 1113
Router(config-if-cfm-mep)# sla operation profile test-profile2 target mep-id 1113
Router(config-if-cfm-mep)# commit
```

Step 2 Running configuration of CFM on EVPN E-LAN hub and spoke topology.

Example:

```
interface TenGigE 0/0/0/2.100 l2transport
 description bg-elan
 encapsulation dot1q 100
 rewrite ingress tag pop 1 symmetric
 ethernet cfm
  mep domain bd-domain service bd-service mep-id 1111
   sla operation profile test-profile1 target mep-id 1112
   sla operation profile test-profile2 target mep-id 1112
   sla operation profile test-profile1 target mep-id 1113
   sla operation profile test-profile2 target mep-id 1113
 !
```

SLA profiles are configured on the hub node, enabling continuous service-level monitoring between the hub and spoke nodes.

Configure CFM for different domain types and bridge domains

Configure Ethernet CFM continuity checks and MEP cross-checks for various domain levels, multiple MEPs, and multiple services across different EVPN bridge domains.

Use these examples to configure CFM in complex EVPN environments with multiple bridge domains, domain levels, and services.

Before you begin

Ensure you have the necessary domain, service, bridge group, and bridge domain information.

Procedure

Follow these steps for each of these scenarios.

Scenario 1: Up MEPs with the same domain and level

Procedure

Step 1 Enable Ethernet CFM continuity check.

Example:

```
Router(config)# ethernet cfm
Router(config-cfm)# domain BD-DOMAIN level 4 id null
Router(config-cfm-dmn)# service BD-SERVICE bridge group ELAN_FUNC_3 $
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Step 2 Configure MEP cross-check.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-xcheck)# mep-id 5
Router(config-cfm-xcheck)# mep-id 1101
Router(config-cfm-xcheck)# mep-id 1103
```

Step 3 Configure the bridge domain and assign interfaces to the bridge domain.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group ELAN_FUNC_3
Router(config-l2vpn-bg)# bridge-domain FUNC_3
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/0.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/1.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evi 101
```

Step 4 Enable CFM on interfaces with respective MEP IDs.

Example:

```
Router(config)# interface TenGigE0/0/0/0.1 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain BD-DOMAIN service BD-SERVICE mep-id 1103
Router(config-if-cfm)# root
Router(config)# interface TenGigE0/0/0/1.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain BD-DOMAIN service BD-SERVICE mep-id 5
```

Step 5 Running configuration of scenario 1.

Example:

```
ethernet cfm
 domain BD-DOMAIN level 4 id null
  service BD-SERVICE bridge group ELAN_FUNC_3 bridge-domain FUNC_3 id number 100
   continuity-check interval 10s
   mep crosscheck
    mep-id 5
    mep-id 1101
    mep-id 1103
```

```
l2vpn
 bridge group ELAN_FUNC_3
  bridge-domain FUNC_3
   interface TenGigE0/0/0/0.1
   !
   interface TenGigE0/0/0/1.2
   !
   evi 101

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain BD-DOMAIN service BD-SERVICE mep-id 1103

interface TenGigE0/0/0/1.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain BD-DOMAIN service BD-SERVICE mep-id 5
```

Scenario 2: Multiple Up MEPs on AC interfaces in the same bridge domain

Procedure

Step 1 Enable Ethernet CFM continuity check.

Example:

```
Router(config)# ethernet cfm
Router(config-cfm)# domain BD-DOMAIN level 4 id null
Router(config-cfm-dmn)# service BD-SERVICE bridge group ELAN_FUNC_3 bridge-domain FUNC_3 $
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Step 2 Configure MEP cross-check.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-xcheck)# mep-id 1
Router(config-cfm-xcheck)# mep-id 2
Router(config-cfm-xcheck)# mep-id 21
Router(config-cfm-xcheck)# mep-id 22
```

Step 3 Enable Ethernet CFM continuity check for another domain level.

Example:

```
Router(config)# ethernet cfm
Router(config-cfm)# domain BD-DOMAIN1 level 3 id null
Router(config-cfm-dmn)# service BD-SERVICE1 bridge group ELAN_FUNC_3 bridge-domain FUNC_3 $
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Step 4 Configure MEP cross-check.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-xcheck)# mep-id 1001
Router(config-cfm-xcheck)# mep-id 1021
```

```
Router(config-cfm-xcheck)# mep-id 2001
Router(config-cfm-xcheck)# mep-id 2021
```

Step 5 Configure the bridge domain and assign interfaces to the bridge domain.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group ELAN_FUNC_3
Router(config-l2vpn-bg)# bridge-domain FUNC_3
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evi 101
```

Step 6 Enable CFM on interfaces with multiple MEPs for both domains.

Example:

```
Router(config)# interface TenGigE0/0/0/0.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain BD-DOMAIN service BD-SERVICE mep-id 21
Router(config-if-cfm-mep)# exit
Router(config-if-cfm)# mep domain BD-DOMAIN1 service BD-SERVICE1 mep-id 1021
Router(config-if-cfm)# root
Router(config)# interface TenGigE0/0/0/1.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain BD-DOMAIN service BD-SERVICE mep-id 22
Router(config-if-cfm-mep)# exit
Router(config-if-cfm)#mep domain BD-DOMAIN1 service BD-SERVICE1 mep-id 2021
```

Step 7 Running configuration of scenario 2.

Example:

```
ethernet cfm
 domain BD-DOMAIN level 4 id null
    service BD-SERVICE bridge group ELAN_FUNC_3 bridge-domain FUNC_3 id number 100
   continuity-check interval 10s
   mep crosscheck
    mep-id 1
    mep-id 2
    mep-id 21
    mep-id 22

 domain BD-DOMAIN1 level 3 id null
  service BD-SERVICE1 bridge group ELAN_FUNC_3 bridge-domain FUNC_3
   continuity-check interval 10s
   mep crosscheck
    mep-id 1001
    mep-id 1021
    mep-id 2001
    mep-id 2021


l2vpn
 bridge group ELAN_FUNC_3
  bridge-domain FUNC_3
   interface TenGigE0/0/0/0
   interface TenGigE0/0/0/1
   Interface TenGigE0/0/0/2
```

```
   evi 101

 interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 1
  ethernet cfm
    mep domain BD-DOMAIN service BD-SERVICE mep-id 21
    mep domain BD-DOMAIN1 service BD-SERVICE1 mep-id 1021


interface TenGigE0/0/0/1.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain BD-DOMAIN service BD-SERVICE mep-id 22
  mep domain BD-DOMAIN1 service BD-SERVICE1 mep-id 2021
```

Scenario 3: Multiple services for different EVPN bridge domains

Procedure

Step 1 Enable Ethernet CFM continuity check.

Example:

```
Router(config)# ethernet cfm
Router(config-cfm)#  domain evpn-bd level 4 id null
Router(config-cfm-dmn)# service evpn-bd1 bridge group BG1 bridge-domain BD1 $
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Step 2 Configure MEP cross-check.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-xcheck)# mep-id 5
Router(config-cfm-xcheck)# mep-id 6
Router(config-cfm-xcheck)# mep-id 1101
Router(config-cfm-xcheck)# mep-id 1103
```

Step 3 Enable Ethernet CFM continuity check for another service.

Example:

```
Router(config)# ethernet cfm
Router(config-cfm)#  domain evpn-bd level 4 id null
Router(config-cfm-dmn)# service evpn-bd2 bridge group BG2 bridge-domain BD2 $
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Step 4 Configure MEP cross-check.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-xcheck)# mep-id 11
Router(config-cfm-xcheck)# mep-id 21
Router(config-cfm-xcheck)# mep-id 101
```

Step 5 Configure the bridge domain and assign interfaces to the bridge domain.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/0.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/1.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evi 101
```

Step 6 Configure another bridge domain and assign interfaces to the bridge domain.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG2
Router(config-l2vpn-bg)# bridge-domain BD2
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/2.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/5.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evi 201
```

Step 7 Enable CFM on interfaces with corresponding domain, service, and MEP IDs.

Example:

```
Router(config)# interface TenGigE0/0/0/0.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain evpn-bd service evpn-bd1 mep-id 1103
Router(config-if-cfm)# root
Router(config)# interface TenGigE0/0/0/1.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain evpn-bd service evpn-bd1 mep-id 5
Router(config-if-cfm)# root
Router(config)# interface TenGigE0/0/0/2.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain evpn-bd service evpn-bd2 mep-id 101
Router(config-if-cfm)# root
Router(config)# interface TenGigE0/0/0/5.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain evpn-bd service evpn-bd2 mep-id 11
```

Step 8 Running configuration of scenario 3.

Example:

```
ethernet cfm
 domain evpn-bd level 4 id null
  service evpn-bd1 bridge group BG1 bridge-domain BD1
   continuity-check interval 10s
   mep crosscheck
    mep-id 5
    mep-id 6
    mep-id 1101
    mep-id 1103
  service evpn-bd2 bridge group BG2 bridge-domain BD2
   continuity-check interval 10s
   mep crosscheck
    mep-id 11
    mep-id 21
```

```
    mep-id 101

l2vpn
 bridge group BG1
  bridge-domain BD1
   interface TenGigE0/0/0/0.1
   interface TenGigE0/0/0/1.2
   evi 101
bridge group BG2
  bridge-domain BD2
   interface TenGigE0/0/0/2.1
   interface TenGigE0/0/0/5.2
   evi 201

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain evpn-bd service evpn-bd1 mep-id 1103

interface TenGigE0/0/0/1.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain evpn-bd service evpn-bd1 mep-id 5

interface TenGigE0/0/0/2.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain evpn-bd service evpn-bd2 mep-id 101

interface TenGigE0/0/0/5.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain evpn-bd service evpn-bd2 mep-id 11
```

Scenario 4: Different EVPN bridge domains on different domain levels

Procedure

Step 1 Enable Ethernet CFM continuity check.

Example:

```
Router(config)# ethernet cfm
Router(config-cfm)# domain evpn-bd level 4 id null
Router(config-cfm-dmn)# service evpn-bd1 bridge group BG1 bridge-domain BD1 $
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Step 2 Configure MEP cross-check.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-xcheck)# mep-id 5
Router(config-cfm-xcheck)# mep-id 6
Router(config-cfm-xcheck)# mep-id 1101
Router(config-cfm-xcheck)# mep-id 1103
```

Step 3 Enable Ethernet CFM continuity check for another domain and service.

Example:

```
Router(config)# ethernet cfm
Router(config-cfm)# domain evpn-bd2 level 3 id null
Router(config-cfm-dmn)# service evpn-bd2 bridge group BG2 bridge-domain BD2 $
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Step 4 Configure MEP cross-check for antoher domain.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-xcheck)# mep-id 11
Router(config-cfm-xcheck)# mep-id 21
Router(config-cfm-xcheck)# mep-id 101
Router(config-cfm-xcheck)# mep-id 201
```

Step 5 Configure the bridge domain and assign interfaces to the bridge domain.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/0.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/1.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evi 101
```

Step 6 Configure another bridge domain and assign interfaces to the bridge domain.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG2
Router(config-l2vpn-bg)# bridge-domain BD2
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/2.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/5.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evi 201
```

Step 7 Enable CFM on interfaces with corresponding domain, service, and MEP IDs.

Example:

```
Router(config)# interface TenGigE0/0/0/0.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain evpn-bd service evpn-bd1 mep-id 1103
Router(config-if-cfm)# root
Router(config)# interface TenGigE0/0/0/1.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain evpn-bd service evpn-bd1 mep-id 5
Router(config-if-cfm)# root
Router(config)# interface TenGigE0/0/0/2.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain evpn-bd2 service evpn-bd2 mep-id 101
Router(config-if-cfm)# root
Router(config)# interface TenGigE0/0/0/5.2 l2transport
Router(config-subif)# encapsulation dot1q 2
```

```
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain evpn-bd2 service evpn-bd2 mep-id 11
```

Step 8 Running configuration of scenario 4.

Example:

```
ethernet cfm
 domain evpn-bd level 4 id null
  service evpn-bd1 bridge group BG1 bridge-domain BD1
   continuity-check interval 10s
   mep crosscheck
    mep-id 5
    mep-id 6
    mep-id 1101
    mep-id 1103
   !
  !
 !
 domain evpn-bd2 level 3 id null
  service evpn-bd2 bridge group BG2 bridge-domain BD2
   continuity-check interval 10s
   mep crosscheck
    mep-id 11
    mep-id 21
    mep-id 101
    mep-id 201
   !
  !
 !
!
l2vpn
 bridge group BG1
  bridge-domain BD1
   interface TenGigE0/0/0/0.1
   !
   interface TenGigE0/0/0/1.2
   !
   evi 101
   !
  !
 !
 bridge group BG2
  bridge-domain BD2
   interface TenGigE0/0/0/2.1
   !
   interface TenGigE0/0/0/5.2
   !
   evi 201
   !
  !
 !
!
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain evpn-bd service evpn-bd1 mep-id 1103
  !
 !
!

interface TenGigE0/0/0/1.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
```

```
  mep domain evpn-bd service evpn-bd1 mep-id 5
  !
 !
!

interface TenGigE0/0/0/2.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain evpn-bd2 service evpn-bd2 mep-id 101
  !
 !
!

interface TenGigE0/0/0/5.2 l2transport
 encapsulation dot1q 2
 ethernet cfm
  mep domain evpn-bd2 service evpn-bd2 mep-id 11
  !
 !
!
```

CFM on EVPN E-Line single-homing

A CFM on EVPN E-Line single-homing is a network monitoring solution that

- operates on networks using EVPN E-Line single-homed devices
- monitors E-Line services providing high-speed Layer 2 connectivity, and
- ensures high resiliency for the Layer 2 services.

CFM up MEP is supported only on single-homing Layer 2 main and subinterfaces.

How CFM on EVPN E-Line single-homing works

Summary

The key components involved in the CFM on EVPN E-Line single-homing process are:

- Cisco routers (Nodes 1, 2, and 3): These routers form the network topology and are interconnected.
- CFM: Provides fault detection and monitoring capabilities within the EVPN E-Line service.
- EVPN E-Line service: A point-to-point Ethernet VPN service that supports single-homing configurations.

CFM on EVPN E-Line single-homing enables routers to exchange monitoring messages over a full mesh to detect faults and ensure continuous service integrity. This process supports rapid fault detection and high availability in single-homed EVPN E-Line deployments.

Workflow

These stages describe how CFM on EVPN E-Line single-homing works.

1. Cisco routers (Nodes 1, 2, and 3) establish full mesh connectivity to support CFM operations within the EVPN E-Line topology.
2. CFM messages are exchanged between the routers to monitor connectivity and detect faults.

3. Faults detected by CFM trigger notifications and enable rapid troubleshooting within the EVPN E-Line single-homing environment.
4. Routers maintain continuous monitoring to ensure service integrity and performance.

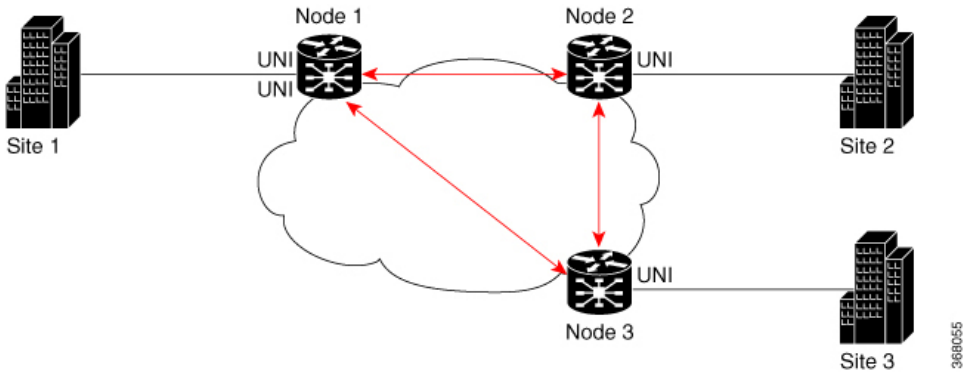
Result

This process enables robust fault management and monitoring for EVPN E-Line single-homing deployments, ensuring high availability and rapid fault detection in the network.

Configure CFM on EVPN E-Line single-homing

Enable and validate CFM to monitor service continuity and endpoint liveliness for EVPN E-Line service.

Use this task to configure CFM service continuity checks, Maintenance End Point (MEP) cross-checks, and enable CFM on interfaces for EVPN E-Line single-homing deployments.



Procedure

Step 1 Enable CFM continuity check.

Example:

```
Router# ethernet cfm
Router(config-cfm# domain xcup1 level 7 id null
Router(config-cfm-dmn)# service xcup1 xconnect group evpn_vpws_Bundle_ether203 p2p evpn_vpws-100 id
 number 4001
Router(config-cfm-dmn-svc)# continuity-check interval 1s
```

Step 2 Configure MEP cross-check to validate remote MEPs.

Example:

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-dmn-svc)# mep-id 4001
Router(config-cfm-dmn-svc)# commit
```

Repeat the above configurations for node 2 and node 3, with the respective mep-id values. For node 2, configure MEP cross-check with respective mep-id values of node 1 and node 3 (2001 and 3001 respectively, in this example). For node 3, configure MEP cross-check with respective mep-id values of node 1 and node 2 (4001 and 2001 respectively, in this example).

Step 3 Enable CFM on the interface.

Example:

```
Router# configure
Router(config)# interface Bundle-Ether203.2001 l2transport
Router(config-subif)# encapsulation dot1q 2001
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain xcup1 service xcup1 mep-id 2001
Router(config-if-cfm-mep)# commit
```

You must repeat the above configurations for node 2 and node 3, with the respective *mep-id* values.

Step 4 Running configuration of CFM on EVPN E-Line.

Example:

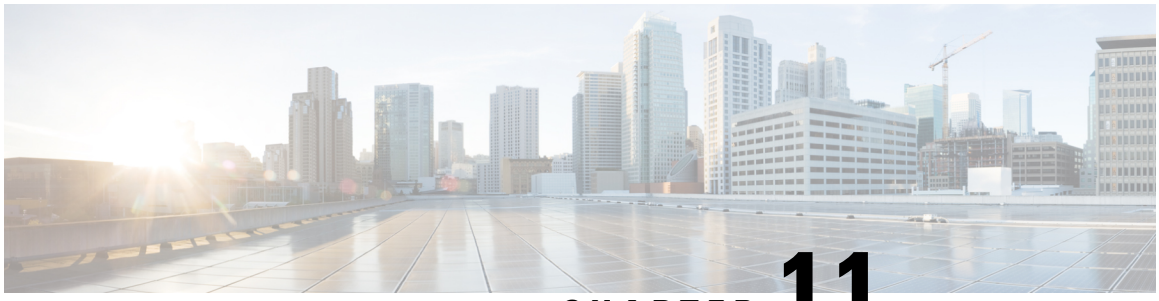
```
ethernet cfm
 domain xcup1 level 7 id null
  service xcup1 xconnect group evpn_vpws_Bundle_ether203 p2p evpn_vpws-100 id number 4001
   continuity-check interval 1s
   mep crosscheck
    mep-id 4001
   !
  !
 !
!
interface Bundle-Ether203.2001 l2transport
 encapsulation dot1q 2001
 ethernet cfm
  mep domain xcup1 service xcup1 mep-id 2001
 !
```

Step 5 Use the **show ethernet cfm services** command to verify CFM on EVPN E-Line.

Example:

```
Router# show ethernet cfm services

   Summary for Domain xcup1 (level 7), Service xcup1
================================================================================
Domain MIB index: 1, Service MIB index: 1
Domain ID: NULL, Service ID: UINT: 4001
Service configured on P2P cross-connect evpn_vpws-100 in group evpn_vpws_Bundle_ether
CCM interval: 1s
Local MEPs: 1 total
Peer MEPs: 2 total
MIPs: 0, MIP creation rule: always
```

CFM is configured on EVPN E-Line, providing continuous service monitoring and fault detection through MEP cross-checks and continuity checks, ensuring network reliability and rapid troubleshooting.



CHAPTER 11

EVPN MPLS Transport

This chapter guides users in configuring and optimizing EVPN bridging and E-Line services over MPLS transport, including BGP-LU, LDP, and segment routing, with steps for traffic engineering, preferred paths, and service resilience.

EVPN bridging and E-Line services over BGP-LU underlay

A BGP Labeled Unicast (BGP-LU) underlay is a network transport method that

- enables configuration of end-to-end services between data centers
- supports various EVPN E-LAN and E-Line services, and
- provides load balancing at the transport, BGP-LU, and service levels.

Table 39: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Bridging and E-Line Services over BGP-LU Underlay	Release 26.2.1	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 Bridging and E-Line Services over BGP-LU Underlay	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D

EVPN Bridging and E-Line Services over BGP-LU Underlay	Release 25.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Modular Systems (8800 [LC ASIC: P100]) You can now configure end-to-end services between data centers using the BGP Labeled Unicast (BGP-LU) underlay with segment routing. The feature supports EVPN E-LAN and E-Line services and enables load balancing across transport, BGP-LU, and service levels using segment routing.
EVPN Bridging and E-Line Services over BGP-LU Underlay	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on the Cisco 8011-4G24Y4H-I routers.
EVPN Bridging and E-Line Services over BGP-LU Underlay	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The EVPN Bridging and E-Line Services over BGP-LU Underlay functionality is now extended to the Cisco 8712-MOD-M routers.
EVPN Bridging and E-Line Services over BGP-LU Underlay	Release 24.3.1	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*) *The EVPN Bridging and E-Line Services over BGP-LU Underlay functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
EVPN Bridging and E-Line Services over BGP-LU Underlay	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The EVPN Bridging and E-Line Services over BGP-LU Underlay functionality is now extended to routers with the 88-LC1-36EH line cards.
EVPN Bridging and E-Line Services over BGP-LU Underlay	Release 7.11.1	You can configure end-to-end services between data centers using the BGP Labeled Unicast (BGP-LU) underlay. This feature allows you to configure various EVPN E-LAN and E-Line services, and enables load balancing at transport, BGP-LU, and service level.

End-to-end EVPN services with BGP-LU underlay

The EVPN bridging and E-Line services over BGP-LU underlay feature enables configuration of end-to-end EVPN services between data centers. This feature supports equal-cost multipath (ECMP) load balancing at three levels: transport, BGP-LU, and service.

This feature supports these services:

- EVPN aliasing over BGP-LU using IGP, including segment routing (SR) or non-SR protocols such as LDP or IGP.
- E-Line services over BGP-LU using IGP.

By leveraging BGP with label switching, this underlay provides seamless connectivity and efficient traffic distribution across data center networks. It allows network administrators to build scalable and resilient inter-data center services with flexible load balancing capabilities.

How EVPN bridging and E-Line services over BGP-LU underlay works

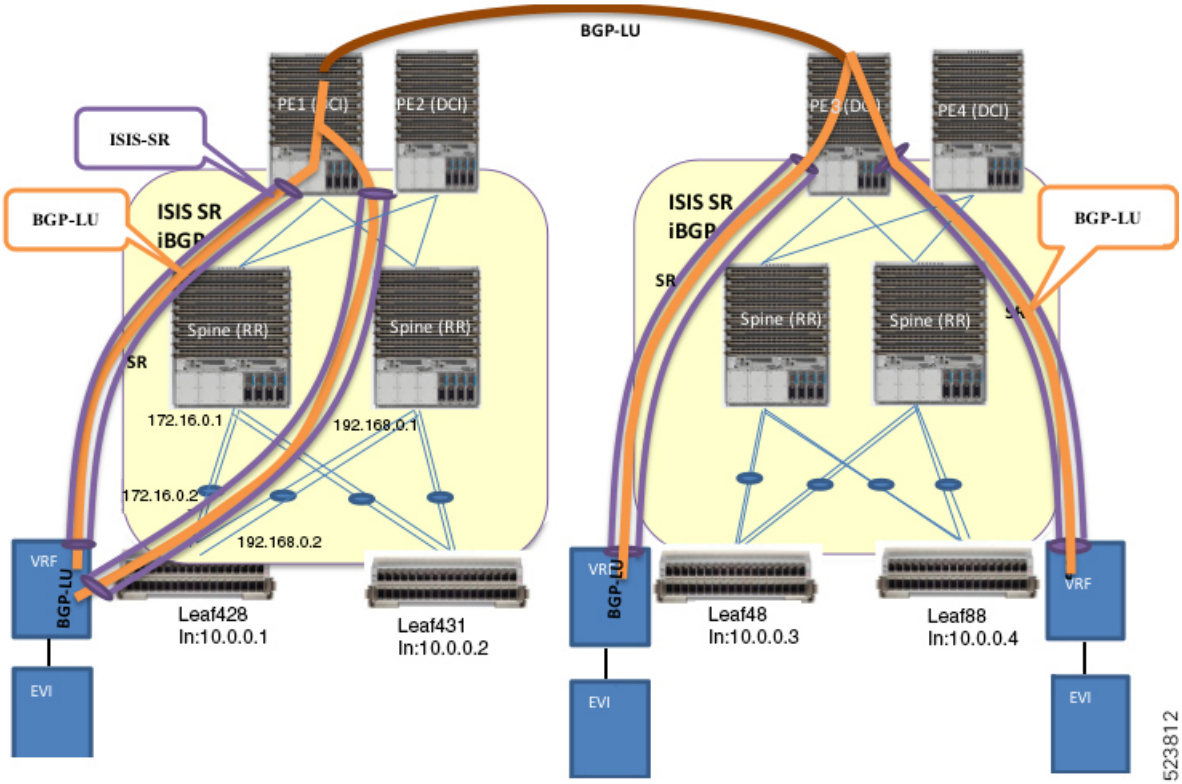
Summary

The key components involved in the EVPN bridging and E-Line services over BGP-LU underlay process are:

- Leaf nodes: Configure EVPN with bridging and inter-subnet routing; act as default gateways for local hosts.
- Spine nodes: Route traffic between leaf nodes and serve as route reflectors (RR) for iBGP.
- Provider edge (PE) device and Data Center Interconnect (DCI): Connect spine nodes across data centers.
- IS-IS labeled IGP and iBGP: Provide internal routing and route reflection across leaf, spine, and DCI nodes.
- BGP Labeled Unicast (BGP-LU): Establishes labeled routes tunneled through IS-IS Segment Routing (SR) paths between data centers.

EVPN bridging and E-Line services over BGP-LU underlay use leaf and spine nodes with IS-IS and iBGP routing to establish labeled tunnels across data centers. This setup ensures scalable, resilient bridging and routing with efficient load balancing across interconnected sites.

Workflow



These are the stages of EVPN bridging and E-Line services over BGP-LU underlay.

1. Configure EVPN with bridging and inter-subnet routing on leaf nodes.
2. Connect hosts to leaf nodes, which route traffic across spine nodes.
3. Connect spine nodes through PE devices and DCI for inter-data center connectivity.
4. Enable IS-IS labeled IGP and iBGP across leaf, spine, and DCI nodes, with spine nodes acting as route reflectors.
5. Configure IS-IS SR policy across leaf, spine, and DCI nodes.
6. Establish BGP-LU sessions between data centers.
7. Learn BGP-LU routes on leaf nodes and tunnel them through IS-IS SR labeled paths.

   For example, leaf node Leaf428 learns BGP-LU routes for remote loopback addresses 10.0.0.3 and 10.0.0.4.

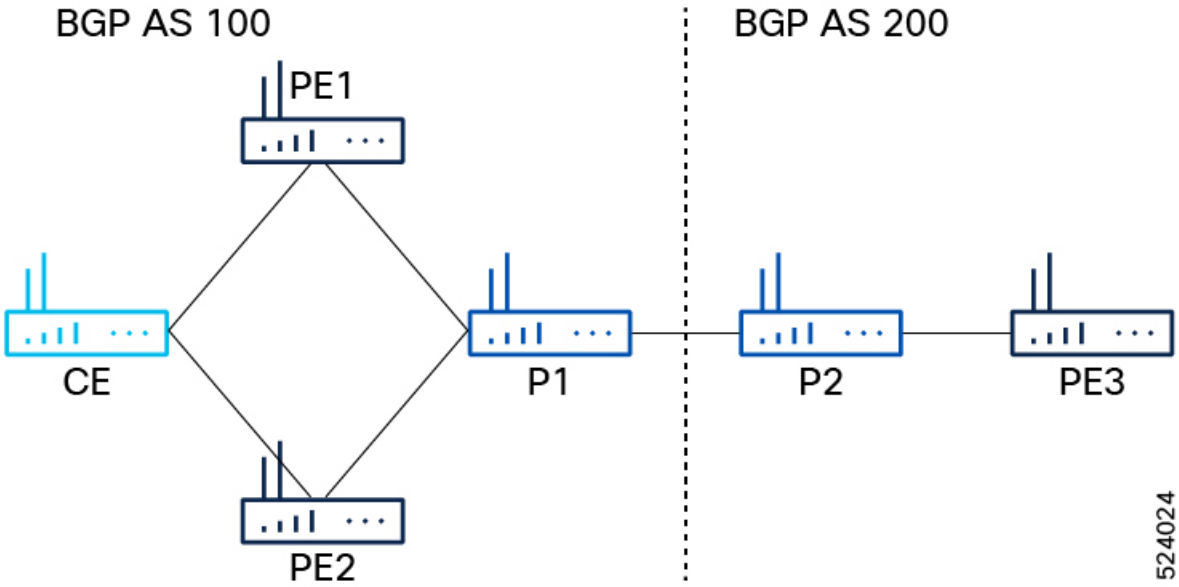
Result

This process enables scalable and resilient EVPN bridging and E-Line services with efficient load balancing and routing across interconnected data centers.

Configure EVPN bridging and E-Line services over BGP-LU underlay with LDP

Configure EVPN Bridging and E-Line Services overging and E-Line Services over a BGP-LU underlay network using LDP.

This task applies to a network topology with P routers (P1, P2) and PE routers (PE1, PE2, PE3) connected across two BGP autonomous systems (AS 100 and AS 200). P1 connects to P2, and P1 connects to PE1 and PE2 in AS 100, while P2 connects to PE3 in AS 200.



Before you begin

Ensure you have the network topology as described and access to configure IGP, MPLS, and BGP on all routers.

Procedure

Step 1 Configure IGP, MPLS, and BGP on PE1, PE2, and PE3.

a) Configure IGP on PE1 and PE2

Example:

```
Router(config)#router ospf pyats_test
Router(config-ospf)#router-id 54.54.54.54
Router(config-ospf)#redistribute bgp 100
Router(config-ospf)#mpls ldp sync
Router(config-ospf)#mpls ldp auto-config
Router(config-ospf)#area 0
Router(config-ospf-ar)#interface loopback0
Router(config-ospf-ar-if)#exit
Router(config-ospf-ar)#interface FourHundredGigE0/0/0/2
Router(config-ospf-ar-if)#commit
```

b) Configure MPLS on PE1 and PE2.

Example:

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 54.54.54.54
Router(config-ldp)# address-family ipv4
Router(config-ldp-af)# label
Router(config-ldp-af-lbl)# local
Router(config-ldp-af-lbl)# allocate for host-routes
Router(config-ldp-af-lbl-lcl)# root
Router(config)# mpls ldp
Router(config-ldp)# interface FourHundredGigE0/0/0/2
```

c) Configure BGP on PE1 and PE2.

Example:

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 54.54.54.54
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#exit
Router(config-bgp)#address-family l2vpn evpn
Router(config-bgp-af)#retain route-target all
Router(config-bgp-af)#exit
Router(config-bgp)#neighbor-group IBGP-PEERS
Router(config-bgp-nbrgrp)#remote-as 100
Router(config-bgp-nbrgrp)#update-source loopback0
Router(config-bgp-nbrgrp)#address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#address-family l2vpn evpn
```

d) Configure IGP, MPLS, and BGP on PE3.

Example:

```
Router# configure
Router(config)#router ospf pyats_test
Router(config-ospf)#router-id 51.51.51.51
Router(config-ospf)#redistribute bgp 200
Router(config-ospf)#mpls ldp sync
Router(config-ospf)#mpls ldp auto-config
Router(config-ospf)#area 0
Router(config-ospf-ar)#interface loopback0
Router(config-ospf-ar-if)#exit
Router(config-ospf-ar)#interface FourHundredGigE0/0/0/10
Router(config-ospf-ar-if)#commit
Router(config)# mpls ldp
Router(config-ldp)# router-id 51.51.51.51
Router(config-ldp)# address-family ipv4
Router(config-ldp-af)# label
Router(config-ldp-af-lbl)# local
Router(config-ldp-af-lbl-lcl)# allocate for host-routes
Router(config-ldp-af-lbl-lcl)# root
Router(config)# mpls ldp
Router(config-ldp)# interface FourHundredGigE0/0/0/10
Router(config-ldp-if)# root
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 54.54.54.54
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#exit
Router(config-bgp)#address-family l2vpn evpn
Router(config-bgp-af)#retain route-target all
Router(config-bgp-af)#exit
Router(config-bgp)# neighbor 56.56.56.56
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#update-source loopback0
Router(config-bgp-nbr)#address-family ipv4 unicast
```

```
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#address-family l2vpn evpn
```

Step 2 Configure iBGP peer on P1 and PE2.

a) Configure iBGP peer on P1.

Example:

```
Router(config-bgp)# neighbor 52.52.52.52
Router(config-bgp-nbr)# use neighbor-group IBGP-PEERS
```

b) Configure iBGP peer on PE2.

Example:

```
Router(config-bgp)# neighbor 55.55.55.55
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# use neighbor-group IBGP-PEERS
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
```

Step 3 Configure P2 as route reflector.

Example:

```
Router(config)# prefix-set LOOPBACKS
Router(config-pfx)# 53.53.53.53,
Router(config-pfx)# 54.54.54.54,
Router(config-pfx)# 55.55.55.55,
Router(config-pfx)# 52.52.52.52,
Router(config-pfx)# 56.56.56.56,
Router(config-pfx)# 51.51.51.51
Router(config-pfx)# end-set
Router(config)# route-policy passall
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# route-policy MATCH_LOOPBACKS
Router(config-rpl)#if destination in LOOPBACKS then
Router(config-rpl-if)#pass
Router(config-rpl-if)#else
Router(config-rpl-else)#drop
Router(config-rpl-else)#endif
Router(config-rpl)#end-policy
```

Step 4 Configure route policy, IGP, MPLS, and BGP on P1 and P2.

a) Configure route policy, IGP, MPLS, and BGP on P1.

Example:

```
Router(config)#router ospf pyats_test
Router(config-ospf)#router-id 52.52.52.52
Router(config-ospf)#redistribute bgp 100
Router(config-ospf)#mpls ldp sync
Router(config-ospf)#mpls ldp auto-config
Router(config-ospf)#area 0
Router(config-ospf-ar)#interface loopback0
Router(config-ospf-ar-if)#exit
Router(config-ospf-ar)#interface FourHundredGigE0/0/0/11
Router(config-ospf-ar-if)#exit
Router(config-ospf-ar)#interface FourHundredGigE0/0/0/12
Router(config-ospf-ar-if)#root
Router(config)# router static
```

```
Router(config-static)# address-family ipv4 unicast
Router(config-static-afi)# 100.0.0.2/32 FourHundredGigE0/0/0/13
Router(config-static-afi-if)# root
Router(config)# mpls ldp
Router(config-ldp)# router-id 52.52.52
Router(config-ldp)# address-family ipv4
Router(config-ldp-af)# label
Router(config-ldp-af-lbl)# local
Router(config-ldp-af-lbl-lcl)# allocate for host-routes
Router(config-ldp-af-lbl-lcl)# root
Router(config)# mpls ldp
Router(config-ldp)# interface FourHundredGigE0/0/0/11
Router(config-ldp-if)# exit
Router(config-ldp)# interface FourHundredGigE0/0/0/12
```

b) Configure route policy, IGP, MPLS, and BGP on P2.

Example:

```
Router(config)#router ospf pyats_test
Router(config-ospf)#router-id 56.56.56.56
Router(config-ospf)#redistribute bgp 200
Router(config-ospf)#mpls ldp sync
Router(config-ospf)#mpls ldp auto-config
Router(config-ospf)#area 0
Router(config-ospf-ar)#interface loopback0
Router(config-ospf-ar-if)#passive enable
Router(config-ospf-ar-if)#exit
Router(config-ospf-ar)#interface FourHundredGigE0/0/0/2
Router(config-ospf-ar)#root
Router(config)# router static
Router(config-static)# address-family ipv4 unicast
Router(config-static-afi)# 100.0.0.1/32 FourHundredGigE0/0/0/5
Router(config-static-afi)# root
Router(config)# mpls ldp
Router(config-ldp)# router-id 56.56.56.56
Router(config-ldp)# address-family ipv4
Router(config-ldp-af)# label
Router(config-ldp-af-lbl)# local
Router(config-ldp-af-lbl-lcl)# allocate for host-routes
Router(config-ldp-af-lbl-lcl)# root
Router(config)# mpls ldp
Router(config-ldp)# interface FourHundredGigE0/0/0/4
```

c) Configure router reflector client on P2, which is essential for copying the EVPN routes between the AS.

Example:

```
Router(config)#router bgp 200
Router(config-bgp)#bgp router-id 56.56.56.56
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#network 51.51.51.51/32
Router(config-bgp-af)#network 52.52.52.52/32
Router(config-bgp-af)#network 53.53.53.53/32
Router(config-bgp-af)#network 54.54.54.54/32
Router(config-bgp-af)#network 55.55.55.55/32
Router(config-bgp-af)#network 56.56.56.56/32
Router(config-bgp-af)#redistribute connected
Router(config-bgp-af)#redistribute ospf 0
Router(config-bgp-af)#allocate-label all
Router(config-bgp-af)#exit
Router(config-bgp)#address-family l2vpn evpn
Router(config-bgp-af)#retain route-target all
```

```
Router(config-bgp-af)#exit
Router(config-bgp)#neighbor-group IBGP-PEERS
Router(config-bgp-nbrgrp)#remote-as 200
Router(config-bgp-nbr)#update-source loopback0
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#address-family l2vpn evpn
Router(config-bgp-nbr-af)#route-reflector-client
```

Step 5 Configure P1 and P2 as eBGP neighbor.

a) Configure P1 as eBGP neighbor.

Example:

```
Router(config-bgp)# neighbor 100.0.0.1
Router(config-bgp-nbr)#remote-as 100
Router(config-bgp-nbr)#ebgp-multihop 255
Router(config-bgp-nbr)#address-family ipv4 labeled-unicast
Router(config-bgp-nbr-af)#next-hop-self
Router(config-bgp-nbr-af)#route-policy passall in
Router(config-bgp-nbr-af)#route-policy MATCH_LOOPBACKS out
Router(config-bgp-nbr-af)#send-extended-community-ebgp
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#address-family l2vpn evpn
Router(config-bgp-nbr-af)#route-policy passall in
Router(config-bgp-nbr-af)#route-policy passall out
Router(config-bgp-nbr-af)#next-hop-unchanged
```

b) Configure P2 as eBGP neighbor.

Example:

```
Router(config-bgp)# neighbor 100.0.0.2
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#ebgp-multihop 255
Router(config-bgp-nbr)#address-family ipv4 labeled-unicast
Router(config-bgp-nbr-af)#next-hop-self
Router(config-bgp-nbr-af)#route-policy passall in
Router(config-bgp-nbr-af)#route-policy MATCH_LOOPBACKS out
Router(config-bgp-nbr-af)#send-extended-community-ebgp
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#address-family l2vpn evpn
Router(config-bgp-nbr-af)#route-policy passall in
Router(config-bgp-nbr-af)#route-policy passall out
Router(config-bgp-nbr-af)#next-hop-unchanged
```

Step 6 Configure PE1, PE2, and PE3 as iBGP neighbor.

a) Configure PE3 as iBGP neighbor.

Example:

```
Router(config-bgp)#neighbor 51.51.51.51
Router(config-bgp-nbr)#use neighbor-group IBGP-PEERS
```

b) Configure PE1 and PE2 as iBGP neighbor.

Example:

```
Router(config-bgp)#neighbor 54.54.54.54
Router(config-bgp-nbr)#use neighbor-group IBGP-PEERS
Router(config-bgp-nbr)#exit
Router(config-bgp)#neighbor 55.55.55.55
Router(config-bgp-nbr)#use neighbor-group IBGP-PEERS
```

Step 7 For P1, the iBGP peers are PE1 and PE2, and the eBGP peer is P2.

a) For P1, the iBGP peers are PE1 and PE2, and the eBGP peer is P2.

Example:

```
Router(config)# prefix-set LOOPBACKS
Router(config-pfx)# 53.53.53.53,
Router(config-pfx)# 54.54.54.54,
Router(config-pfx)# 55.55.55.55,
Router(config-pfx)# 52.52.52.52,
Router(config-pfx)# 56.56.56.56,
Router(config-pfx)# 51.51.51.51
Router(config-pfx)# end-set
Router(config)# route-policy passall
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# route-policy MATCH_LOOPBACKS
Router(config-rpl)#if destination in LOOPBACKS then
Router(config-rpl-if)#pass
Router(config-rpl-if)#else
Router(config-rpl-else)#drop
Router(config-rpl-else)#endif
Router(config-rpl)#end-policy
```

b) Configure router reflector client.

Example:

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 52.52.52.52
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#network 51.51.51.51/32
Router(config-bgp-af)#network 52.52.52.52/32
Router(config-bgp-af)#network 53.53.53.53/32
Router(config-bgp-af)#network 54.54.54.54/32
Router(config-bgp-af)#network 55.55.55.55/32
Router(config-bgp-af)#network 56.56.56.56/32
Router(config-bgp-af)#redistribute connected
Router(config-bgp-af)#redistribute ospf 0
Router(config-bgp-af)#allocate-label all
Router(config-bgp-af)#exit
Router(config-bgp)#address-family l2vpn evpn
Router(config-bgp-af)#retain route-target all
Router(config-bgp-af)#exit
Router(config-bgp)#neighbor-group IBGP-PEERS
Router(config-bgp-nbrgrp)#remote-as 100
Router(config-bgp-nbr)#update-source loopback0
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#address-family l2vpn evpn
Router(config-bgp-nbr-af)#route-reflector-client
```

Step 8 Configure L2VPN and EVPN on PE1, PE2, and PE3.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface Bundle-Ether3.1
Router(config-l2vpn-bg-bd-ac)# evi 1
Router(config-l2vpn-bg-bd-ac)# root
```

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg2
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface Bundle-Ether3.2
Router(config-l2vpn-bg-bd-ac)# evi 2
Router(config-l2vpn-bg-bd-ac-evi)# root
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# exit
Router(config-evpn)# evi 2
Router(config-evpn-evi)# advertise-mac
```

Step 9 Use these show commands to verify that you have successfully configured EVPN bridging and E-Line services over BGP-LU underlay with LDP.

- show evpn internal-label vpn-id 1 detail
- show bgp l2vpn evpn route-type inclusive-mcast
- show l2vpn forwarding bridge-domain mac location 0/RP0/CPU0
- show bgp l2vpn evpn route-type mac-advertisement

Example:

```
Router# show evpn internal-label vpn-id 1 detail

VPN-ID     Encap  Ethernet Segment Id         EtherTag     Label
---------- ------ -------------------------- ----------   --------
1          MPLS   0040.0000.0000.0000.0001    0            24010
   Multi-paths resolved: TRUE (Remote all-active)
   Multi-paths Internal label: 24010
    EAD/ES  (ID:0x0000000000000652)
                54.54.54.54                               0
                55.55.55.55                               0
    EAD/EVI (ID:0x0000000000000649)
                54.54.54.54                               24000
                55.55.55.55                               24000
   Summary pathlist (ID 0x000000000000064d):
 0x02000001 (P) 54.54.54.54                               24000
 0x02000002 (P) 55.55.55.55                               24000

Router# show bgp l2vpn evpn route-type inclusive-mcast

BGP router identifier 51.51.51.51, local AS number 200
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0
BGP table nexthop route policy:
BGP main routing table version 100
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 51.51.51.51:1 (default for vrf bd1)
Route Distinguisher Version: 94
*> [3][0][32][51.51.51.51]/80
                      0.0.0.0                                  0 i N
```

```
*>i[3][0][32][54.54.54.54]/80
                      54.54.54.54                   100      0 100 i N
*>i[3][0][32][55.55.55.55]/80
                      55.55.55.55                   100      0 100 i N
Route Distinguisher: 51.51.51.51:2 (default for vrf bd2)
Route Distinguisher Version: 100
*> [3][0][32][51.51.51.51]/80
                      0.0.0.0                                0 i N
*>i[3][0][32][54.54.54.54]/80
                      54.54.54.54                   100      0 100 i N
*>i[3][0][32][55.55.55.55]/80
                      55.55.55.55                   100      0 100 i N
Route Distinguisher: 54.54.54.54:1
Route Distinguisher Version: 92
*>i[3][0][32][54.54.54.54]/80
                      54.54.54.54                   100      0 100 i N
Route Distinguisher: 54.54.54.54:2
Route Distinguisher Version: 99
*>i[3][0][32][54.54.54.54]/80
                      54.54.54.54                   100      0 100 i N
Route Distinguisher: 55.55.55.55:1
Route Distinguisher Version: 67
*>i[3][0][32][55.55.55.55]/80
                      55.55.55.55                   100      0 100 i N
Route Distinguisher: 55.55.55.55:2
Route Distinguisher Version: 96
*>i[3][0][32][55.55.55.55]/80
                      55.55.55.55                   100      0 100 i N

Processed 10 prefixes, 10 paths
```

Router# **show l2vpn forwarding bridge-domain mac location 0/RP0/CPU0**

```
 To Resynchronize MAC table from the Network Processors, use the command...
    l2vpn resynchronize forwarding mac-address-table location <r/s/i>

Mac Address    Type    Learned from/Filtered on    LC learned Resync Age/Last Change Mapped to

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

0000.cccc.dddd dynamic FH0/0/0/0.2                N/A        12 Mar 13:17:36        N/A   --> MAC
0000.cccc.dddd was locally learned from interface FH0/0/0/0.2
0000.aaaa.bbbb EVPN    BD id: 1                   N/A        N/A                    N/A   --> MAC
0000.aaaa.bbbb was advertised from PE1/PE2
```

Router# **show bgp l2vpn evpn route-type mac-advertisement**

```
BGP router identifier 51.51.51.51, local AS number 200
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0
BGP table nexthop route policy:
BGP main routing table version 100
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 51.51.51.51:2 (default for vrf bd2)
Route Distinguisher Version: 100
*>i[2][0][48][0000.aaaa.bbbb][0]/104 -->
```

```
                     54.54.54.54                   100      0 100 i N
* i                   55.55.55.55                   100      0 100 i N
*> [2][0][48][0000.cccc.dddd][0]/104 -->
                     0.0.0.0                                0 i N
Route Distinguisher: 54.54.54.54:2
Route Distinguisher Version: 99
*>i[2][0][48][0000.aaaa.bbbb][0]/104
                     54.54.54.54                   100      0 100 i N
Route Distinguisher: 55.55.55.55:2
Route Distinguisher Version: 96
*>i[2][0][48][0000.aaaa.bbbb][0]/104
                     55.55.55.55                   100      0 100 i N
Processed 4 prefixes, 5 paths
```

Configure EVPN bridging and E-Line services over BGP-LU underlay with SR

Configure EVPN bridging and E-Line services over a BGP-LU underlay network using SR for efficient traffic engineering and simplified forwarding.

This task applies to networks where Segment Routing is enabled within the IGP domain using IS-IS, and BGP-LU is used to advertise loopback addresses with associated SR labels. The configuration involves PE and Route Reflector (RR) nodes participating in EVPN services over the SR-enabled underlay.

Before you begin

Ensure all participating nodes support IS-IS with Segment Routing and BGP-LU. Have loopback interfaces configured for router IDs and SR prefix SIDs assigned.

Procedure

Step 1 Configure IS-IS with SR on all participating nodes.

Configure SR with IS-IS to enable SR within the IGP domain and assign prefix SIDs to the router loopback interface. Configure all the participating nodes, which include the PE or Route Reflector (RR) nodes in the underlay to run IS-IS and advertise their loopbacks with SIDs. The prefix SID varies for each node and instance.

Example:

```
Router#config
Router(config)#router isis igp1
Router(config-isis)#address-family ipv4 unicast
Router(config-isis-af)#segment-routing mpls sr-prefer
Router(config-isis-af)#segment-routing prefix-sid-map advertise-local
Router(config-isis-af)#exit
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#segment-routing mpls sr-prefer
Router(config-isis-af)#segment-routing prefix-sid-map advertise-local
Router(config-isis-af)#exit
Router(config-isis)#interface Loopback 0
Router(config-isis-if)#passive
Router(config-isis-if)#address-family ipv4 unicast
Router(config-isis-if-af)#prefix-sid index 1
Router(config-isis-if-af)#exit
Router(config-isis-if)#address-family ipv6 unicast
Router(config-isis-if-af)#prefix-sid index 101
```

Step 2 Configure BGP on all nodes.

Ensure that the **bgp router-id** is the loopback address used for peering and SR SID assignment. The router ID varies for each node.

Example:

```
Router(config)#router bgp 65600
Router(config-bgp)#nsr
Router(config-bgp)#bgp router-id 192.168.1.1
Router(config-bgp)#bgp graceful-restart
Router(config-bgp)#ibgp policy out enforce-modifications
```

Step 3 Enable the BGP-LU address families and advertise the router loopback address with an associated label derived from the SR node SID.

Configure all the participating nodes to advertise the router loopback address through BGP-LU.

Example:

The network and SID index vary for each node. The following is a sample configuration for IPv4 and IPv6 unicast address families.

```
Router#config
Router(config)#router bgp 64600
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#additional-paths receive
Router(config-bgp-af)#additional-paths send
Router(config-bgp-af)#nexthop trigger-delay critical 50
Router(config-bgp-af)#network 192.168.1.1/32 route-policy SID (1)
Router(config-bgp-af)#allocate-label all
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#additional-paths receive
Router(config-bgp-af)#additional-paths send
Router(config-bgp-af)#nexthop trigger-delay critical 50
Router(config-bgp-af)#network 2001:DB8::/32 route-policy SID (101)
Router(config-bgp-af)#allocate-label all
```

Example:

Sample route-policy configuration.

```
Router#config
Router(config)#route-policy SID ($SID)
Router(config-rpl)#set label-index $SID
Router(config-rpl)#end-policy
```

Step 4 Configure BGP neighbors and neighbor groups.

Configure the BGP neighbors or neighbor groups to establish iBGP peering between all the participating nodes such as PEs and RRs, and enable the BGP-LU and EVPN address families.

a) Create a session group with remote AS and update source as loopback0.

Example:

```
Router(config)#router bgp 64600
Router(config-bgp)#session-group current
Router(config-bgp-sngrp)#remote-as 64600
Router(config-bgp-sngrp)#update-source Loopback 0
```

b) Configure a BGP neighbor group for BGP-LU for IPv4 and IPv6 address families.

Example:

```
Router(config-bgp)#neighbor-group bgp-lu-rr
Router(config-bgp-nbrgrp)#use session-group current
Router(config-bgp-nbrgrp)#address-family ipv4 labeled-unicast
Router(config-bgp-nbrgrp-af)#next-hop-self
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#address-family ipv6 labeled-unicast
Router(config-bgp-nbrgrp-af)#next-hop-self
Router(config-bgp-nbrgrp-af)#exit
```

c) Configure BGP neighbor group for L2VPN services such as VPLS-VPWS and EVPN address families.

Example:

```
Router(config-bgp)#neighbor-group bgp-lu-rr
Router(config-bgp-nbrgrp)#use session-group current
Router(config-bgp-nbrgrp)#address-family ipv4 labeled-unicast
Router(config-bgp-nbrgrp-af)#next-hop-self
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#address-family ipv6 labeled-unicast
Router(config-bgp-nbrgrp)#address-family l2vpn vpls-vpws
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#address-family l2vpn evpn
```

d) Associate an individual BGP neighbor with a predefined neighbor group.

Example:

```
Router(config)#router bgp 64600
Router(config-bgp)#neighbor 192.168.101.1
Router(config-bgp-nbr)#use neighbor-group bgp-lu-rr
Router(config-bgp-nbr)#commit
```

L2VPN services over segment routing traffic engineering tunnel

Segment routing is a source routing technology that

- enables the source device to select a path and encode it in the packet header as an ordered list of segments
- uses segments as identifiers for various instructions, and
- supports traffic engineering by creating tunnels between source and destination pairs where the source specifies the path for the packet to follow.

Segment routing for traffic engineering and preferred tunnel path

Segment routing for traffic engineering (SR-TE) establishes a tunnel between a source and destination pair using source routing. The source calculates the path and encodes it as segments in the packet header. Each segment represents an end-to-end path that directs routers in the provider core network to follow the specified route instead of the shortest path determined by the IGP. The destination remains unaware of the tunnel's existence.

Using segment routing to transport MPLS L2VPN services enhances network resilience and convergence compared to MPLS LDP. Segment routing integrates directly with the MPLS architecture without modifying the forwarding plane. In an MPLS data plane segment-routing network, label distribution is handled by the

IGP, eliminating the need for LDP or other signaling protocols. This simplification reduces protocol interactions, making the network more robust and stable. Additionally, segment routing optimizes bandwidth usage and reduces latency compared to traditional MPLS networks.

Preferred tunnel path functionality enables mapping pseudowires to specific traffic-engineering tunnel paths. Attachment circuits connect to designated SR traffic engineering tunnel interfaces rather than remote PE router IP addresses reachable via IGP or LDP. The traffic engineering tunnel then transports traffic from the source to destination PE routers. An SR Policy selects a path when it is valid and has the highest preference value among all candidate paths.

These are the supported services:

- L2VPN preferred path
- VPWS over SR-TE policy
- Decoupled mode for L2VPN and EVPN VPWS services
- VPWS PW over preferred SR-TE using statically configured SR policy
- EVPN PW over preferred SR-TE using On-Demand Nexthop SR policy
- Call admission control for L2VPN point-to-point services over circuit-style SR-TE policies

L2VPN preferred path

L2VPN preferred-path allows you to steer PW traffic over an SR-TE tunnel at the granularity of per PW level. It is recommended to use preferred-path configuration together with statically configured SR policy. EVPN ODN configuration allows you to steer PW traffic over an SR-TE tunnel at the granularity of per EVPN instance (EVI) level. When both the preferred-path and EVPN ODN configurations are used on a PW, preferred-path configuration takes precedence.

EVPN VPLS preferred path over SR-TE policies

EVPN VPLS preferred path over SR-TE policy is a traffic engineering feature that

- explicitly sets a specific path for EVPN VPLS pseudowire traffic based on a SR-TE policy
- ensures traffic follows the defined route between service PE routers for EVPN VPLS services, and
- allows selection of the preferred path on a per EVPN instance (EVI) basis.

Table 40: Feature History Table

Feature Name	Release Information	Feature Description
EVPN VPLS preferred path over SR-TE policies	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 control Layer 2 VPN traffic paths by leveraging SR-TE policies, ensuring optimized and deterministic traffic flows in the network using EVPN VPLS preferred path over SR-TE policy. This feature allows you to bind an EVPN VPLS pseudowire to an SR-TE tunnel interface, overriding the default IGP shortest path.

EVPN VPLS pseudowire path selection with SR-TE policy

EVPN VPLS pseudowire path selection can be explicitly controlled by binding the pseudowire to an SR-TE policy rather than relying on default IGP-based routing between PE routers. This capability allows precise steering of pseudowire traffic through a predefined SR-TE tunnel.

Key highlights:

- Explicit path steering: Traffic for EVPN VPLS pseudowires is directed according to a specific segment-list, enabling precise control over the route through the network.
- Resource reservation: The SR-TE policy can reserve bandwidth and enforce latency or other service-level requirements to guarantee quality for EVPN VPLS services.
- Stable and predictable paths: The selected path for the pseudowire remains consistent, even during IGP metric changes or topology modifications, improving service stability.

How EVPN VPLS preferred paths over SR-TE policies work

This process applies to routers where EVPN VPLS services are deployed with SR-TE policies and enables explicit routing of L2 VPN traffic over a preferred MPLS path rather than the default shortest path.

Summary

The key components involved in the process are:

- PE routers (PE1 and PE2): Configure EVPN VPLS instances to signal MAC and IP reachability and establish pseudowires. PE1 also defines and enforces the SR-TE policy.
- CE devices (CE1 and CE2): Source and destination of Ethernet frames carried over the EVPN VPLS service.
- P routers: Forward labeled packets through the MPLS core based on segment routing labels, adhering to the SR-TE policy.

EVPN VPLS traffic between CE1 and CE2 is explicitly routed over the MPLS core according to the SR-TE policy on PE1. This provides precise traffic engineering control, ensuring L2 VPN traffic follows the operator's preferred path instead of the default shortest path.

Workflow

These stages describe how EVPN VPLS preferred paths over SR-TE policies work.

1. PE1 and PE2 configure EVPN VPLS instances to enable BGP EVPN signaling of MAC and IP reachability and to establish pseudowires between them.
2. PE1 defines an SR-TE policy with the destination set to PE2's loopback interface, specifying an explicit MPLS forwarding path through the provider (P) routers.
3. PE1 maps incoming traffic from CE1 to the EVPN VPLS service.
4. PE1 steers pseudowire traffic into the SR-TE tunnel, ensuring packets follow the explicit path defined by the SR-TE policy.
5. PE1 encapsulates each Ethernet frame from CE1 with a service label for EVPN VPLS and a stack of SR labels representing the SR-TE path.
6. P routers forward the labeled packets through the MPLS core based solely on the segment routing labels, strictly following the SR-TE policy.
7. Upon arrival at PE2, the SR labels are removed, and the EVPN VPLS service label is used to forward the original Ethernet frame to CE2.

Configure EVPN VPLS preferred path over SR-TE policy

Configure EVPN VPLS preferred paths using SR-TE policies to control traffic forwarding on PE routers.

This task applies when you want to define explicit SR-TE paths for EVPN VPLS services by configuring Prefix-SIDs, Adjacency-SIDs, segment lists, and SR-TE policies on PE routers.

Before you begin

Ensure IS-IS routing protocol is enabled and operational on PE routers PE1, PE2, and PE3.

Procedure

Step 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
```

Step 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
```

Step 3 Create segment lists that specify the order of segment traversal for preferred paths.

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:

on PE2

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE2-PE1
Router(config-sr-te-sl)# index 1 mpls label 16100
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment-list name PE2-PE3-PE1
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# index 2 mpls label 16100
Router(config-sr-te-sl)# commit
```

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
```

Step 4 Configure SR-TE policies on each PE router with candidate paths and set preference values to control policy selection.

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 2.2.2.2
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 2.2.2.2
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 3.3.3.3
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
```

Step 5 Configure EVPN VPLS over SR-TE Policy.

Example:

```
Router(config)# extcommunity-set opaque color_500
Router(config-ext)# 500
Router(config-ext)# end-set
```

Define the color in the extended community.

Example:

```
Router(config)# route-policy RPL_color_500
Router(config-rpl)# if evpn-route-type is 1 or evpn-route-type is 3 then
Router(config-rpl-if)# set extcommunity color color_500
Router(config-rpl-elseif)# endif
Router(config-rpl)# end-policy
```

The evpn-route-type 3 needs to be colored, as it provides the label to carry the BUM traffic, when BUM traffic must be sent to a remote PE3 from PE1to be sent through SR-TE policy color 500.

For unicast traffic, the unicast label for a MAC is advertised by route type 2, but when route type 2 is colored using RPL, an error message is observed:

```
UTC: l2vpn_mgr[1291]: %L2-EVPN-4-ODN_ON_UNSUPPORTED_ROUTE : EVPN: ODN/AS Nexthop received on unsupported
Route-Type 2 (MAC/IP Advertisement), SR-TE BSID tunnel, bridge domain EVPN_ELAN, ESI
0000.0000.0000.0000.0000.
```

Associating the access interface (even for single-homing) with an ESI and coloring per EAD/EVI route to steer the traffic through the SR-TE policy:

Example:

```
Router(config)# evpn
Router(config-evpn)# evi 500
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# route-policy export RPL_color_500
Router(config-evpn-evi-bgp-rpl)# exit
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# root
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN-ELAN-500
Router(config-l2vpn-bg)# bridge-domain EVPN-ELAN-500
Router(config-l2vpn-bg-bd)# interface Bundle-Ether3.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evpn
Router(config-l2vpn-bg-bd-evpn)# evi 500
```

Step 6 Verify the EVPN VPLS preferred path over SR-TE policy configuration.

The prefix-SID and adjacency-SID must be in the SR topology.

Example:

```
PE1# show segment-routing traffic-eng ipv4 topology | inc Prefix
  Prefix SID:
    Prefix 1.1.1.1, label 16100 (regular)
  Prefix SID:
    Prefix 3.3.3.3, label 16300 (regular)
  Prefix SID:
    Prefix 2.2.2.2, label 16200 (regular)
```

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: 2.2.2.2
 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: 2.2.2.2)
  Status:
    Admin: up  Operational: up for 05:44:25 (since Feb  1 17:32:34.434)
  Candidate-paths:
    Preference 500:
      Explicit: segment-list PE1-PE2 (active)
        Weight: 0, Metric Type: IGP
          16200 [Prefix-SID, 2.2.2.2]
    Preference 400:
      Explicit: segment-list PE1-PE3-PE2 (inactive)
      Inactive Reason: unresolved first label
        Weight: 0, Metric Type: IGP
  Attributes:
    Binding SID: 27498
      Allocation mode: dynamic
      State: Programmed
      Policy selected: yes
    Forward Class: 0
```

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 12.1.9.2        0
                              Pop          BE121               121.1.0.2       0
```

EVPN ELAN services with SR-TE ODN, RT1, and RT3

EVPN ELAN services with SR-TE ODN, RT1, and RT3 are networking services that

- manage customer MAC addresses as routable entities distributed over a BGP core using EVPN
- leverage SR-TE for flexible, scalable source routing, and
- enable dynamic, policy-based traffic steering for BUM and unicast traffic using route coloring and multihoming scenarios.

Table 41: Feature History Table

Feature Name	Release Information	Feature Description
EVPN ELAN services with SR-TE ODN, RT1, and RT3	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 enable dynamic traffic steering through SR-TE policy, improving traffic management by directing BUM and unicast traffic over optimized SR-TE paths. This capability is achieved by coloring route type 1 (RT1) and route type 3 (RT3) routes with a non-zero extended community color, which triggers traffic to follow the SR-TE policy path instead of the default IGP path.

Key aspects of policy-based dynamic traffic steering for EVPN ELAN ODN

This feature integrates EVPN's MAC mobility and multihoming with SR-TE's traffic engineering capabilities and enables policy-based dynamic traffic steering to BGP next hops using the SR color extended community value.

Key aspects include:

- The service head-end calculates the segment-list path for traffic steering.
- Dynamically created SR paths are maintained as B-SIDs in the Forwarding Information Base (FIB).
- Path selection is based on matching attribute fields and Network Layer Reachability Information (NLRI) fields on EVPN route types RT1 and RT3.
- Supports selective path steering for BUM traffic, whether colored or non-colored.
- Dynamic steering paths are created to aliasing endpoints in the case of all-active preferred (AApF) designated headend (DHD).
- The service head-end dynamically optimizes paths to all endpoints (designated forwarder and non-designated forwarders) reachable to known T-MAC addresses for all-active redundancy (RT1 Ethernet auto-discovery per EVI).
- Failover of egress PE endpoints does not disrupt forwarding through dynamic steering paths. Receiving a MAC mass withdraw or detecting an egress PE down triggers failover on the ingress PE.
- Traffic continues to flow on remaining endpoints (all-active preferred standby, AApS) with load balancing or switches to a new active endpoint (AApF).

- A single MAC withdraw does not break forwarding; the association of the last MAC in a policy to the dynamic steering path is removed correctly. The ingress PE forwards subsequent traffic as BUM until the MAC is relearned through EVPN RT 2.

Benefits of EVPN ELAN services with SR-TE ODN, RT1, and RT3

- Optimized path selection: Dynamically steers traffic to preferred paths using SR color extended communities.
- Enhanced resilience: Supports rapid failover and convergence in multi-homed environments.
- Simplified operations: Automates path creation and traffic steering, reducing manual intervention.
- Scalability: Leverages Segment Routing's inherent scalability for large-scale deployments.
- Policy-driven control: Allows granular control over traffic flow based on EVPN route types and BGP policies.

How EVPN ELAN services with SR-TE ODN, RT1, and RT3 work

EVPN ELAN ODN and AS services integrate EVPN ELAN with SR-TE to provide policy-based dynamic traffic steering for customer MAC addresses over a BGP core. It uses EVPN route types 1 and 3, along with SR color extended communities, to establish and manage optimized paths, particularly in multi-homing scenarios. A non-zero ESI is required for proper operation.

Summary

The key components involved in the process are:

- EVPN ELAN: Manages customer MACs as routable addresses and distributes them over a BGP core.
- SR-TE: Provides flexible and scalable source routing with paths defined by segment lists.
- BGP Core: Distributes EVPN routes and SR-TE policies.
- Service head-end (ingress PE): Calculates and initiates SR-TE paths, applies policies.
- Egress PE (end-points): Receives traffic, handles MACs, and participates in multihoming.
- SR color extended community: Used for policy-based dynamic traffic steering.
- EVPN route type 1 (RT1): Ethernet A-D per ES route, used for all-active redundancy and MAC mass-withdraw.
- EVPN route type 3 (RT3): Inclusive Multicast Ethernet Tag route, used for BUM traffic.
- Ethernet Segment Identifier (ESI): A non-zero identifier for multihoming.

This process enables highly resilient, policy-driven traffic engineering for EVPN ELAN services, optimizing path selection and ensuring rapid convergence during failures in multi-homed environments.

Workflow

These stages describe how EVPN ELAN services with SR-TE ODN and auto-steering RT1 and RT3 work.

1. EVPN ELAN configuration: Network administrators configure EVPN ELAN features, including bridge groups, bridge domains, and EVPN EVI instances, ensuring a non-zero ESI for multi-homing.

2. SR-TE infrastructure setup: The SR global block is configured, and SR-TE policies are defined, including on-demand color policies and explicit segment lists that specify preferred paths.
3. BGP EVPN Integration: BGP is configured to carry L2VPN EVPN routes. Route policies are applied to import and export EVPN routes, specifically setting SR color extended communities based on EVPN Route Type 1 or 3.
4. Path calculation and encoding: The service head-end (ingress PE) calculates dynamic optimized paths based on the configured SR-TE policies and EVPN routes. These paths are encoded into packet headers as segments.
5. Traffic steering: Traffic is steered dynamically to the BGP next-hop based on the SR color extended community value. SR-TE ensures that traffic follows the specified segment list paths instead of the shortest IGP path.
6. Multihoming and redundancy: In multi-homing scenarios, the system uses EVPN RT1 to manage all-active redundancy. BGP policy routing can prefer one path over another or maintain equal preference for a given MAC.
7. BUM traffic handling: EVPN RT3 is used to handle BUM traffic, which can be selectively steered over colored or non-colored paths.
8. Failover and convergence: Upon a MAC mass-withdraw or detection of an egress PE failure, the ingress PE triggers a failover. Traffic is then redirected to remaining active end-points (AApS) with load balancing or to a new active end-point (AApF) until MACs are re-learned through EVPN RT2.

Configure EVPN ELAN services with SR-TE ODN, RT1, and RT3

Configure EVPN ELAN services with SR-TE ODN and auto-steering RT1 and RT3.

This task applies when you want to leverage PCE-based path computation and dynamic SR-TE policies to optimize EVPN ELAN services.

Procedure

Step 1 Configure SR in IS-IS and configure interfaces BE2, BE3, BE4, and BE5.

a) Enable SR in IS-IS.

Example:

```
Router(config)# router isis isp
Router(config-isis)# net 01.0000.0000.0001.00
Router(config-isis)# distribute instance-id 32 level 2 throttle 5
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# mpls traffic-eng level-1
Router(config-isis-af)# mpls traffic-eng router-id 1.2.3.4
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# segment-routing prefix-sid-map advertise-local
Router(config-isis-af)# exit
```

b) Configure interface BE2.

Example:

```
Router(config-isis)# interface Bundle-Ether2
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# exit
Router(config-isis-if)# exit
```

c) Configure interface BE3.

Example:

```
Router(config-isis)# interface Bundle-Ether3
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# exit
Router(config-isis-if)# exit
```

d) Configure interface BE4.

Example:

```
Router(config-isis)# interface Bundle-Ether4
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# exit
```

e) Configure interface BE5.

Example:

```
Router(config-isis)# interface Bundle-Ether5
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# exit
```

f) Configure Loopback interface.

Example:

```
Router(config-isis)# interface Loopback0
Router(config-isis-if)# passive
Router(config-isis-if)# address-family ipv4 unicast
```

Step 2 Configure SR-TE policy.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# on-demand color 1
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn)# pcep
Router(config-sr-te-color-dyn-pce)# exit
Router(config-sr-te-color-dyn)# metric type igp
Router(config-sr-te-color-dyn)# exit
Router(config-sr-te-color)# exit
Router(config-sr-te)# on-demand color 2
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn)# pcep
Router(config-sr-te-color-dyn-pce)# exit
Router(config-sr-te-color-dyn)# metric type igp
Router(config-sr-te-color-dyn)# exit
Router(config-sr-te-color)# exit
```

Step 3 Configure PCE and PCC.

a) Configure PCC on R1.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# pcc
Router(config-sr-te-pcc)# source-address ipv4 1.2.3.4 >>>>> This is the node address
Router(config-sr-te-pcc)# pce address ipv4 3.4.5.6 >>>>> This is the PCE server address
Router(config-sr-te-pcc)# commit
```

b) Configure PCE on R3.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# exit
Router(config)# pce
Router(config-pce)# address ipv4 3.4.5.6 >>>>> Configure the address only on a PCE server
Router(config-pce)# commit
```

Step 4 Configure BGP.

Configure BGP on the routers to establish neighborship with EVPN. When you enable an SR policy on R1, use the **nexthop validation color-extcomm sr-policy** command to instruct BGP that, instead of nexthop reachability validation of BGP routes, the validation is done for SR policy-installed color nexthop addresses. When the nexthop address of such a route is reachable, the route is added to the routing table.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 1.2.3.4
Router(config-bgp)# nexthop validation color-extcomm sr-policy
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 2.3.4.5
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 3.4.5.6
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-af)# exit
!
```

Step 5 Configure SR color.

Example:

```
Router(config)# extcommunity-set opaque color1
Router(config-ext)# 1
Router(config-ext)# end-set
Router(config)# extcommunity-set opaque color2
Router(config-ext)# 2
Router(config-ext)# end-set
Router(config)# extcommunity-set opaque color3
Router(config-ext)# 3
Router(config-ext)# end-set
Router(config)# extcommunity-set opaque color4
```

```
Router(config-ext)# 4
Router(config-ext)# end-set
```

Step 6 Configure EVPN route policy.

Example:

```
Router(config)# route-policy route_policy_1
Router(config-rpl)# if evpn-route-type is 1 then
Router(config-rpl-if)# set extcommunity color color1
Router(config-rpl-if)# elseif evpn-route-type is 3 then
Router(config-rpl-elseif)# set extcommunity color3
Router(config-rpl-elseif)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# route-policy route_policy_2
Router(config-rpl)# if evpn-route-type is 1 then
Router(config-rpl-if)# set extcommunity color color2
Router(config-rpl-if)# elseif evpn-route-type is 3 then
Router(config-rpl-elseif)# set extcommunity color4
Router(config-rpl-elseif)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
```

Configure EVPN route policy for tail-end coloring by exporting EVPN policy to color EVPN type 1 or 3 route. At the head-end, ODN uses color advertised by tail-end to create SR-TE tunnel.

Example:

```
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# route-policy export route_policy_1
```

Configure EVPN route policy for head-end coloring by importing EVPN policy to color EVPN type 1 or 3 route at the head-end. ODN uses color configured by head-end to create SR-TE tunnel.

Example:

```
Router(config)# evpn
Router(config-evpn)# evi 2
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# route-policy import route_policy_2
```

Example:

Step 7 Configure EVPN ELAN over SR-TE policy.

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 Bundle-Ether3.1
Router (config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 1
Router (config-l2vpn-bg-bd-evi)# exit
Router (config-l2vpn-bg-bd)# 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 Bundle-Ether4.1
Router (config-l2vpn-bg-bd-ac)# exit
```

```
Router (config-l2vpn-bg-bd)# evi 2
Router (config-l2vpn-bg-bd-evi)# commit
```

Step 8 Use these show commands to verify EVPN ELAN over SR-TE using ODN configuration.

a) View PCE topology on R3.

Example:

```
Router# show pce ipv4 topology summary
Wed Jul 27 22:00:37.109 UTC

PCE's topology database summary:
--------------------------------

Topology nodes:                3
Prefixes:                      3
Prefix SIDs:
  Total:                       0
  Regular:                     0
  Strict:                      0
Links:
  Total:                      12
  EPE:                         0
Adjacency SIDs:
  Total:                      12
  Unprotected:                12
  Protected:                   0
  EPE:                         0

Private Information:
Lookup Nodes                   0
Consistent                    no

Update Stats (from IGP and/or BGP):
  Nodes added:                 7
  Nodes deleted:               0
  Links added:                32
  Links deleted:               0
  Prefix added:               47
  Prefix deleted:              0

Topology Ready Summary:
  Ready:                      yes
  PCEP allowed:               yes
  Last HA case:         migration
  Timer value (sec):          40
  Timer:
    Running: no   >>>>> Check if the timer is running.
```

b) View PCE configuration on R3.

This show command works only on the PCE server.

Example:

```
Router# show pce ipv4 peer
Wed Jul 27 22:02:00.421 UTC

PCE's peer database:
--------------------
Peer address: 1.2.3.4
  State: Up
  Capabilities: Stateful, Segment-Routing, Update, Instantiation, SRv6

Peer address: 3.4.5.6
```

```
  State: Up
  Capabilities: Stateful, Segment-Routing, Update, Instantiation, SRv6
```

c) View PCC configuration on both R1 and R3.

Example:

```
Router# show segment-routing traffic-eng pcc ipv4 peer
Wed Jul 27 22:01:47.566 UTC

PCC's peer database:
--------------------

Peer address: 3.4.5.6,
  Precedence: 255, (best PCE)
  State up
  Capabilities: Stateful, Update, Segment-Routing, Instantiation, SRv6


Router# show segment-routing traffic-eng policy
Wed Jul 27 22:03:47.253 UTC

SR-TE policy database
---------------------

Color: 1, End-point: 3.4.5.6
  Name: srte_c_1_ep_3.4.5.6
  Status:
    Admin: up  Operational: up for 00:31:53 (since Jul 27 21:31:54.148)
  Candidate-paths:
.........
    Preference: 100 (BGP ODN) (active)
      Requested BSID: dynamic
.........
      Dynamic (pce 3.4.5.6) (valid)
        Metric Type: IGP,   Path Accumulated Metric: 10
          24005 [Adjacency-SID, 42.0.0.2 - 42.0.0.1]
  Attributes:
    Binding SID: 24021
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0


Router# show segment-routing traffic-eng forwarding policy color 1 endpoint ipv4 3.4.5.6 detail
Wed Jul 27 22:08:09.961 UTC

SR-TE Policy Forwarding database
--------------------------------

Color: 1, End-point: 3.4.5.6
  Name: srte_c_1_ep_3.4.5.6
  Binding SID: 24021
  Active LSP:
    Candidate path:
      Preference: 100 (BGP ODN)
    Local label: 24020
    Segment lists:
      SL[0]:
        Name: dynamic
        Switched Packets/Bytes: 0/0
        Paths:
```

```
    Path[0]:
      Outgoing Label: Pop
      Outgoing Interfaces: Bundle-Ether3
      Next Hop: 42.0.0.1
      Switched Packets/Bytes: 0/0
      FRR Pure Backup: No
      ECMP/LFA Backup: No
      Internal Recursive Label: Unlabelled (recursive)
      Label Stack (Top -> Bottom): { Pop }
      Path-id: 1, Weight: 1

Policy Packets/Bytes Switched: 0/0
```

EVPN ELAN automated steering to flex-algorithms

EVPN ELAN automated steering to flex-algorithms is a networking capability that

- integrates EVPN ELAN with SR-TE
- applies automated steering policies to direct EVPN traffic, and
- utilizes Segment Routing flex-algorithm paths for constraint-based path computation.

Table 42: Feature History Table

Feature Name	Release Information	Feature Description
EVPN ELAN automated steering to flex-algorithms	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 automatically steer EVPN ELAN traffic onto paths computed by Segment Routing flex-algorithms. Flex-algorithms enable the definition of custom routing computations in the IGP (IS-IS or OSPF) based on specific network constraints, such as minimizing latency or maximizing bandwidth. Automated steering then applies policies to classify EVPN traffic and direct it to the appropriate flex-algorithm path, ensuring that different types of EVPN services can utilize network resources optimally according to their requirements.

EVPN ELAN integration with SR-TE automated steering

This feature integrates EVPN ELAN, a Layer 2 VPN service, with SR-TE using automated steering to optimize traffic routing. This capability enables automated, policy-based classification of EVPN ELAN traffic and directs it onto specific SR-TE paths computed by flex-algorithms.

Core functionality includes:

- Automated traffic steering: EVPN ELAN traffic is classified based on configured policies and automatically steered onto designated SR-TE paths.

- Flex-algorithm integration: Flex-algorithms compute paths using user-defined constraints such as low latency, high bandwidth, or path disjointness.
- EVPN ELAN support: These steering capabilities apply directly to EVPN ELAN services, improving service differentiation and resource utilization.

Key features:

- Constraint-based routing: Flex-algorithms allow multiple independent shortest path first (SPF) computations within the IGP (IS-IS or OSPF), each with unique constraints.
- Policy-driven steering: Automated steering policies classify EVPN traffic by criteria like EVI, MAC address, or IP prefix and map it to specific flex-algorithm IDs.
- Dynamic path selection: Paths are dynamically computed and updated by the IGP based on Flex-algorithm definitions and real-time network topology.
- Network slicing: Supports creation of virtual network slices with distinct routing behaviors tailored for different EVPN services.

Benefits of EVPN ELAN automated steering to flex-algorithms

- Optimized resource utilization: Ensures that critical EVPN services use paths that meet their specific requirements, for example, latency-sensitive traffic on low-latency paths.
- Enhanced service differentiation: Provides a powerful mechanism to offer differentiated services within the same network infrastructure.
- Simplified traffic engineering: Automates complex traffic engineering decisions, reducing manual configuration and operational overhead.
- Increased network agility: Allows for rapid adaptation to changing network conditions and service demands.
- Scalability: Leverages the inherent scalability of Segment Routing for managing a large number of paths and services.

How EVPN ELAN automated steering to flex-algorithms works

EVPN ELAN automated steering to flex-algorithms enables the dynamic routing of EVPN traffic over paths computed by Segment Routing flex-algorithms, allowing for constraint-based traffic engineering and optimized resource utilization based on service requirements.

Summary

The key components involved in the process are:

- EVPN ELAN: Provides Layer 2 VPN services, including MAC/IP route advertisement and forwarding.
- SR-TE: The underlying technology for source routing, enabling explicit path control.
- Flex-algorithm (flex-algo): A Segment Routing extension that allows the definition and advertisement of multiple constraint-based shortest path computations within the IGP.
- Automated steering (AS): A policy-driven mechanism to classify traffic and automatically direct it onto specific SR-TE paths, including flex-algo paths.

- IGP (IS-IS/OSPF): Advertises flex-algorithm definitions, topology, and flex-algo Segment Identifiers (SIDs).
- BGP EVPN: Distributes EVPN routes (MAC/IP routes) and can carry information relevant for steering.
- Ingress PE router: Classifies incoming EVPN traffic and applies the automated-steering policy to select the appropriate flex-algo path.

This process ensures that EVPN ELAN traffic is automatically and dynamically routed over paths that meet specific performance or resource utilization requirements, leading to optimized network resource allocation, enhanced service differentiation, and improved operational efficiency.

Workflow

These stages describe how EVPN ELAN automated steering to flex-algorithms works.

1. Flex-algorithm definition and advertisement: Network administrators define one or more flex-algorithms within the IGP (IS-IS or OSPF). Each flex-algorithm includes specific constraints such as, metric type, excluded SIDs, and affinity bits. The IGP then advertises these flex-algorithm definitions and computes corresponding flex-algo SIDs across the network.
2. EVPN route distribution: BGP EVPN distributes EVPN routes throughout the network, providing reachability information for EVPN endpoints.
3. Automated steering policy configuration: On the ingress PE router, automated steering policies are configured. These policies define criteria for classifying EVPN ELAN traffic, for example, based on EVI, MAC address, source/destination IP address and map the classified traffic to a specific flex-algorithm ID.
4. Path computation and SID stack generation: When EVPN ELAN traffic arrives at the ingress PE and matches an automated steering policy, the ingress PE identifies the target flex-algorithm, and then computes the path to the destination based on the selected flex-algorithm's constraints and generates the corresponding SR-TE SID stack.
5. Traffic forwarding: The EVPN ELAN traffic is encapsulated with the computed flex-algo SID stack and forwarded along the constraint-based path. The intermediate Segment Routing-enabled routers forward the traffic purely based on the SIDs in the stack.
6. Dynamic adaptation: If network conditions change, for example, link failures, congestion or the IGP re-computes flex-algorithm paths, the ingress PE dynamically adapts the SID stack for affected traffic flows, ensuring continuous adherence to the steering policy and flex-algorithm constraints.

Configure EVPN ELAN automated steering to flex-algorithms

Demonstrate and implement automated EVPN traffic steering by associating an Ethernet Virtual Instance (EVI) with flex-algo-based SR-MPLS policies using IS-IS and color extended communities.

Automated steering to flex-algo-based SR policies leverages the color extcommunity, so when EVPN routes are imported with a color matching an on-demand SR policy, the SR policy using flex-algo is automatically selected for that flow.

Procedure

Step 1 Configure IS-IS with SR-MPLS and flex-algorithms.

Example:

```
Router# configure
Router(config)# router isis 100
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# net 49.0001.0000.0001.00
Router(config-isis)# instance 100
Router(config-isis)# flex-algo 128
Router(config-isis-flex-algo)# metric-type te! Use TE metric for algorithm 128
Router(config-isis-flex-algo)# prefix-metric ! Advertise prefix-SIDs for algo 128
Router(config-isis-flex-algo)# advertise-definition ! Advertise flex-algo definition
```

Step 2 Configure SR-TE with on-demand policy for flex-algorithms.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# on-demand color 128
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn)# exit
Router(config-sr-te-color)# constraints
Router(config-sr-te-color-const)# segments
Router(config-sr-te-color-const-seg)# sid-algorithm 128
Router(config-sr-te-color-const-seg)# commit
```

Step 3 Define color (Extcommunity) and route policy for EVPN.

Example:

```
Router(config)# extcommunity-set opaque color-128
Router(config-ext)# 128
Router(config-ext)# end-set
Router(config)# route-policy SET-EVPN-COLOR
Router(config-rpl)# if evpn-route-type is 2 then
Router(config-rpl-if)# set extcommunity color color-128
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
```

Step 4 Configure BGP EVPN and apply the route policy.

Example:

```
Router(config)# router bgp 65000
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# neighbor 10.0.0.2
Router(config-bgp-nbr)# remote-as 65000
Router(config-bgp-nbr)# route-policy SET-EVPN-COLOR out ! Apply color when advertising EVPN routes
```

Step 5 Enable EVPN ELAN service.

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 Bundle-Ether3
Router (config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 2
Router (config-l2vpn-bg-bd-evi)# root
Router(config)# evpn
Router(config-evpn)# evi 2
```

VPWS over SR-TE policy

A virtual private wire service (VPWS) is a Layer 2 VPN service that

- connects two locations through a PW
- encapsulates PW traffic within an MPLS label stack used for routing, and
- uses SR-TE policies to control the path the PW takes through the network instead of relying on Label Distribution Protocol (LDP).

For more information on SR-TE policy, see the *Configure SR-TE Policies* section in the *Segment Routing Configuration Guide for Cisco 8000 Series Routers, IOS XR*.

Table 43: Feature History Table

Feature Name	Release Information	Feature Description
VPWS over SR-TE Policy	Release 26.2.1	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.
VPWS over SR-TE Policy	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *The VPWS over SR-TE policy functionality is now extended to the Cisco 8712-MOD-M routers.
VPWS over SR-TE Policy	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The VPWS over SR-TE policy functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
VPWS over SR-TE Policy	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *The VPWS over SR-TE policy functionality is now extended to routers with the 88-LC1-36EH line cards.

Feature Name	Release Information	Feature Description
VPWS over SR-TE Policy	Release 7.7.1	Using the SR-TE policy, you can now set the preferred path between two endpoints for Virtual Private Wire Service (VPWS). This gives you the advantage of a reduced number of dedicated networks to provision IP and VPN services across SR-TE tunnels. The VPWS is a method to provide Ethernet-based point to point communication over MPLS or IP networks.

VPWS preferred path using SR-TE policy

Using SR-TE policies, you can set a preferred path between two endpoints for VPWS. This approach reduces the need to provision multiple dedicated networks for IP and VPN services across SR-TE tunnels, simplifying network management and improving efficiency.

VPWS provides Ethernet-based point-to-point communication over MPLS or IP networks. Cisco 8000 series routers support VPWS implementation through these methods:

- EVPN VPWS
- LDP VPWS
- EVPN VPWS On-Demand nexthop

How segment routing traffic engineering steers pseudowire traffic

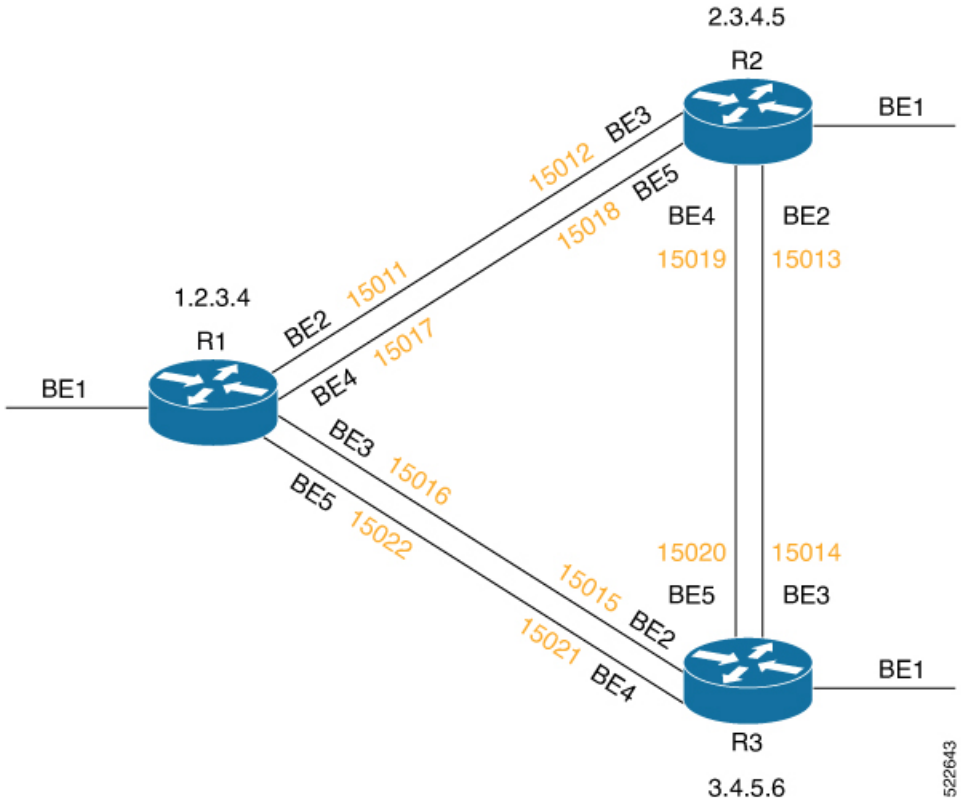
Summary

The key components involved in the process are:

- Pseudowire (PW): A virtual point-to-point connection between routers that carries encapsulated traffic.
- Routers R1, R2, and R3: Network devices where R1 and R3 are endpoints of the PW, and R2 is an intermediate router.
- Segment Routing Traffic Engineering (SR-TE) policy: A routing policy that defines a preferred path for traffic steering.

SR-TE steers pseudowire traffic by enforcing a preferred path through an intermediate router instead of the default direct route. This approach improves network resource utilization and traffic control.

Workflow



These stages describe how SR-TE steers pseudowire traffic.

1. By default, the PW between routers R1 and R3 takes the direct path from R1 to R3.
2. You configure an SR-TE policy with a preferred path that specifies traffic should be steered through router R2.
3. The SR-TE policy enforces the traffic to follow the defined path, steering the PW traffic from R1 to R3 via R2 instead of the direct route.

Result

This process enables controlled traffic engineering by steering pseudowire traffic along a preferred path, improving network resource utilization and traffic management.

Decoupled mode for L2VPN and EVPN VPWS services

Decoupled mode is a network capability in L2VPN and EVPN VPWS that

- improves network reliability and resilience
- enables PW to function independently from the AC, maintaining PW traffic continuity despite AC failure, and
- employs the xconnect, a virtual connection that links the AC and PW segments, to efficiently manage state transitions.

Table 44: Feature History Table

Feature Name	Release Information	Feature Description
Decoupled mode for L2VPN and EVPN VPWS services	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
Decoupled mode for L2VPN and EVPN VPWS services	Release 25.2.1	Introduced in this release on: Fixed Systems (8200, 8700, (8010 [ASIC: A100])(select variants only*)); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100, K100]) Decoupled mode improves fault tolerance by allowing the PE router to maintain the PW in an active state independently of the AC status. Unlike the traditional coupled mode, which requires both AC and PW to be active for traffic flow, decoupled mode ensures uninterrupted PW traffic even during AC failures. *This feature is now supported on the Cisco 8011-4G24Y4H-I routers. The feature introduces these changes: **CLI**: • **decoupled-mode** **YANG Data Model**: • `Cisco-IOS-XR-l2vpn-cfg.yang` • `Cisco-IOS-XR-l2vpn-oper.yang` (see GitHub, YANG Data Models Navigator)

Decoupled mode states

Decoupled mode uses the XTC to manage traffic flow dynamically by transitioning between states based on the operational status of the AC and PW:

- Bound state: Both AC and PW segments are operational. The XC is bound, allowing traffic to flow seamlessly between the AC and PW segments.
- Unbound state: Either the AC or PW segment fails. The XC enters an unbound state and drops any pending data on the AC. If the AC fails, the PE router brings down the PW connection, stopping traffic flow or rerouting it through alternative paths. If the PW fails, the XC brings down the AC connection and stops traffic flow.

- Bound down state: When the AC segment fails, the XC keeps the PW active, maintaining PW traffic flow between PE routers. If the PW segment fails, the XC enters an unbound state, brings down the AC segment, and stops traffic flow.

Before Release 25.2.1, the default configuration for VPWS was the coupled mode, which supported only bound and unbound XC states. For more information, see the VPWS over SR-TE policy, on page 301.

From Release 25.2.1, you can configure the decoupled mode in addition to the existing coupled mode. Decoupled mode allows PW traffic to continue forwarding even when the AC experiences a failure, enhancing service resilience.

Limitations for decoupled mode

The router does not support decoupled mode for these configurations:

- PWHE AC segment
- Multi-segment PW
- Local switching between AC
- BGP auto-discovery for PW
- Multihoming EVPN circuits

How decoupled mode works

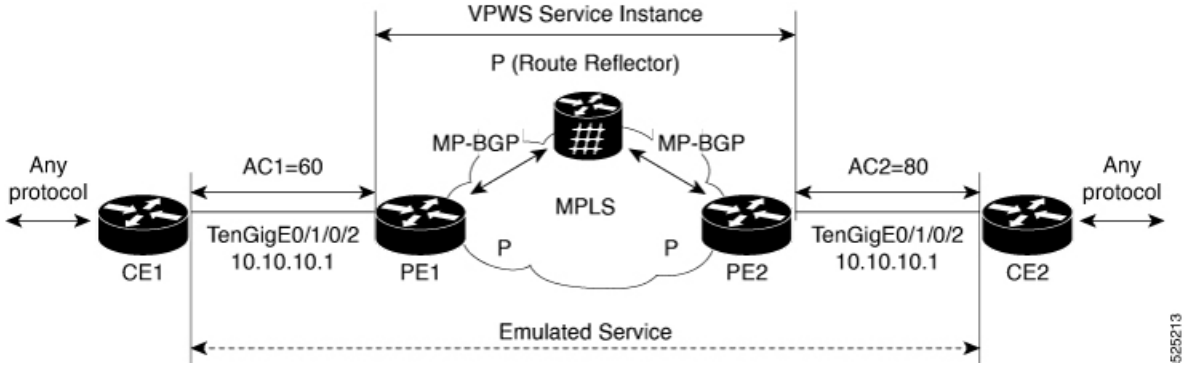
Summary

The key components involved in the process are:

- Customer edge device (CE1 and CE2): End devices connected to the provider network.
- Provider edge router (PE1 and PE2): Network devices managing PW and AC connections.
- Route reflector: Facilitates routing information exchange between PE1 and PE2.

Decoupled Mode improves network resilience by enabling PE routers to independently manage PW and AC connections. This separation ensures continuous PW traffic and dynamic fault management between CE devices and PE routers.

Workflow



These stages describe how the decoupled mode works.

The stages described here are not sequential. They illustrate different aspects of how the decoupled mode works, and may occur independently or in varying order depending on the scenario.

1. Scenario 1: Both AC and PW are active
   - CE1 sends a packet to PE1 via the active AC.
   - PE1 looks up the packet destination and forwards it over the active PW to PE2.
   - PE1 encapsulates and transmits the packet to PE2.
   - PE2 receives the packet, determines the next hop, and forwards it to CE2 via the active AC.
   - CE2 successfully receives the packet.
2. Scenario 2: AC experiences a failure
   - CE1 sends a packet to PE1.
   - PE1 detects the AC failure but keeps the PW operational without signaling it inactive.
   - The cross-connect (XC) transitions to a bound-down state, allowing PW traffic to continue.
   - PE1 forwards the packet over the active PW to PE2.
   - PE2 forwards the packet to CE2 via the active AC.
   - CE2 successfully receives the packet.
3. Scenario 3: PW experiences a failure
   - CE1 sends a packet to PE1 via the active AC.
   - PE1 detects the PW failure and signals the AC to become inactive.
   - The route reflector updates PE2 about the AC state change.
   - PE2 blocks traffic from CE2, and PE1 blocks traffic from CE1.
   - The XC drops any pending data on the AC, stopping all traffic flow.

Result

This process ensures network resilience by maintaining PW traffic independently of AC status, allowing dynamic fault handling and uninterrupted communication between CE devices through PE routers.

Configure decoupled mode for L2VPN and EVPN VPWS services

Enable decoupled mode on PE routers to maintain PW activity even if the AC fails, ensuring continuous Layer 2 VPN service.

Decoupled mode allows a point-to-point pseudowire service to extend a Layer 2 network over a Layer 3 IP and MPLS infrastructure, improving service resilience.

Procedure

Step 1 Enable L2VPN services and create a point-to-point pseudowire service.

Enable L2VPN services and configure a p2p pseudowire service that extends a Layer 2 network across a Layer 3 IP and MPLS network.

Example:

```
Router# configuration
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group g1
Router(config-l2vpn-xc)# p2p xc1
```

Step 2 Enable decoupled mode on the point-to-point pseudowire service.

Enable decoupled mode within p2p services, which keeps the PW active even if the AC experiences a failure.

Example:

```
Router(config-l2vpn-xc-p2p)# decoupled-mode
```

Step 3 Configure the interface and establish the pseudowire with the appropriate neighbor and service ID.

Enable interface and configure PW, ipv4 , EVPN or MPLS services for establishing the required L2VPN service over a network.

Example:

```
Router(config-l2vpn-xc-p2p)# interface TenGigE0/1/0/2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 service-id 10011
Router(config-l2vpn-xc-p2p)# commit
```

Step 4 Use the **show l2vpn** command to verify the XC, AC, and PW states.

Example:

In this example, both the AC and PW are in the bound state, and the XC is in the up state.

```
Router# show l2vpn xconnect group g1 xc-name p1 detail
Group g1, XC p1, state is up; Interworking none
Decoupled mode: Enabled
  AC: GigabitEthernet0/0/0/2.1, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1504; XC ID 0x1; interworking none
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor 1.1.1.1, PW ID 1, state is up ( established )
    PW class not set, XC ID 0xfff80013
    Encapsulation MPLS, protocol LDP
    Source address 2.2.2.2
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    Ignore MTU mismatch: Disabled
    Transmit MTU zero: Disabled
    LSP : Up
    Nexthop type: IPV4 1.1.1.1
```

Example:

In this example, due to a local AC fault, the AC is in the down state and the PW is in the up state.

```
Router# show l2vpn xconnect group g1 xc-name p1 detail

Group g1, XC p1, state is down; Interworking none
Decoupled mode: Enabled
  AC: GigabitEthernet0/0/0/2.1, state is down (Admin)
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1504; XC ID 0x1; interworking none
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor 1.1.1.1, PW ID 1, state is up ( established )
    PW class not set, XC ID 0xfff80013
    Encapsulation MPLS, protocol LDP
    Source address 2.2.2.2
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    Ignore MTU mismatch: Disabled
    Transmit MTU zero: Disabled
    LSP : Up
    Nexthop type: IPV4 1.1.1.1
```

Step 5 Use the **show l2vpn forwarding xconnect <xc-id> detail location <location>** command to verify the xconnect and PW binding state.

Example:

In this example, the XC is in the down state, both AC and PW are in a bound state, and the segment can still carry traffic across the PW despite the AC failure.

```
Router# show l2vpn forwarding xconnect 0x1 detail location 0/0/CPU0

Local interface: GigabitEthernet0/0/0/2.1, Xconnect id: 0x1, Status: down
  Segment 1
    AC, GigabitEthernet0/0/0/2.1, status: Bound
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
  Segment 2
    MPLS, Destination address: 1.1.1.1, pw-id: 1, status: Bound
    Pseudowire label: 24014
    Control word disabled
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
    PD System Data:  0x00000000, 0x00000000, 0x00000000, 0x01000000, 0x22000000,
                     0x00000000, 0x00000000
```

Example:

In this example, the XC is in the up state and both AC and PW are in bound state.

```
Router# show l2vpn forwarding xconnect 0x1 detail location 0/0/CPU0

Local interface: GigabitEthernet0/0/0/2.1, Xconnect id: 0x1, Status: up
  Segment 1
    AC, GigabitEthernet0/0/0/2.1, status: Bound
    Statistics:
```

```
    packets: received 0, sent 0
    bytes: received 0, sent 0
Segment 2
  MPLS, Destination address: 1.1.1.1, pw-id: 1, status: Bound
  Pseudowire label: 24014
  Control word disabled
```

VPWS PW over preferred SR-TE using statically configured SR policy

A preferred path configuration is a feature that

- allows setting the preferred path between two endpoints for EVPN VPWS PW or LDP VPWS PW using a statically configured policy
- supports both bundle AC and physical AC, and
- enables control over the forwarding path to optimize network performance and reliability.

Restrictions for VPWS PW over preferred SR-TE using statically configured SR policy

- EVPN VPWS SR policy is not supported on EVPN VPWS dual homing.
- EVPN validates if the route is for a single home nexthop, otherwise it issues an error message about a dangling SR TE policy, and continues to set up EVPN-VPWS without it. EVPN relies on ESI value being zero to determine if this is a single home or not. If the AC is a Bundle-Ether interface running LACP then you must manually configure the ESI value to zero to overwrite the auto-sense ESI as EVPN VPWS multihoming is not supported.

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

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

Configure VPWS pseudowire over preferred SR-TE using statically configured SR policy

Ensure successful configuration of VPWS PW over a preferred SR-TE path using a statically configured SR policy for both EVPN VPWS and LDP VPWS.

This task applies when you want to leverage statically configured SR-TE policies to control the preferred path for VPWS pseudowires in your network, enhancing traffic engineering and path selection.

Before you begin

- Confirm that IS-IS is running as the IGP.
- Ensure MPLS and SR features are supported and enabled on your devices.
- For LDP VPWS, configure MPLS LDP as a baseline for label distribution.

Procedure

Step 1 Configure SR in IS-IS, adjacency-SID, and prefix-SID.

Enable SR in IS-IS and configure adjacency-SID on BE2, BE3, BE4, and BE5.

a) Enable SR in IS-IS

Example:

```
Router(config)# router isis isp
Router(config-isis)# net 01.0000.0000.0001.00
Router(config-isis)# distribute link-state instance-id 32 level 2 throttle 5
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# mpls traffic-eng level-1
Router(config-isis-af)# mpls traffic-eng router-id 1.2.3.4
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# segment-routing prefix-sid-map advertise-local
```

b) Configure adjacency-SID on BE2.

Example:

```
Router(config)# router isis isp
Router(config-isis)# interface Bundle-Ether2
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# adjacency-sid index 11
Router(config-isis-if-af)# adjacency-sid absolute 15011
```

c) Configure adjacency-SID on BE3.

Example:

```
Router(config)# router isis isp
Router(config-isis)# interface Bundle-Ether3
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# adjacency-sid index 16
Router(config-isis-if-af)# adjacency-sid absolute 15016
```

d) Configure adjacency-SID on BE4.

Example:

```
Router(config)# router isis isp
Router(config-isis)# interface Bundle-Ether4
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# adjacency-sid index 17
Router(config-isis-if-af)# adjacency-sid absolute 15017
```

e) Configure adjacency-SID on BE5.

Example:

```
Router(config)# router isis isp
Router(config-isis)# interface Bundle-Ether5
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# adjacency-sid index 22
Router(config-isis-if-af)# adjacency-sid absolute 15022
```

f) Configure prefix-SID.

Example:

```
Router(config)# router isis isp
Router(config-isis)# interface Loopback0
Router(config-isis-if)# passive
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# prefix-sid index 1001
```

Step 2 Configure SR on interfaces.

Configure SR on BE2, BE3, BE4, and BE5.

Example:

```
Router(config)# segment-routing
Router(config-sr)# adjacency-sid
Router(config-sr)# exit
Router(config)# interface Bundle-Ether2
Router(config-if)# address-family ipv4 unicast
Router(config-if-af)# l2-adjacency-sid absolute 15011 next-hop 0.0.0.0
Router(config)# interface Bundle-Ether3
Router(config-if)# address-family ipv4 unicast
Router(config-if-af)# l2-adjacency-sid absolute 15016 next-hop 0.0.0.0
Router(config)# interface Bundle-Ether4
Router(config-if)# address-family ipv4 unicast
Router(config-if-af)# l2-adjacency-sid absolute 15017 next-hop 0.0.0.0
Router(config)# interface Bundle-Ether5
Router(config-if)# address-family ipv4 unicast
Router(config-if-af)# l2-adjacency-sid absolute 15022 next-hop 0.0.0.0
```

Step 3 Define segment lists for adjacency-SIDs

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment_list_r3_1
Router(config-sr-te-sl)# index 10 mpls label 15011
Router(config-sr-te-sl)# index 20 mpls label 15013
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment_list_r3_2
Router(config-sr-te-sl)# index 10 mpls label 15011
Router(config-sr-te-sl)# index 20 mpls label 15019
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment_list_r3_3
Router(config-sr-te-sl)# index 10 mpls label 15017
Router(config-sr-te-sl)# index 20 mpls label 15013
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list segment_list_r3_4
Router(config-sr-te-sl)# index 10 mpls label 15017
Router(config-sr-te-sl)# index 20 mpls label 15019
Router(config-sr-te-sl)# exit
```

Step 4 Configure static SR-TE policy.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy policy_r3_1
Router(config-sr-te-policy)# color 10 end-point ipv4 3.4.5.6
Router(config-sr-te-policy)# candidate-paths
```

```
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-pp-info)# explicit segment-list segment_list_r3_1
Router(config-sr-te-pp-info)# weight 10

Router(config-sr-te-pp-info)# explicit segment-list segment_list_r3_2
Router(config-sr-te-pp-info)# weight 10

Router(config-sr-te-pp-info)# explicit segment-list segment_list_r3_3
Router(config-sr-te-pp-info)# weight 10

Router(config-sr-te-pp-info)# explicit segment-list segment_list_r3_4
Router(config-sr-te-pp-info)# weight 10
```

Step 5 Configure EVPN VPWS over the statically configured SR-TE policy.

Example:

Note
Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to get the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name policy_r3_1

SR-TE policy database
---------------------
Color: 10, End-point: 3.4.5.6
 Name: sr-te policy srte_c_10_ep_3.4.5.6
```

Set the preferred path with the SR-TE policy name.

```
Router(config)# l2vpn
Router(config-l2vpn)# pw-class c1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_10_ep_3.4.5.6
Router(config-l2vpn-pwc-mpls)# root
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xg
Router(config-l2vpn-xc)# p2p xc200
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether1.200
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 2 target 3 source 4
Router(config-l2vpn-xc-p2p-pw)# pw-class c1
!
```

Step 6 Configure LDP VPWS over statically configured policy.

It is recommended to configure MPLS LDP as the baseline setup to ensure proper label distribution and network stability, particularly when handling scenarios involving unconfiguration and reconfiguration of loopback interfaces.

Example:

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 1.2.3.4
```

Attach the policy to the L2VPN instance.

Note
Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to get the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name policy_r3_1

SR-TE policy database
---------------------
Color: 10, End-point: 3.4.5.6
 Name: sr-te policy srte_c_10_ep_3.4.5.6
```

Set the preferred path with the SR-TE policy name.

```
Router(config)# l2vpn
Router(config-l2vpn)# pw-class c
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_10_ep_3.4.5.6
Router(config-l2vpn-pwc-mpls)# commit
Router(config-l2vpn-pwc-mpls)# root
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xg
Router(config-l2vpn-xc)# p2p xc100
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether1.100
Router(config-l2vpn-xc-p2p)# neighbor ipv4 3.4.5.6 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# pw-class c
```

Step 7 Use these show commands to verify the VPWS PW over preferred SR-TE using statically configured SR policy.

These commands help confirm the status and forwarding path of the VPWS pseudowire, ensuring it follows the intended static policy for optimized network performance and reliability.

- show l2vpn xconnect interface Bundle-Ether 1.100 detail
- show segment-routing traffic-eng policy color 10 endpoint ipv4 3.4.5.6 detail
- show segment-routing traffic-eng forwarding policy color 10 endpoint ipv4 3.4.5.6 detail

Example:

```
Router# show l2vpn xconnect interface Bundle-Ether 1.100 detail
Mon May 16 14:19:42.833 UTC

Group xg, XC xc100, state is up; Interworking none
  AC: Bundle-Ether1.100, state is up
  PW: neighbor 3.4.5.6, PW ID 100, state is up ( established )
    PW class c, XC ID 0xa0000001
    Encapsulation MPLS, protocol LDP
    Source address 1.2.3.4
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    Preferred path Active : SR TE srte_c_10_ep_3.4.5.6 (BSID:24011, IFH:0xf000014), Statically
configured, fallback enabled
    Ignore MTU mismatch: Disabled
    Transmit MTU zero: Disabled
    Tunnel : Up


Router# show segment-routing traffic-eng policy color 10 endpoint ipv4 3.4.5.6 detail
Mon May 16 14:02:16.550 UTC

SR-TE policy database
---------------------

Color: 10, End-point: 3.4.5.6
  Name: srte_c_10_ep_3.4.5.6
  Status:
```

```
    Admin: up  Operational: up for 00:05:09 (since May 16 13:57:06.627)

Router# show segment-routing traffic-eng forwarding policy color 10 endpoint ipv4 3.4.5.6 detail
Mon May 16 14:04:49.061 UTC

SR-TE Policy Forwarding database
--------------------------------

 Color: 10, End-point: 3.4.5.6
  Name: srte_c_10_ep_3.4.5.6
  Binding SID: 24011
  Active LSP:
    Candidate path:
      Preference: 100 (configuration)
      Name: policy_r3_1
    Local label: 24021
    Segment lists:
      SL[0]:
       Name: segment_list_r3_1
        Switched Packets/Bytes: 0/0
        Paths:
          Path[0]:
            Outgoing Label: 15013
            Outgoing Interfaces: Bundle-Ether2
  .........

      SL[1]:
        Name: segment_list_r3_2
        Switched Packets/Bytes: 0/0
        Paths:
          Path[0]:
            Outgoing Label: 15019
            Outgoing Interfaces: Bundle-Ether2
  .........

      SL[2]:
        Name: segment_list_r3_3
        Switched Packets/Bytes: 0/0
        Paths:
          Path[0]:
            Outgoing Label: 15013
            Outgoing Interfaces: Bundle-Ether4
  .........

      SL[3]:
        Name: segment_list_r3_4
        Switched Packets/Bytes: 0/0
        Paths:
          Path[0]:
            Outgoing Label: 15019
            Outgoing Interfaces: Bundle-Ether4

  .........
  Policy Packets/Bytes Switched: 0/0
```

EVPN PW over preferred SR-TE using on-demand nexthop SR policy

An EVPN PW over preferred SR-TE using On-Demand Nexthop (ODN) SR policy is a network feature that

- enables dynamic selection of the optimal path for traffic forwarding from source to destination in point-to-point services
- leverages IOS XR Traffic Controller (XTC) for path computation and control, and
- supports both EVPN E-LAN and EVPN E-Line service types.

Multi-domain service provisioning with on-demand nexthop and segment routing traffic engineering

Provisioning multi-domain services such as Layer 2 VPN and Layer 3 VPN across routing domains introduces complexity and scalability challenges. The ODN feature with SR-TE addresses these challenges by delegating the computation of an end-to-end Label Switched Path (LSP) to a Path Computation Element (PCE). This approach avoids route redistribution by incorporating constraints and policies directly within the PCE.

ODN uses BGP dynamic SR-TE capabilities to add the computed path to the PCE. The PCE dynamically discovers and downloads the optimal end-to-end path based on specified requirements. ODN triggers an SR-TE auto-tunnel according to the configured BGP policy. The PCE maintains an up-to-date view of the network by learning real-time topology information through BGP and/or IGP, enabling precise and adaptive path computation for multi-domain services.

Note The maximum number of supported auto-provisioned SR-TE policies is 1000.

IOS XR Traffic Controller

The PCE defines a set of procedures that enable a path computation client (PCC) to delegate control of head-end tunnels it sources to a PCE peer. This delegation allows the PCE peer to request updates and modifications to the parameters of label-switched paths (LSPs) controlled by the PCC. Additionally, the PCE can initiate path computations and perform network-wide orchestration to optimize traffic engineering.

Key aspects include:

- The PCC reports and delegates control of its head-end tunnels to the PCE peer.
- The PCE peer requests the PCC to update and modify LSP parameters.
- The PCC permits the PCE to initiate path computations.
- The PCE performs network-wide orchestration based on the delegated control.

Configure EVPN VPWS PW over preferred SR-TE using ODN SR policy

Configure EVPN VPWS pseudowires over a preferred SR-TE path using ODN SR policy.

This task applies when you want to leverage PCE-based path computation and dynamic SR-TE policies to optimize EVPN VPWS pseudowire forwarding.

Procedure

Step 1 Configure SR in IS-IS and configure interfaces BE2, BE3, BE4, and BE5.

a) Enable SR in IS-IS.

Example:

```
Router(config)# router isis isp
Router(config-isis)# net 01.0000.0000.0001.00
Router(config-isis)# distribute instance-id 32 level 2 throttle 5
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# mpls traffic-eng level-1
Router(config-isis-af)# mpls traffic-eng router-id 1.2.3.4
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# segment-routing prefix-sid-map advertise-local
Router(config-isis-af)# exit
```

b) Configure interface BE2.

Example:

```
Router(config-isis)# interface Bundle-Ether2
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# exit
Router(config-isis-if)# exit
```

c) Configure interface BE3.

Example:

```
Router(config-isis)# interface Bundle-Ether3
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# exit
Router(config-isis-if)# exit
```

d) Configure interface BE4.

Example:

```
Router(config-isis)# interface Bundle-Ether4
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# exit
```

e) Configure interface BE5.

Example:

```
Router(config-isis)# interface Bundle-Ether5
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# exit
```

f) Configure Loopback interface.

Example:

```
Router(config-isis)# interface Loopback0
Router(config-isis-if)# passive
Router(config-isis-if)# address-family ipv4 unicast
```

Step 2 Configure SR-TE policy.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# on-demand color 1
```

```
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn)# pcep
Router(config-sr-te-color-dyn-pce)# exit
Router(config-sr-te-color-dyn)# metric type igp
Router(config-sr-te-color-dyn)# exit
Router(config-sr-te-color)# exit
Router(config-sr-te)# on-demand color 2
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn)# pcep
Router(config-sr-te-color-dyn-pce)# exit
Router(config-sr-te-color-dyn)# metric type igp
Router(config-sr-te-color-dyn)# exit
Router(config-sr-te-color)# exit
```

Step 3 Configure PCE and PCC.

a) Configure PCC on R1.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# pcc
Router(config-sr-te-pcc)# source-address ipv4 1.2.3.4 >>>>> This is the node address
Router(config-sr-te-pcc)# pce address ipv4 3.4.5.6 >>>>> This is the PCE server address
Router(config-sr-te-pcc)# commit
```

b) Configure PCE on R3.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# exit
Router(config)# pce
Router(config-pce)# address ipv4 3.4.5.6 >>>>> Configure the address only on a PCE server
Router(config-pce)# commit
```

Step 4 Configure BGP.

Configure BGP on the routers to establish neighborship with EVPN. When you enable an SR policy on R1, use the **nexthop validation color-extcomm sr-policy** command to instruct BGP that, instead of nexthop reachability validation of BGP routes, the validation is done for SR policy-installed color nexthop addresses. When the nexthop address of such a route is reachable, the route is added to the routing table.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 1.2.3.4
Router(config-bgp)# nexthop validation color-extcomm sr-policy
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 2.3.4.5
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 3.4.5.6
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-af)# exit
!
```

Step 5 Configure SR color.

Example:

```
Router(config)# extcommunity-set opaque color1
Router(config-ext)# 1
Router(config-ext)# end-set
Router(config)# extcommunity-set opaque color2
Router(config-ext)# 2
Router(config-ext)# end-set
```

Step 6 Configure EVPN route policy.

Example:

```
Router(config)# route-policy route_policy_1
Router(config-rpl)# if evpn-route-type is 1 then
Router(config-rpl-if)# set extcommunity color color1
Router(config-rpl-if)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# route-policy route_policy_2
Router(config-rpl)# if evpn-route-type is 1 then
Router(config-rpl-if)# set extcommunity color color2
Router(config-rpl-if)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
```

Configure EVPN route policy for tail-end coloring by exporting EVPN policy to color EVPN type 1 route. At the head-end, ODN uses color advertised by tail-end to create SR-TE tunnel.

Example:

```
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# route-policy export route_policy_1
```

Configure EVPN route policy for head-end coloring by importing EVPN policy to color EVPN type 1 route at the head-end. ODN uses color configured by head-end to create SR-TE tunnel.

Example:

```
Router(config)# evpn
Router(config-evpn)# evi 2
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# route-policy import route_policy_2
```

Step 7 Configure EVPN VPWS over SR-TE policy.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xg
Router(config-l2vpn-xc)# p2p xc100
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether1.100
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 1 source 2
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# p2p xc200
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether1.200
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 2 target 3 source 4
```

Step 8 Use these show commands to verify EVPN VPWS PW over SR-TE using ODN configuration.

a) View PCE topology on R3.

Example:

```
Router# show pce ipv4 topology summary
Wed Jul 27 22:00:37.109 UTC

PCE's topology database summary:
--------------------------------

Topology nodes:                3
Prefixes:                      3
Prefix SIDs:
  Total:                       0
  Regular:                     0
  Strict:                      0
Links:
  Total:                      12
  EPE:                         0
Adjacency SIDs:
  Total:                      12
  Unprotected:                12
  Protected:                   0
  EPE:                         0

Private Information:
Lookup Nodes                   0
Consistent                    no

Update Stats (from IGP and/or BGP):
  Nodes added:                 7
  Nodes deleted:               0
  Links added:                32
  Links deleted:               0
  Prefix added:               47
  Prefix deleted:              0

Topology Ready Summary:
  Ready:                      yes
  PCEP allowed:               yes
  Last HA case:         migration
  Timer value (sec):           40
  Timer:
    Running: no   >>>>> Check if the timer is running.
```

b) View PCE configuration on R3.

This show command works only on the PCE server.

Example:

```
Router# show pce ipv4 peer
Wed Jul 27 22:02:00.421 UTC

PCE's peer database:
--------------------
Peer address: 1.2.3.4
  State: Up
  Capabilities: Stateful, Segment-Routing, Update, Instantiation, SRv6

Peer address: 3.4.5.6
  State: Up
  Capabilities: Stateful, Segment-Routing, Update, Instantiation, SRv6
```

c) View PCC configuration on both R1 and R3.

Example:

```
Router# show segment-routing traffic-eng pcc ipv4 peer
Wed Jul 27 22:01:47.566 UTC

PCC's peer database:
--------------------

Peer address: 3.4.5.6,
  Precedence: 255, (best PCE)
  State up
  Capabilities: Stateful, Update, Segment-Routing, Instantiation, SRv6


Router# show l2vpn xconnect interface Bundle-Ether 1.100 detail
Mon Jul 25 19:19:01.443 UTC

Group xg, XC xc100, state is up; Interworking none
  AC: Bundle-Ether1.100, state is up
  EVPN: neighbor 3.4.5.6, PW ID: evi 1, ac-id 1, state is up ( established )
    XC ID 0xa0000002
    Encapsulation MPLS
    Encap type Ethernet, control word enabled
    Sequencing not set
    Preferred path Active : SR TE srte_c_1_ep_3.4.5.6 (BSID:24021, IFH:0xf000054), On-Demand,
fallback enabled
    Ignore MTU mismatch: Enabled
    Transmit MTU zero: Enabled
    Tunnel : Up


Router# show segment-routing traffic-eng policy
Wed Jul 27 22:03:47.253 UTC

SR-TE policy database
---------------------

Color: 1, End-point: 3.4.5.6
  Name: srte_c_1_ep_3.4.5.6
  Status:
    Admin: up  Operational: up for 00:31:53 (since Jul 27 21:31:54.148)
  Candidate-paths:
.........
    Preference: 100 (BGP ODN) (active)
      Requested BSID: dynamic
.........
      Dynamic (pce 3.4.5.6) (valid)
        Metric Type: IGP,   Path Accumulated Metric: 10
          24005 [Adjacency-SID, 42.0.0.2 - 42.0.0.1]
  Attributes:
    Binding SID: 24021
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0


Router# show segment-routing traffic-eng forwarding policy color 1 endpoint ipv4 3.4.5.6 detail
Wed Jul 27 22:08:09.961 UTC

SR-TE Policy Forwarding database
--------------------------------
```

```
Color: 1, End-point: 3.4.5.6
  Name: srte_c_1_ep_3.4.5.6
  Binding SID: 24021
  Active LSP:
    Candidate path:
      Preference: 100 (BGP ODN)
    Local label: 24020
    Segment lists:
      SL[0]:
        Name: dynamic
        Switched Packets/Bytes: 0/0
        Paths:
          Path[0]:
            Outgoing Label: Pop
            Outgoing Interfaces: Bundle-Ether3
            Next Hop: 42.0.0.1
            Switched Packets/Bytes: 0/0
            FRR Pure Backup: No
            ECMP/LFA Backup: No
            Internal Recursive Label: Unlabelled (recursive)
            Label Stack (Top -> Bottom): { Pop }
            Path-id: 1, Weight: 1

  Policy Packets/Bytes Switched: 0/0
```

Call admission control for L2VPN P2P services over circuit-style SR-TE policies

Call Admission Control (CAC) is a network control method that

- ensures available bandwidth and network resources are not overloaded by excessive traffic
- regulates traffic admission in L2VPN point-to-point (P2P) services, and
- operates specifically within circuit-style SR-TE policies.

Table 45: Feature History Table

Feature Name	Release Information	Feature Description
Call Admission Control for L2VPN P2P Services over Circuit-Style SR-TE Policies	Release 26.2.1	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.
Call Admission Control for L2VPN P2P Services over Circuit-Style SR-TE Policies	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Feature Name	Release Information	Feature Description
Call Admission Control for L2VPN P2P Services over Circuit-Style SR-TE Policies	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) * This feature is now supported on Cisco 8712-MOD-M routers.
Call Admission Control for L2VPN P2P Services over Circuit-Style SR-TE Policies	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
Call Admission Control for L2VPN P2P Services over Circuit-Style SR-TE Policies	Release 7.9.1	This feature allows you to configure guaranteed bandwidth for Layer 2 point-to-point (P2P) services steered over Circuit-Style SR-TE policies. This guaranteed bandwidth ensures that a Circuit-Style SR-TE policy has sufficient bandwidth to accommodate a Layer 2 P2P service. At the same time, it prevents a Layer 2 P2P service from being steered over a Circuit-Style SR-TE policy when there is insufficient available bandwidth.

Circuit-style SR-TE policies and CAC in L2VPN P2P services

Circuit-style SR-TE policies steer traffic along designated network paths tailored to the requirements of each L2VPN P2P service. Concurrently CAC manages the aggregate bandwidth demand of all active L2VPN P2P services to ensure it does not exceed the available capacity of network links.

By integrating CAC with circuit-style SR-TE policies, network administrators can optimize traffic routing while maintaining network capacity within limits. This combination enables efficient traffic management and prevents oversubscription of network resources, ensuring service quality and network stability.

For information about circuit-style SR-TE policies, refer to *Circuit-Style SR-TE Policies* in the *Segment Routing Configuration Guide for for Cisco 8000 Series Series Routers*.

Call admission control in circuit-style SR-TE policies

CAC prevents resource oversubscription in a network by allocating and reserving the necessary resources before enabling a service. CAC ensures that a aircuit-style SR-TE policy has sufficient bandwidth to support a Layer 2 P2P service.

CAC manages bandwidth by:

- Tracking the total bandwidth assigned to a Circuit-Style SR-TE policy.
- Monitoring the available bandwidth on the policy, considering all Layer 2 P2P services currently steered over it.
- Evaluating the bandwidth requested by a new Layer 2 P2P service seeking admission.

CAC prevents steering a Layer 2 P2P service over a circuit-style SR-TE policy if there is insufficient available bandwidth, thereby maintaining network resource integrity and service quality.

Limitations of CAC for L2VPN P2P services over circuit-style SR-TE policies

- LDP-signaled L2 P2P services and EVPN VPWS L2 P2P services are supported.
- If a PW is configured with a bandwidth value but is not configured with a preferred path, then the PW stays down with the "admitted bandwidth" set to 0. See L2VPN preferred path, on page 280.

Configure CAC for L2VPN P2P services over circuit-style SR-TE policies

Configure CAC to manage bandwidth for L2VPN P2P services using circuit-style SR-TE policies.

CAC ensures that bandwidth is reserved and controlled for pseudowires (PWs) in EVPN VPWS and LDP-signaled L2 P2P services to maintain service quality and prevent oversubscription.

Procedure

Step 1 Configure CAC for EVPN VPWS L2 P2P services.

To configure CAC for EVPN VPWS L2 P2P services, use the **admission-control bandwidth** *bandwidth* command. The range for *bandwidth* is from 1 to 4294967295 kbps.

Example:

```
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p evpn_vpws_1001
Router(config-l2vpn-xc-p2p)# interface TenGigE0/1/0/1.1001
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1001 target 10001 source 20001
Router(config-l2vpn-xc-p2p-pw)# admission-control bandwidth 24000
```

Step 2 Running configuration of CAC for EVPN VPWS L2 P2P services.

Example:

```
l2vpn
 xconnect group evpn_vpws
  p2p evpn_vpws_1001
   interface TenGigE0/1/0/1.1001
   neighbor evpn evi 1001 target 10001 source 20001
```

```
    admission-control bandwidth 24000
   !
  !
 !
!
```

Step 3 Use the **show l2vpn cac-db** command to display the total bandwidth of the policy, the available bandwidth, and the reserved bandwidth.

Example:

```
Router# show l2vpn cac-db

Policy Name: sr-srte_c_100_ep_3.3.3.3
  Total Bandwidth: 24000
  Available Bandwidth: 11000
  Reserved Bandwidth: 13000
  Service count: 1
  Pseudowire info:
  EVPN/AToM  VPN ID     AC ID      Reqd BW(kbps)   Alloc BW(kbps)  State
  ------------------------------------------------------------------------
  EVPN     1          1          13000              13000             NOT CONF
```