Configuring Virtual Private LAN Services

Virtual Private LAN Services (VPLS) enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider.

This module explains VPLS and how to configure it.

Prerequisites for Virtual Private LAN Services

Before you configure Virtual Private LAN Services (VPLS), ensure that the network is configured as follows:

- Configure IP routing in the core so that provider edge (PE) devices can reach each other via IP.
- Configure Multiprotocol Label Switching (MPLS) in the core so that a label switched path (LSP) exists between PE devices.
- Configure a loopback interface for originating and terminating Layer 2 traffic. Ensure that PE devices can access the loopback interface of the other device. Note that the loopback interface is not required in all cases. For example, tunnel selection does not need a loopback interface when VPLS is directly mapped to a traffic engineering (TE) tunnel.

  Note VPLS over TE Tunnel/TE FRR is not supported on the Cisco ASR 900 RSP3 module.

- Identify peer PE devices and attach Layer 2 circuits to VPLS at each PE device.

Restrictions for Virtual Private LAN Services

The following general restrictions apply to all transport types under Virtual Private LAN Services (VPLS):

- If you do not enable the EFP feature template, then there is no traffic flow between EFP and VFI (when EFP is with Split Horizon group and VFI is default). But when you enable the EFP feature template, then there is traffic flow between EFP and VFI because of design limitations.
- Supported maximum values:
  - Total number of virtual forwarding instances (VFIs): 4096 (4 K)
  - Total number of VFIs on the Cisco ASR 900 RSP3 module: 4096 (3584 hubs and 512 Spokes)
  - Total number of VC on the Cisco ASR 900 RSP3: 8192 (4096 EOMPLS and 4096 VFIs)
  - Maximum neighbors per VFI on the Cisco ASR 900 RSP3: 64
- Effective with Cisco IOS XE Release 3.18.2SP, the RSP3 Module only supports VPLS over Port-channel (PoCH) and bridge domain interfaces (BDI).
- VPLS over TE tunnel/TE FRR is not supported on the RSP3 Module.
- Effective Cisco IOS XE Everest 16.6.1, for VPLS to work with labeled BGP (RFC3107) on the Cisco ASR 900 RSP3 module, you must enable the following command, without which you will receive object down failure in the console:

  ```
  router bgp [as-no]
  address-family ipv4
  bgp mpls-local-label
  ```

- Fragmentation is not supported for VPLS and VPWS traffic.

  Note TTL decrements on PE imposition for VPLS traffic.

- EoMPLS/XC statistics are not supported.
- L2VPN traffic is not load balanced for inner payload src-ip, dst-ip, src-dst-ip hashing algorithms in the egress PoCh interface. We recommend you to use other hashing algorithms like src-mac, dst-mac, src-dst-mac.
- Software-based data plane is not supported.
- The Border Gateway Protocol (BGP) autodiscovery process does not support dynamic, hierarchical VPLS.
- Load sharing and failover on redundant customer-edge-provider-edge (CE-PE) links are not supported.
- On the Cisco ASR 900 RSP3 module, VPLS imposition traffic always undergoes a recirculation in the hardware.
- Point to Multipoint (P2MP) Resource Reservation Protocol (RSVP) for MPLS Traffic Engineering (MPLS-TE) is not supported over VPLS on the Cisco RSP2 and RSP3 routers.
- Traffic drops are observed for lower sized MPLS pseudowire packets.

- If ECMP is established with same IGP next hops:
  - When VPLS circuit destination is learnt in IGP and if ECMP is established with same IGP next hops then VPLS traffic is load balanced based on VC label only if FAT PW is not enabled. If FAT PW is enabled, then the load balancing happens based on VC label and FAT PW label.

  If ECMP is established with different IGP next hops:
  - When VPLS circuit destination is learnt in IGP and if ECMP is established with different IGP next hops, VPLS traffic is not load balanced. Enabling FAT has no significance in this scenario.
  - When VPLS circuit destination is learnt in Labelled BGP and the BGP next hop is reachable through more than one different IGP next hops having equal cost (ECMP through different IGP next hops), VPLS traffic is load balanced based on VC label only if FAT PW is not configured. If FAT PW is enabled, then the load balancing happens based on VC label and FAT PW label, provided all of the following conditions are met:
    - The head end has learnt more than 15 Global IPv4 prefixes from the same BGP peer to which VPLS circuit ends.
    - BGP LU is configured to assign local label and advertise the same over the BGP

      If the above conditions are not met load balancing is not performed.

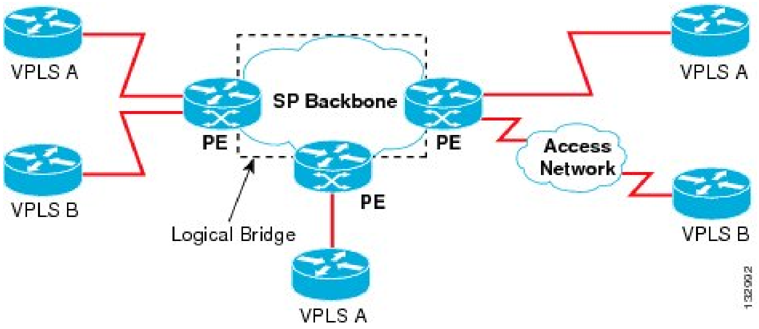
Information About Virtual Private LAN Services

VPLS Overview

Virtual Private LAN Services (VPLS) enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider. From the enterprise perspective, the service provider's public network looks like one giant Ethernet LAN. For the service provider, VPLS provides an opportunity to deploy another revenue-generating service on top of the existing network without major capital expenditures. Operators can extend the operational life of equipment in their network.

VPLS uses the provider core to join multiple attachment circuits together to simulate a virtual bridge that connects the multiple attachment circuits together. From a customer point of view, there is no topology for VPLS. All customer edge (CE) devices appear to connect to a logical bridge emulated by the provider core (see the figure below).

Figure 1: VPLS Topology



Full-Mesh Configuration

A full-mesh configuration requires a full mesh of tunnel label switched paths (LSPs) between all provider edge (PE) devices that participate in Virtual Private LAN Services (VPLS). With a full mesh, signaling overhead and packet replication requirements for each provisioned virtual circuit (VC) on a PE can be high.

You set up a VPLS by first creating a virtual forwarding instance (VFI) on each participating PE device. The VFI specifies the VPN ID of a VPLS domain, the addresses of other PE devices in the domain, and the type of tunnel signaling and encapsulation mechanism for each peer PE device.

The set of VFIs formed by the interconnection of the emulated VCs is called a VPLS instance; it is the VPLS instance that forms the logic bridge over a packet switched network. After the VFI has been defined, it needs to be bound to an attachment circuit to the CE device. The VPLS instance is assigned a unique VPN ID.

PE devices use the VFI to establish a full-mesh LSP of emulated VCs to all other PE devices in the VPLS instance. PE devices obtain the membership of a VPLS instance through static configuration using the Cisco IOS CLI.

A full-mesh configuration allows the PE device to maintain a single broadcast domain. When the PE device receives a broadcast, multicast, or unknown unicast packet on an attachment circuit (AC), it sends the packet out on all other ACs and emulated circuits to all other CE devices participating in that VPLS instance. The CE devices see the VPLS instance as an emulated LAN.

To avoid the problem of a packet looping in the provider core, PE devices enforce a “split-horizon” principle for emulated VCs. In a split horizon, if a packet is received on an emulated VC, it is not forwarded on any other emulated VC.

The packet forwarding decision is made by looking up the Layer 2 VFI of a particular VPLS domain.

A VPLS instance on a particular PE device receives Ethernet frames that enter on specific physical or logical ports and populates a MAC table similarly to how an Ethernet switch works. The PE device can use the MAC address to switch these frames into the appropriate LSP for delivery to the another PE device at a remote site.

If the MAC address is not available in the MAC address table, the PE device replicates the Ethernet frame and floods it to all logical ports associated with that VPLS instance, except the ingress port from which it just entered. The PE device updates the MAC table as it receives packets on specific ports and removes addresses not used for specific periods.

Static VPLS Configuration

Virtual Private LAN Services (VPLS) over Multiprotocol Label Switching-Transport Profile (MPLS-TP) tunnels allows you to deploy a multipoint-to-multipoint layer 2 operating environment over an MPLS-TP network for services such as Ethernet connectivity and multicast video. To configure static VPLS, you must specify a static range of MPLS labels using the **mpls label range** command with the **static** keyword.

H-VPLS

Hierarchical VPLS (H-VPLS) reduces signaling and replication overhead by using full-mesh and hub-and-spoke configurations. Hub-and-spoke configurations operate with split horizon to allow packets to be switched between pseudowires (PWs), effectively reducing the number of PWs between provider edge (PE) devices.

Note Split horizon is the default configuration to avoid broadcast packet looping.

Supported Features

Multipoint-to-Multipoint Support

In a multipoint-to-multipoint network, two or more devices are associated over the core network. No single device is designated as the Root node; all devices are considered as Root nodes. All frames can be exchanged directly between the nodes.

Non-Transparent Operation

A virtual Ethernet connection (VEC) can be transparent or non-transparent with respect to Ethernet protocol data units (PDUs). The VEC non-transparency allows users to have a Frame Relay-type service between Layer 3 devices.

Circuit Multiplexing

Circuit multiplexing allows a node to participate in multiple services over a single Ethernet connection. By participating in multiple services, the Ethernet connection is attached to multiple logical networks. Some examples of possible service offerings are VPN services between sites, Internet services, and third-party connectivity for intercompany communications.

MAC-Address Learning, Forwarding, and Aging

Provider edge (PE) devices must learn remote MAC addresses and directly attached MAC addresses on ports that face the external network. MAC address learning accomplishes this by deriving the topology and forwarding information from packets originating at customer sites. A timer is associated with stored MAC addresses. After the timer expires, the entry is removed from the table.

Jumbo Frame Support

Jumbo frame support provides support for frame sizes between 1548 and 9216 bytes. You use the CLI to establish the jumbo frame size for any value specified in the above range. The default value is 1500 bytes in any Layer 2/VLAN interface. You can configure jumbo frame support on a per-interface basis.

Q-in-Q Support and Q-in-Q to EoMPLS VPLS Support

With 802.1Q tunneling (Q-in-Q), the customer edge (CE) device issues VLAN-tagged packets and VPLS forwards these packets to a far-end CE device. Q-in-Q refers to the fact that one or more 802.1Q tags may be located in a packet within the interior of the network. As packets are received from a CE device, an additional VLAN tag is added to incoming Ethernet packets to segregate traffic from different CE devices. Untagged packets originating from a CE device use a single tag within the interior of the VLAN switched network, whereas previously tagged packets originating from the CE device use two or more tags.

VPLS Services

Transparent LAN Service

Transparent LAN Service (TLS) is an extension to the point-to-point port-based Ethernet over Multiprotocol Label Switching (EoMPLS), which provides bridging protocol transparency (for example, bridge protocol data units [BPDUs]) and VLAN values. Bridges see this service as an Ethernet segment. With TLS, the PE device forwards all Ethernet packets received from the customer-facing interface (including tagged and untagged packets, and BPDUs) as follows:

- To a local Ethernet interface or an emulated virtual circuit (VC) if the destination MAC address is found in the Layer 2 forwarding table.
- To all other local Ethernet interfaces and emulated VCs belonging to the same VPLS domain if the destination MAC address is a multicast or broadcast address or if the destination MAC address is not found in the Layer 2 forwarding table.

Note You must enable Layer 2 protocol tunneling to run the Cisco Discovery Protocol (CDP), the VLAN Trunking Protocol (VTP), and the Spanning-Tree Protocol (STP).

Ethernet Virtual Connection Service

Ethernet Virtual Connection Service (EVCS) is an extension to the point-to-point VLAN-based Ethernet over MPLS (EoMPLS) that allows devices to reach multiple intranet and extranet locations from a single physical port. With EVCS, the provider edge (PE) device forwards all Ethernet packets with a particular VLAN tag received from the customer-facing interface (excluding bridge protocol data units [BPDUs]) as follows:

- To a local Ethernet interface or to an emulated virtual circuit (VC) if the destination MAC address is found in the Layer 2 forwarding table.
- To all other local Ethernet interfaces and emulated VCs belonging to the same Virtual Private LAN Services (VPLS) domain if the destination MAC address is a multicast or a broadcast address or if the destination MAC address is not found in the Layer 2 forwarding table.

Note Because it has only local significance, the demultiplexing VLAN tag that identifies a VPLS domain is removed before the packet is forwarded to the outgoing Ethernet interfaces or emulated VCs.

VPLS Statistics

VPLS statistic feature supports packet and byte count in ingress and egress directions. The following are the required criteria to enable this feature:

- Metro Aggregation services license
- Special SDM template

  Use the following commands to enable or disable VPLS statistics feature:

  ```
  sdm prefer vpls_stats_enable
  sdm prefer vpls_stats_disable
  ```

After template configuration, the node is auto reloaded.

Restrictions

- EFP statistics is not supported when VPLS statistics is enabled.
- Transit packet drops data is not supported.
- There is a sync time of 10 seconds between the software and the hardware for fetching the statistics.

- If access rewrite is configured (pop 1), VC statistics show 4 bytes less than the actual size (in both imposition and disposition node) because pop 1 removes the VLAN header.
- VC statistics do not account LDP and VC label. It displays what is received from access in both imposition and disposition node.

Example

The following example shows a sample VPLS Statics counter output:

```
router#show mpls l2transport vc 2200 detail

Local interface: Gi0/14/2 up, line protocol up, Ethernet:100 up
  Destination address: 10.163.123.218, VC ID: 2200, VC status: up
    Output interface: Te0/7/2, imposed label stack {24022 24025}
    Preferred path: not configured
    Default path: active
    Next hop: 10.163.122.74
  Create time: 20:31:49, last status change time: 16:27:32
    Last label FSM state change time: 16:27:44
  Signaling protocol: LDP, peer 10.163.123.218:0 up
    Targeted Hello: 10.163.123.215(LDP Id) -> 10.163.123.218, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: configured and enabled
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: No fault
      Last BFD dataplane     status rcvd: Not sent
      Last BFD peer monitor  status rcvd: No fault
      Last local AC  circuit status rcvd: No fault
      Last local AC  circuit status sent: No fault
      Last local PW i/f circ status rcvd: No fault
      Last local LDP TLV     status sent: No fault
     Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
    MPLS VC labels: local 110, remote 24025
    Group ID: local 40, remote 67109248
    MTU: local 9000, remote 9000
    Remote interface description: TenGigE0_0_2_3.2200
  Sequencing: receive disabled, send disabled
  Control Word: Off (configured: autosense)
  SSO Descriptor: 10.163.123.218/2200, local label: 110
  Dataplane:
    SSM segment/switch IDs: 16911/90633 (used), PWID: 71
  VC statistics:
    transit packet totals: receive 100, send 200
    transit byte totals:   receive 12800, send 25600
    transit packet drops:  receive 0, seq error 0, send 0
```

How to Configure Virtual Private LAN Services

Provisioning a Virtual Private LAN Services (VPLS) link involves provisioning the associated attachment circuit and a virtual forwarding instance (VFI) on a provider edge (PE) device.

In Cisco IOS XE Release 3.7S, the L2VPN Protocol-Based CLIs feature was introduced. This feature provides a set of processes and an improved infrastructure for developing and delivering Cisco IOS software on various Cisco platforms. This feature introduces new commands and modifies or replaces existing commands to achieve a consistent functionality across Cisco platforms and provide cross-Operating System (OS) support.

This section consists of tasks that use the commands existing prior to Cisco IOS XE Release 3.7S and a corresponding task that uses the commands introduced or modified by the L2VPN Protocol-Based CLIs feature.

Configuring PE Layer 2 Interfaces on CE Devices

You can configure the Ethernet flow point (EFP) as a Layer 2 virtual interface. You can also select tagged or untagged traffic from a customer edge (CE) device.

Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device

Note When Ethernet Virtual Connection Service (EVCS) is configured, a provider edge (PE) device forwards all Ethernet packets with a particular VLAN tag to a local Ethernet interface or emulated virtual circuit (VC) if the destination MAC address is found in the Layer 2 forwarding table.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no ip address** [*ip-address mask*] [**secondary**]
5. **negotiation auto**
6. **service instance** *si-id* **ethernet**
7. **encapsulation dot1q** *vlan-id*
8. **bridge-domain** *bd-id*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/1`	Specifies an interface and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies the service instance ID and enters service instance configuration mode.
Step 7	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance. Ensure that the interface on the adjoining customer edge (CE) device is on the same VLAN as this PE device.
Step 8	**bridge-domain** *bd-id* **Example:** `Device(config-if-srv)# bridge-domain 100`	Binds a service instance to a bridge domain instance.

Step 9	**end**
	Example:
	`Device(config-if-srv)# end`
		Exits service instance configuration mode and returns to privileged EXEC mode.

Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device: Alternate Configuration

Note When Ethernet Virtual Connection Service (EVCS) is configured, the PE device forwards all Ethernet packets with a particular VLAN tag to a local Ethernet interface or an emulated virtual circuit (VC) if the destination MAC address is found in the Layer 2 forwarding table.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no ip address** [*ip-address mask*] [**secondary**]
5. **negotiation auto**
6. **service instance** *si-id* **ethernet**

7. **encapsulation dot1q** *vlan-id*
8. **exit**
9. **exit**
10. **bridge-domain** *bd-id*
11. **member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*]
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/1`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 7	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance. • Ensure that the interface on the adjoining customer edge (CE) device is on the same VLAN as this provider edge (PE) device.

	Command or Action	Purpose
Step 8	**exit** **Example:** `Device(config-if-srv)# exit`	Exits service instance configuration mode and returns to interface configuration mode.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 10	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 11	**member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*] **Example:** `Device(config-bdomain)# member gigabitethernet0/0/1 service-instance 1000`	Binds a service instance to a bridge domain instance.

Step 12	**end**
	Example:
	`Device(config-bdomain)# end`
		Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring Access Ports for Untagged Traffic from a CE Device

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no ip address** [*ip-address mask*] [**secondary**]
5. **negotiation auto**
6. **service instance** *si-id* **ethernet**
7. **encapsulation untagged**
8. **bridge-domain** *bd-id*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/0`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 7	**encapsulation untagged** **Example:** `Device(config-if-srv)# encapsulation untagged`	Defines the matching criteria to map untagged ingress Ethernet frames on an interface to the appropriate service instance. • Ensure that the interface on the adjoining customer edge (CE) device is on the same VLAN as this provider edge (PE) device.
Step 8	**bridge-domain** *bd-id* **Example:** `Device(config-if-srv)# bridge-domain 100`	Binds a service instance or MAC tunnel to a bridge domain instance.
Step 9	**end** **Example:**	Exits service instance configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-if-srv)# end`	

Configuring Access Ports for Untagged Traffic from a CE Device: Alternate Configuration

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no ip address** [*ip-address mask*] [**secondary**]
5. **negotiation auto**
6. **service instance** *si-id* **ethernet**
7. **encapsulation untagged**
8. **exit**
9. **exit**
10. **bridge-domain** *bd-id*
11. **member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*]
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/4/4`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.

	Command or Action	Purpose
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 7	**encapsulation untagged** **Example:** `Device(config-if-srv)# encapsulation untagged`	Defines the matching criteria to map untagged ingress Ethernet frames on an interface to the appropriate service instance. • Ensure that the interface on the adjoining customer edge (CE) device is on the same VLAN as this provider edge (PE) device.
Step 8	**exit** **Example:** `Device(config-if-srv)# exit`	Exits service instance configuration mode and returns to interface configuration mode.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 10	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 11	**member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*] **Example:** `Device(config-bdomain)# member gigabitethernet0/4/4 service-instance 1000`	Binds a service instance to a bridge domain instance.

Step 12	**end**
	Example:
	`Device(config-bdomain)# end`
		Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring Q-in-Q EFP

Note When a thread-local storage (TLS) is configured, the provider edge (PE) device forwards all Ethernet packets received from the customer edge (CE) device to all local Ethernet interfaces and emulated virtual circuits (VCs) that belong to the same Virtual Private LAN Services (VPLS) domain if the MAC address is not found in the Layer 2 forwarding table.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no ip address** [*ip-address mask*] [**secondary**]
5. **negotiation auto**
6. **service instance** *si-id* **ethernet**
7. **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id*
8. **bridge-domain** *bd-id*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/2`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.

	Command or Action	Purpose
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 7	**encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200 second-dot1q 400`	Defines the matching criteria to map Q-in-Q ingress frames on an interface to the appropriate service instance. • Ensure that the interface on the adjoining CE device is on the same VLAN as this PE device.
Step 8	**bridge-domain** *bd-id* **Example:** `Device(config-if-srv)# bridge-domain 100`	Binds a service instance or a MAC tunnel to a bridge domain instance.

Step 9	**end**
	Example:
	`Device(config-if-srv)# end`
		Exits service instance configuration mode and returns to privileged EXEC mode.

Configuring Q-in-Q EFP: Alternate Configuration

Note When a thread-local storage (TLS) is configured, the provider edge (PE) device forwards all Ethernet packets received from the customer edge (CE) device to all local Ethernet interfaces and emulated virtual circuits (VCs) belonging to the same Virtual Private LAN Services (VPLS) domain if the MAC address is not found in the Layer 2 forwarding table.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no ip address** [*ip-address mask*] [**secondary**]
5. **negotiation auto**
6. **service instance** *si-id* **ethernet**
7. **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id*
8. **exit**
9. **exit**
10. **bridge-domain** *bd-id*
11. **member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*]
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. - Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/2`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 7	**encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200 second-dot1q 400`	Defines the matching criteria to map Q-in-Q ingress frames on an interface to the appropriate service instance. - Ensure that the interface on the adjoining CE device is on the same VLAN as this PE device.
Step 8	**exit** **Example:** `Device(config-if-srv)# exit`	Exits service instance configuration mode and returns to interface configuration mode.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 10	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 11	**member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*] **Example:** `Device(config-bdomain)# member gigabitethernet0/0/2 service-instance 1000`	Binds a service instance to a bridge domain instance.

Step 12	**end**
	Example:
	`Device(config-bdomain)# end`
		Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring MPLS on a PE Device

To configure Multiprotocol Label Switching (MPLS) on a provider edge (PE) device, configure the required MPLS parameters.

Note Before configuring MPLS, ensure that IP connectivity exists between all PE devices by configuring Interior Gateway Protocol (IGP), Open Shortest Path First (OSPF), or Intermediate System to Intermediate System (IS-IS) between PE devices.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **mpls label protocol** {**ldp** | **tdp**}
4. **mpls ldp logging neighbor-changes**
5. **mpls ldp discovery hello holdtime** *seconds*
6. **mpls ldp router-id** *interface-type-number* [**force**]
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 3	**mpls label protocol** {**ldp** | **tdp**}
	Example:
	`Device(config)# mpls label protocol ldp`
		Specifies the label distribution protocol for the platform.

Step 4	**mpls ldp logging neighbor-changes** **Example:** `Device(config)# mpls ldp logging neighbor-changes`	(Optional) Generates system error logging (syslog) messages when LDP sessions go down.
Step 5	**mpls ldp discovery hello holdtime** *seconds* **Example:** `Device(config)# mpls ldp discovery hello holdtime 5`	Configures the interval between the transmission of consecutive LDP discovery hello messages or the hold time for an LDP transport connection.
Step 6	**mpls ldp router-id** *interface-type-number* [**force**] **Example:** `Device(config)# mpls ldp router-id loopback0 force`	Specifies a preferred interface for the LDP router ID.

Step 7	**end**
	Example:
	`Device(config)# end`
		Exits global configuration mode and returns to privileged EXEC mode.

Configuring a VFI on a PE Device

The virtual forwarding interface (VFI) specifies the VPN ID of a Virtual Private LAN Services (VPLS) domain, the addresses of other provider edge (PE) devices in the domain, and the type of tunnel signaling and encapsulation mechanism for each peer.

Note Only Multiprotocol Label Switching (MPLS) encapsulation is supported.

Note You must configure BDI on the bridge domain that has the association with the VFI.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l2 vfi** *name* **manual**
4. **vpn id** *vpn-id*
5. **neighbor** *remote-router-id vc-id* {**encapsulation** *encapsulation-type* | **pw-class** *pw-name*} [**no-split-horizon**]
6. **bridge-domain** *bd-id*
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *name* **manual** **Example:** `Device(config)# l2 vfi vfi110 manual`	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks and enters VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 110`	Configures a VPN ID for a VPLS domain. • The emulated VCs bound to this Layer 2 virtual routing and forwarding (VRF) instance use this VPN ID for signaling.
Step 5	**neighbor** *remote-router-id vc-id* {**encapsulation** *encapsulation-type* | **pw-class** *pw-name*} [**no-split-horizon**] **Example:** `Device(config-vfi)# neighbor 172.16.10.2 4 encapsulation mpls`	Specifies the type of tunnel signaling and encapsulation mechanism for each VPLS peer. **Note** Split horizon is the default configuration to avoid broadcast packet looping and to isolate Layer 2 traffic. Use the **no-split-horizon** keyword to disable split horizon and to configure multiple VCs per spoke into the same VFI.
Step 6	**bridge-domain** *bd-id* **Example:** `Device(config-vfi)# bridge-domain 100`	Specifies a bridge domain.

	Command or Action	Purpose

Step 7	**end**
	Example:
	`Device(config-vfi)# end`
		Exits VFI configuration mode and returns to privileged EXEC mode.

Configuring a VFI on a PE Device: Alternate Configuration

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l2vpn vfi context** *name*
4. **vpn id** *id*
5. **member** *ip-address* [*vc-id*] **encapsulation mpls**
6. **exit**
7. **bridge-domain** *bd-id*
8. **member vfi** *vfi-name*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn vfi context** *name* **Example:** `Device(config)# l2vpn vfi context vfi110`	Establishes a L2VPN VFI between two or more separate networks, and enters VFI configuration mode.
Step 4	**vpn id** *id* **Example:** `Device(config-vfi)# vpn id 110`	Configures a VPN ID for a Virtual Private LAN Services (VPLS) domain. The emulated virtual circuits (VCs) bound to this Layer 2 virtual routing and forwarding (VRF) instance use this VPN ID for signaling.
Step 5	**member** *ip-address* [*vc-id*] **encapsulation mpls** **Example:**	Specifies the devices that form a point-to-point Layer 2 VPN (L2VPN) virtual forwarding interface (VFI)

	Command or Action	Purpose
	`Device(config-vfi)# member 172.16.10.2 4 encapsulation mpls`	connection and Multiprotocol Label Switching (MPLS) as the encapsulation type.
Step 6	**exit** **Example:** `Device(config-vfi)# exit`	Exits VFI configuration mode and returns to global configuration mode.
Step 7	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies a bridge domain and enters bridge-domain configuration mode.
Step 8	**member vfi** *vfi-name* **Example:** `Device(config-bdomain)# member vfi vfi110`	Binds a VFI instance to a bridge domain instance.

Step 9	**end**
	Example:
	`Device(config-bdomain)# end`
		Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring Static Virtual Private LAN Services

Note Static VPLS with TP tunnel is *not* supported on the Cisco ASR 900 RSP3 module.

To configure static Virtual Private LAN Services (VPLS), perform the following tasks:

- Configuring a Pseudowire for Static VPLS
- Configuring VFI for Static VPLS
- Configuring a VFI for Static VPLS: Alternate Configuration
- Configuring an Attachment Circuit for Static VPLS
- Configuring an Attachment Circuit for Static VPLS: Alternate Configuration
- Configuring an MPLS-TP Tunnel for Static VPLS with TP
- Configuring a VFI for Static VPLS: Alternate Configuration

Configuring a Pseudowire for Static VPLS

Note Pseudowire for Static VPLS is *not* supported on the Cisco ASR 900 RSP3 module.

The configuration of pseudowires between provider edge (PE) devices helps in the successful transmission of the Layer 2 frames between PE devices.

Use the pseudowire template to configure the virtual circuit (VC) type for the virtual path identifier (VPI) pseudowire. In the following task, the pseudowire will go through a Multiprotocol Label Switching (MPLS)-Tunneling Protocol (TP) tunnel.

The pseudowire template configuration specifies the characteristics of the tunneling mechanism that is used by the pseudowires, which are:

- Encapsulation type
- Control protocol
- Payload-specific options
- Preferred path

Perform this task to configure a pseudowire template for static Virtual Private LAN Services (VPLS).

Note Ensure that you perform this task before configuring the virtual forwarding instance (VFI) peer. If the VFI peer is configured before the pseudowire class, the configuration is incomplete until the pseudowire class is configured. The **show running-config** command displays an error stating that configuration is incomplete.

```
Device# show running-config | sec vfi

l2 vfi config manual
 vpn id 1000
 ! Incomplete point-to-multipoint vfi config
```

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **template type pseudowire** *name*
4. **encapsulation mpls**
5. **signaling protocol none**
6. **preferred-path interface Tunnel-tp** *interface-number*
7. **exit**
8. **interface pseudowire** *number*
9. **source template type pseudowire** *name*
10. **neighbor** *peer-address vcid-value*
11. **label** *local-pseudowire-label remote-pseudowire-label*
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**template type pseudowire** *name* **Example:** `Device(config)# template type pseudowire static-vpls`	Specifies the template type as pseudowire and enters template configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-template)# encapsulation mpls`	Specifies the tunneling encapsulation. • For Any Transport over MPLS (AToM), the encapsulation type is MPLS.
Step 5	**signaling protocol none** **Example:** `Device(config-template)# signaling protocol none`	Specifies that no signaling protocol is configured for the pseudowire class.
Step 6	**preferred-path interface Tunnel-tp** *interface-number* **Example:** `Device(config-template)# preferred-path interface Tunnel-tp 1`	(Optional) Specifies the path that traffic uses: an MPLS Traffic Engineering (TE) tunnel or destination IP address and Domain Name Server (DNS) name.
Step 7	**exit** **Example:** `Device(config-template)# exit`	Exits template configuration mode and returns to global configuration mode.
Step 8	**interface pseudowire** *number* **Example:** `Device(config)# interface pseudowire 1`	Establishes a pseudowire interface and enters interface configuration mode.
Step 9	**source template type pseudowire** *name* **Example:**	Configures the source template type of the configured pseudowire.

	Command or Action	Purpose
	`Device(config-if)# source template type pseudowire static-vpls`	
Step 10	**neighbor** *peer-address vcid-value* **Example:** `Device(config-if)# neighbor 10.0.0.1 123`	Specifies the peer IP address and VC ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 11	**label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-if)# label 301 17`	Configures an Any Transport over MPLS (AToM) static pseudowire connection by defining local and remote circuit labels.

Step 12	**end**
	Example:
	`Device(config-if)# end`
		Exits interface configuration mode and returns to privileged EXEC mode.

Configuring VFI for Static VPLS

Note Ensure that you perform this task after configuring the pseudowire. If the VFI peer is configured before the pseudowire, the configuration is incomplete until the pseudowire is configured. The output of the **show running-config** command displays an error stating that configuration is incomplete.

```
Device# show running-config | sec vfi

l2 vfi config manual
 vpn id 1000
 ! Incomplete point-to-multipoint vfi config
```

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **mpls label range** *minimum-value maximum-value* [**static** *minimum-static-value maximum-static-value*]
4. **pseudowire-class** [*pw-class-name*]
5. **encapsulation mpls**
6. **protocol** {**l2tpv2** | **l2tpv3** | **none**} *[l2tp-class-name]*
7. **exit**
8. **l2 vfi** *vfi-name* **manual**
9. **vpn id** *vpn-id*
10. **neighbor** *ip-address* **pw-class** *pw-name*
11. **mpls label** *local-pseudowire-label remote-pseudowire-label*
12. **mpls control-word**
13. **neighbor** *ip-address* **pw-class** *pw-name*

14. **mpls label** *local-pseudowire-label remote-pseudowire-label*
15. **mpls control-word**
16. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**mpls label range** *minimum-value maximum-value* [**static** *minimum-static-value maximum-static-value*] **Example:** `Device(config)# mpls label range 16 200 static 300 500`	Configures the range of local labels available for use with Multiprotocol Label Switching (MPLS) applications on packet interfaces.
Step 4	**pseudowire-class** [*pw-class-name*] **Example:** `Device(config)# pseudowire-class static_vpls`	Specifies the name of a Layer 2 pseudowire class and enters pseudowire class configuration mode.
Step 5	**encapsulation mpls** **Example:** `Device(config-pw-class)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 6	**protocol** {**l2tpv2** | **l2tpv3** | **none**} *[l2tp-class-name]* **Example:** `Device(config-pw-class)# protocol none`	Specifies that no signaling protocol will be used in Layer 2 Tunneling Protocol Version 3 (L2TPv3) sessions.
Step 7	**exit** **Example:** `Device(config-pw-class)# exit`	Exits pseudowire class configuration mode and returns to global configuration mode.
Step 8	**l2 vfi** *vfi-name* **manual** **Example:** `Device(config)# l2 vfi static-vfi manual`	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks, and enters Layer 2 VFI manual configuration mode.

	Command or Action	Purpose
Step 9	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 100`	Specifies the VPN ID.
Step 10	**neighbor** *ip-address* **pw-class** *pw-name* **Example:** `Device(config-vfi)# neighbor 10.3.4.4 pw-class static_vpls`	Specifies the IP address of the peer and the pseudowire class.
Step 11	**mpls label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-vfi)# mpls label 301 17`	Configures an Any Transport over MPLS (AToM) static pseudowire connection by defining local and remote circuit labels.
Step 12	**mpls control-word** **Example:** `Device(config-vfi)# mpls control-word`	(Optional) Enables the MPLS control word in an AToM static pseudowire connection.
Step 13	**neighbor** *ip-address* **pw-class** *pw-name* **Example:** `Device(config-vfi)# neighbor 2.3.4.3 pw-class static_vpls`	Specifies the IP address of the peer and the pseudowire class.
Step 14	**mpls label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-vfi)# mpls label 302 18`	Configures an AToM static pseudowire connection by defining local and remote circuit labels.
Step 15	**mpls control-word** **Example:** `Device(config-vfi)# mpls control-word`	(Optional) Enables the MPLS control word in an AToM static pseudowire connection.

Step 16	**end**
	Example:
	`Device(config-vfi)# end`
		Exits Layer 2 VFI manual configuration mode and returns to privileged EXEC mode.

Configuring a VFI for Static VPLS: Alternate Configuration

Note Ensure that you perform this task after configuring the pseudowire. If the VFI peer is configured before the pseudowire, the configuration is incomplete until the pseudowire is configured. The output of the **show running-config** command displays an error stating that configuration is incomplete.

```
Device# show running-config | sec vfi

l2 vfi config manual
 vpn id 1000
 ! Incomplete point-to-multipoint vfi config
```

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l2vpn vfi context** *vfi-name*
4. **vpn id** *vpn-id*
5. **exit**
6. **interface** *type number*
7. **encapsulation mpls**
8. **neighbor** *ip-address vc-id*
9. **label** *local-pseudowire-label remote-pseudowire-label*
10. **control-word** {**include** | **exclude**}
11. **exit**
12. **bridge-domain** *bd-id*
13. **member vfi** *vfi-name*
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks and enters VFI configuration mode.

	Command or Action	Purpose
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 100`	Specifies the VPN ID.
Step 5	**exit** **Example:** `Device(config-vfi)# exit`	Exits VFI configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface pseudowire 100`	Specifies an interface and enters interface configuration mode.
Step 7	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies an encapsulation type for tunneling Layer 2 traffic over a pseudowire.
Step 8	**neighbor** *ip-address vc-id* **Example:** `Device(config-if)# neighbor 10.3.4.4 100`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 9	**label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-if)# label 301 17`	Configures an Any Transport over MPLS (AToM) static pseudowire connection by defining local and remote circuit labels.

Step 10	**control-word** {**include** | **exclude**}
	Example:
	`Device(config-if)# control-word include`
		(Optional) Enables the Multiprotocol Label Switching (MPLS) control word in an AToM dynamic pseudowire connection.

Step 11	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 12	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 24`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 13	**member vfi** *vfi-name* **Example:**	Binds a service instance to a bridge domain instance.

	Command or Action	Purpose
	`Device(config-bdomain)# member vfi vpls1`	

Step 14	**end**
	Example:
	`Device(config-bdomain)# end`
		Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring an Attachment Circuit for Static VPLS

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet** *slot/interface*
4. **service instance** *si-id* **ethernet**
5. **encapsulation dot1q** *vlan-id*
6. **rewrite ingress tag pop** *number* [**symmetric**]
7. **bridge-domain** *bd-id*
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface gigabitethernet** *slot/interface* **Example:** `Device(config)# interface gigabitethernet 0/0/1`	Specifies an interface and enters interface configuration mode. • Ensure that the interfaces between the customer edge (CE) and provider edge (PE) devices that run Ethernet over MPLS (EoMPLS) are in the same subnet. All other interfaces and backbone devices do not need to be in the same subnet.
Step 4	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 100 ethernet`	Configures an Ethernet service instance on an interface and enters service instance configuration mode.

	Command or Action	Purpose
Step 5	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance. • Ensure that the interface on the adjoining CE device is on the same VLAN as this PE device.
Step 6	**rewrite ingress tag pop** *number* [**symmetric**] **Example:** `Device(config-if-srv)# rewrite ingress tag pop 1 symmetric`	(Optional) Specifies the encapsulation adjustment to be performed on a frame ingressing a service instance and the tag to be removed from a packet.
Step 7	**bridge-domain** *bd-id* **Example:** `Device(config-if-srv)# bridge-domain 24`	(Optional) Binds a service instance or a MAC tunnel to a bridge domain instance.

Step 8	**end**
	Example:
	`Device(config-if-srv)# end`
		Exits service instance configuration mode and returns to privileged EXEC mode.

Configuring an Attachment Circuit for Static VPLS: Alternate Configuration

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet** *slot/interface*
4. **service instance** *si-id* **ethernet**
5. **encapsulation dot1q** *vlan-id*
6. **rewrite ingress tag pop** *number* [**symmetric**]
7. **exit**
8. **exit**
9. **bridge-domain** *bd-id*
10. **member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*]
11. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface gigabitethernet** *slot/interface* **Example:** `Device(config)# interface gigabitethernet 0/0/1`	Specifies an interface and enters interface configuration mode. - Ensure that the interfaces between the customer edge (CE) and provider edge (PE) devices that are running Ethernet over MPLS (EoMPLS) are in the same subnet. All other interfaces and backbone devices do not need to be in the same subnet.
Step 4	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 5	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance. - Ensure that the interface on the adjoining CE device is on the same VLAN as this PE device.
Step 6	**rewrite ingress tag pop** *number* [**symmetric**] **Example:** `Device(config-if-srv)# rewrite ingress tag pop 1 symmetric`	(Optional) Specifies the encapsulation adjustment to be performed on a frame ingressing a service instance and the tag to be removed from a packet.
Step 7	**exit** **Example:** `Device(config-if-srv)# exit`	Exits service instance configuration mode and returns to interface configuration mode.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 9	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies the bridge domain ID and enters bridge-domain configuration mode.

	Command or Action	Purpose
Step 10	**member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*] **Example:** `Device(config-bdomain)# member gigabitethernet0/0/1 service-instance 1000`	(Optional) Binds a service instance to a bridge domain instance.

Step 11	**end**
	Example:
	`Device(config-bdomain)# end`
		Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring an MPLS-TP Tunnel for Static VPLS with TP

Note VPLS with TP/TE is not supported on Cisco ASR 900 RSP3 Module.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface Tunnel-tp** *number*
4. **no ip address**
5. **no keepalive**
6. **tp destination** *ip-address*
7. **bfd** *bfd-template*
8. **working-lsp**
9. **out-label** *number* **out-link** *number*
10. **lsp-number** *number*
11. **exit**
12. **protect-lsp**
13. **out-label** *number* **out-link** *number*
14. **in-label** *number*
15. **lsp-number** *number*
16. **exit**
17. **exit**
18. **interface** *type number*
19. **ip address** *ip-address ip-mask*
20. **mpls tp link** *link-num* {**ipv4** *ip-address* | **tx-mac** *mac-address*}
21. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. - Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface Tunnel-tp** *number* **Example:** `Device(config)# interface Tunnel-tp 4`	Configures a Multiprotocol Label Switching (MPLS) transport profile tunnel and enters interface configuration mode. - Use the same interface as you configured for the pseudowire class.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Disables the IP address configuration.
Step 5	**no keepalive** **Example:** `Device(config-if)# no keepalive`	Disables the keepalive configuration.
Step 6	**tp destination** *ip-address* **Example:** `Device(config-if)# tp destination 10.22.22.22`	Configures the tunnel destination.
Step 7	**bfd** *bfd-template* **Example:** `Device(config-if)# bfd tp`	Binds a single-hop Bidirectional Forwarding Detection (BFD) template to an interface.
Step 8	**working-lsp** **Example:** `Device(config-if)# working-lsp`	Configures the working label switched path (LSP) and enters working interface configuration mode.
Step 9	**out-label** *number* **out-link** *number* **Example:** `Device(config-if-working)# out-label 16 out-link 100`	Configures the out link and out label for the working LSP.

	Command or Action	Purpose
Step 10	**lsp-number** *number* **Example:** `Device(config-if-working)# lsp-number 0`	Configures the ID number for the working LSP.
Step 11	**exit** **Example:** `Device(config-if-working)# exit`	Exits working interface configuration mode and returns to interface configuration mode.
Step 12	**protect-lsp** **Example:** `Device(config-if)# protect-lsp`	Enters protection configuration mode for the label switched path (LSP) and enters protect interface configuration mode.
Step 13	**out-label** *number* **out-link** *number* **Example:** `Device(config-if-protect)# out-label 11 out-link 500`	Configures the out link and out label for the protect LSP.
Step 14	**in-label** *number* **Example:** `Device(config-if-protect)# in-label 600`	Configures the in label for the protect LSP.
Step 15	**lsp-number** *number* **Example:** `Device(config-if-protect)# lsp-number 1`	Configures the ID number for the working protect LSP.
Step 16	**exit** **Example:** `Device(config-if-protect)# exit`	Exits protect interface configuration mode and returns to interface configuration mode.
Step 17	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 18	**interface** *type number* **Example:** `Device(config-if)# interface GigabitEthernet 0/1/0`	Configures a interface and enters interface configuration mode.

	Command or Action	Purpose
Step 19	**ip address** *ip-address ip-mask* **Example:** `Device(config)# ip address 10.0.0.1 255.255.255.0`	(Optional) Configures the IP address and mask if not using an IP-less core.

Step 20	**mpls tp link** *link-num* {**ipv4** *ip-address* | **tx-mac** *mac-address*}
	Example:
	`Device(config-if)# mpls tp link 10 tx-mac 0100.0c99.8877`
		Configures Multiprotocol Label Switching (MPLS) transport profile (TP) link parameters.

Step 21	**end**
	Example:
	`Device(config-if)# end`
		Exits interface configuration mode and returns to privileged EXEC mode.

Configuring a VFI for Static VPLS: Alternate Configuration

Note Ensure that you perform this task after configuring the pseudowire. If the VFI peer is configured before the pseudowire, the configuration is incomplete until the pseudowire is configured. The output of the **show running-config** command displays an error stating that configuration is incomplete.

```
Device# show running-config | sec vfi

l2 vfi config manual
 vpn id 1000
 ! Incomplete point-to-multipoint vfi config
```

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l2vpn vfi context** *vfi-name*
4. **vpn id** *vpn-id*
5. **exit**
6. **interface** *type number*
7. **encapsulation mpls**
8. **neighbor** *ip-address vc-id*
9. **label** *local-pseudowire-label remote-pseudowire-label*
10. **control-word** {**include** | **exclude**}
11. **exit**
12. **bridge-domain** *bd-id*
13. **member vfi** *vfi-name*
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks and enters VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 100`	Specifies the VPN ID.
Step 5	**exit** **Example:** `Device(config-vfi)# exit`	Exits VFI configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface pseudowire 100`	Specifies an interface and enters interface configuration mode.
Step 7	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies an encapsulation type for tunneling Layer 2 traffic over a pseudowire.
Step 8	**neighbor** *ip-address vc-id* **Example:** `Device(config-if)# neighbor 10.3.4.4 100`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 9	**label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-if)# label 301 17`	Configures an Any Transport over MPLS (AToM) static pseudowire connection by defining local and remote circuit labels.

	Command or Action	Purpose

Step 10	**control-word** {**include** | **exclude**}
	Example:
	`Device(config-if)# control-word include`
		(Optional) Enables the Multiprotocol Label Switching (MPLS) control word in an AToM dynamic pseudowire connection.

Step 11	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 12	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 24`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 13	**member vfi** *vfi-name* **Example:** `Device(config-bdomain)# member vfi vpls1`	Binds a service instance to a bridge domain instance.

Step 14	**end**
	Example:
	`Device(config-bdomain)# end`
		Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuration Examples for Virtual Private LAN Services

Example: Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device

This example shows how to configure the tagged traffic:

```
Device(config)# interface GigabitEthernet 0/0/1
Device(config-if)# no ip address
Device(config-if)# negotiation auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation dot1q 200
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# end
```

Example: Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device: Alternate Configuration

The following example shows how to configure the tagged traffic:

```
Device(config)# interface GigabitEthernet 0/0/1
Device(config-if)# no ip address
Device(config-if)# negotiation auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation dot1q 200
Device(config-if-srv)# exit
Device(config-if)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member gigabitethernet0/0/1 service-instance 1000
Device(config-bdomain)# end
```

Example: Configuring Access Ports for Untagged Traffic from a CE Device

The following example shows how to configure access ports for untagged traffic:

```
Device(config)# interface gigabitethernet 0/0/0
Device(config-if)# no ip address
Device(config-if)# negotiation auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation untagged
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# end
```

The following example shows a virtual forwarding interface (VFI) configuration:

```
Device(config)# l2 vfi VPLSA manual
Device(config-vfi)# vpn id 110
Device(config-vfi)# neighbor 10.11.11.11 encapsulation mpls
Device(config-vfi)# neighbor 10.33.33.33 encapsulation mpls
Device(config-vfi)# neighbor 10.44.44.44 encapsulation mpls
Device(config-vfi)# bridge-domain 110
Device(config-vfi)# end
```

The following example shows a VFI configuration for hub and spoke.

```
Device(config)# l2 vfi VPLSB manual
Device(config-vfi)# vpn id 111
Device(config-vfi)# neighbor 10.99.99.99 encapsulation mpls
Device(config-vfi)# neighbor 10.12.12.12 encapsulation mpls
Device(config-vfi)# neighbor 10.13.13.13 encapsulation mpls no-split-horizon
Device(config-vfi)# bridge-domain 111
Device(config-vfi)# end
```

The output of the **show mpls 12transport vc** command displays various information related to a provide edge (PE) device. The VC ID in the output represents the VPN ID; the VC is identified by the combination of the destination address and the VC ID as shown in the command output. The output of the **show mpls l2transport vc detail** command displays detailed information about virtual circuits (VCs) on a PE device.

```
Device# show mpls l2transport vc 201

Local intf     Local circuit        Dest address    VC ID      Status
------------- -------------------- --------------- ---------- ----------
VFI VPLSA      VFI                  10.11.11.11     110        UP
VFI VPLSA      VFI                  10.33.33.33     110        UP
VFI VPLSA      VFI                  10.44.44.44     110        UP
```

The following sample output from the **show vfi** command displays the VFI status:

```
Device# show vfi VPLSA

VFI name: VPLSA, state: up
  Local attachment circuits:
    Vlan2
  Neighbors connected via pseudowires:
  Peer Address     VC ID     Split-horizon
  10.11.11.11          110             Y
  10.33.33.33          110             Y
  10.44.44.44          110             Y



Device# show vfi VPLSB

VFI name: VPLSB, state: up
  Local attachment circuits:
    Vlan2
  Neighbors connected via pseudowires:
  Peer Address     VC ID     Split-horizon
  10.99.99.99       111             Y
  10.12.12.12       111             Y
  10.13.13.13       111             N
```

Example: Configuring Access Ports for Untagged Traffic from a CE Device: Alternate Configuration

The following example shows how to configure the untagged traffic.

```
Device(config)# interface GigabitEthernet 0/4/4
Device(config-if)# no ip address
Device(config-if)# negotiation auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation untagged
Device(config-if-srv)# exit
Device(config-if)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member GigabitEthernet0/4/4 service-instance 10
Device(config-if-srv)# end
```

Example: Configuring Q-in-Q EFP

The following example shows how to configure the tagged traffic.

```
Device(config)# interface GigabitEthernet 0/0/2
Device(config-if)# no ip address
Device(config-if)# negotiate auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation dot1q 200 second-dot1q 400
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# end
```

Use the **show spanning-tree vlan** command to verify that the ports are not in a blocked state. Use the **show vlan id** command to verify that a specific port is configured to send and receive specific VLAN traffic.

Example: Configuring Q-in-Q in EFP: Alternate Configuration

The following example shows how to configure the tagged traffic:

```
Device(config)# interface GigabitEthernet 0/4/4
Device(config-if)# no ip address
Device(config-if)# nonegotiate auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation dot1q 200 second-dot1q 400
Device(config-if-srv)# exit
Device(config-if)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member GigabitEthernet0/4/4 service-instance 1000
Device(config-bdomain)# end
```

Use the **show spanning-tree vlan** command to verify that the port is not in a blocked state. Use the **show vlan id** command to verify that a specific port is configured to send and receive a specific VLAN traffic.

Example: Configuring MPLS on a PE Device

The following example shows a global Multiprotocol Label Switching (MPLS) configuration:

```
Device(config)# mpls label protocol ldp
Device(config)# mpls ldp logging neighbor-changes
Device(config)# mpls ldp discovery hello holdtime 5
Device(config)# mpls ldp router-id Loopback0 force
```

The following sample output from the **show ip cef** command displays the Label Distribution Protocol (LDP) label assigned:

```
Device# show ip cef 192.168.17.7

192.168.17.7/32, version 272, epoch 0, cached adjacency to POS4/1
0 packets, 0 bytes
  tag information set
    local tag: 8149
    fast tag rewrite with PO4/1, point2point, tags imposed: {4017}
  via 10.3.1.4, POS4/1, 283 dependencies
    next hop 10.3.1.4, POS4/1
    valid cached adjacency
    tag rewrite with PO4/1, point2point, tags imposed: {4017}
```

Example: VFI on a PE Device

The following example shows a virtual forwarding instance (VFI) configuration:

```
Device(config)# l2 vfi vfi110 manual
Device(config-vfi)# vpn id 110
Device(config-vfi)# neighbor 172.16.10.2 4 encapsulation mpls
Device(config-vfi)# neighbor 10.16.33.33 encapsulation mpls
Device(config-vfi)# neighbor 198.51.100.44 encapsulation mpls
Device(config-vfi)# bridge-domain 100
Device(config-vfi)# end
```

The following example shows a VFI configuration for a hub-and-spoke configuration:

```
Device(config)# l2 vfi VPLSA manual
Device(config-vfi)# vpn id 110
Device(config-vfi)# neighbor 10.9.9.9 encapsulation mpls
Device(config-vfi)# neighbor 192.0.2.12 encapsulation mpls
Device(config-vfi)# neighbor 203.0.113.4 encapsulation mpls no-split-horizon
Device(config-vfi)# bridge-domain 100
Device(config-vfi)# end
```

The **show mpls 12transport vc** command displays information about the provider edge (PE) device. The **show mpls l2transport vc detail** command displays detailed information about the virtual circuits (VCs) on a PE device.

```
Device# show mpls l2transport vc 201

Local intf     Local circuit        Dest address    VC ID      Status
-------------  -------------------- --------------- ---------- ----------
VFI test1      VFI                  209.165.201.1   201        UP
VFI test1      VFI                  209.165.201.2   201        UP
VFI test1      VFI                  209.165.201.3   201        UP
```

The **show vfi** *vfi-name* command displays VFI status. The VC ID in the output represents the VPN ID; the VC is identified by the combination of the destination address and the VC ID as in the example below.

```
Device# show vfi VPLS-2

VFI name: VPLS-2, state: up
  Local attachment circuits:
    Vlan2
  Neighbors connected via pseudowires:
  Peer Address     VC ID     Split-horizon
  10.1.1.1          2             Y
  10.1.1.2          2             Y
  10.2.2.3          2             N
```

Example: VFI on a PE Device: Alternate Configuration

The following example shows how to configure a virtual forwarding interface (VFI) on a provider edge (PE) device:

```
Device(config)# l2vpn vfi context vfi110
Device(config-vfi)# vpn id 110
Device(config-vfi)# member 172.16.10.2 4 encapsulation mpls
Device(config-vfi)# member 10.33.33.33 encapsulation mpls
Device(config-vfi)# member 10.44.44.44 encapsulation mpls
Device(config-vfi)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member vfi vfi110
Device(config-bdomain)# end
```

The following example shows how to configure a hub-and-spoke VFI configuration:.

```
Device(config)# l2vpn vfi context VPLSA
Device(config-vfi)# vpn id 110
Device(config-vfi)#  member 10.9.9.9 encapsulation mpls
Device(config-vfi)#  member 172.16.10.2 4 encapsulation mpls
Device(config-vfi)#  exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member vfi VPLSA
Device(config-bdomain)# member GigabitEthernet0/0/0 service-instance 100
Device(config-bdomain)# member 10.33.33.33 10 encapsulation mpls
Device(config-bdomain)# end
```

The **show l2vpn atom vc** command displays information about the PE device. The command also displays information about Any Transport over MPLS (AToM) virtual circuits (VCs) and static pseudowires that are enabled to route Layer 2 packets on a device.

```
Device# show l2vpn atom vc

Local intf    Local circuit           Dest address    VC ID      Status
------------- ----------------------- --------------- ---------- ----------
Et0/0.1       Eth VLAN 101            10.0.0.2        101        UP
Et0/0.1       Eth VLAN 101            10.0.0.3        201        DOWN
```

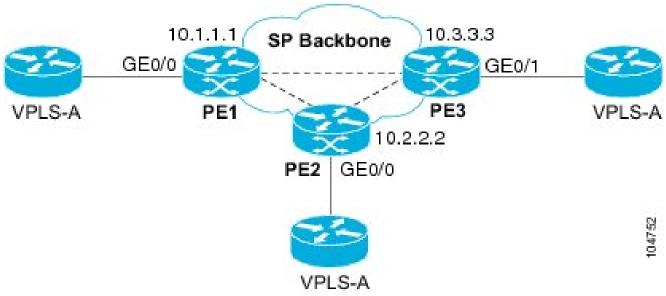
The **show l2vpn vfi** command displays the VFI status. The VC ID in the output represents the VPN ID; the VC is identified by the combination of the destination address and the VC ID as in the example below.

```
Device# show l2vpn vfi VPLS-2

Legend: RT= Route-target

VFI name: serviceCore1, State: UP, Signaling Protocol: LDP
  VPN ID: 100, VPLS-ID: 9:10, Bridge-domain vlan: 100
  RD: 9:10, RT: 10.10.10.10:150
  Pseudo-port Interface: Virtual-Ethernet1000

  Neighbors connected via pseudowires:
  Interface    Peer Address    VC ID      Discovered Router ID   Next Hop
  Pw2000       10.0.0.1        10         10.0.0.1               10.0.0.1
  Pw2001       10.0.0.2        10         10.1.1.2               10.0.0.2
  Pw2002       10.0.0.3        10         10.1.1.3               10.0.0.3
  Pw5          10.0.0.4        10         -                      10.0.0.4
```

Example: Full-Mesh VPLS Configuration

In a full-mesh configuration, each provider edge (PE) device creates a multipoint-to-multipoint forwarding relationship with all other PE devices in the Virtual Private LAN Services (VPLS) domain using a virtual forwarding interface (VFI). An Ethernet or a VLAN packet received from the customer network can be forwarded to one or more local interfaces and/or emulated virtual circuits (VCs) in the VPLS domain. To avoid a broadcast packet loop in the network, packets received from an emulated VC cannot be forwarded to any emulated VC in the VPLS domain on a PE device. Ensure that Layer 2 split horizon is enabled to avoid a broadcast packet loop in a full-mesh network.

Figure 2: Full-Mesh VPLS Configuration



PE 1 Configuration

The following examples shows how to create virtual switch instances (VSIs) and associated VCs:

```
l2 vfi PE1-VPLS-A manual
 vpn id 100
 neighbor 10.2.2.2 encapsulation mpls
 neighbor 10.3.3.3 encapsulation mpls
 bridge domain 100
!
interface Loopback 0
 ip address 10.1.1.1 255.255.0.0
```

The following example shows how to configure the customer edge (CE) device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface GigabitEthernet 0/0/0
 no ip address
 negotiation auto
 service instance 10 ethernet
 encapsulation dot1q 200
 bridge-domain 100
```

PE 2 Configuration

The following example shows how to create VSIs and associated VCs.

```
l2 vfi PE2-VPLS-A manual
 vpn id 100
 neighbor 10.1.1.1 encapsulation mpls
 neighbor 10.3.3.3 encapsulation mpls
 bridge domain 100
!
interface Loopback 0
 ip address 10.2.2.2 255.255.0.0
```

The following example shows how to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface GigabitEthernet 0/0/0
 no ip address
 negotiation auto
 service instance 10 ethernet
 encapsulation dot1q 200
 bridge-domain 100
```

PE 3 Configuration

The following example shows how to create VSIs and associated VCs:

```
l2 vfi PE3-VPLS-A manual
 vpn id 112
 neighbor 10.1.1.1 encapsulation mpls
 neighbor 10.2.2.2 encapsulation mpls
 bridge domain 100
!
interface Loopback 0
 ip address 10.3.3.3 255.255.0.0
```

The following example shows how to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN).

```
interface GigabitEthernet 0/0/1
 no ip address
 negotiation auto
 service instance 10 ethernet
 encapsulation dot1q 200
 bridge-domain 100
!
```

The following sample output from the **show mpls l2 vc** command provides information about the status of the VC:

```
Device# show mpls l2 vc

Local intf       Local circuit        Dest address    VC ID      Status
-------------    -------------------- --------------- ---------- ----------
VFI PE1-VPLS-A   VFI                  10.2.2.2         100        UP
VFI PE1-VPLS-A   VFI                  10.3.3.3         100        UP
```

The following sample output from the **show vfi** command provides information about the VFI:

```
Device# show vfi PE1-VPLS-A

VFI name: VPLSA, state: up
  Local attachment circuits:
    Vlan200
  Neighbors connected via pseudowires:
    10.2.2.2  10.3.3.3
```

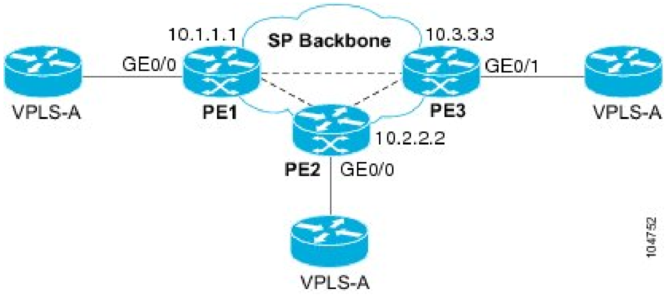
The following sample output from the **show mpls 12transport vc** command provides information about virtual circuits:

```
Device# show mpls l2transport vc detail

Local interface: VFI PE1-VPLS-A up
  Destination address: 10.2.2.2, VC ID: 100, VC status: up
    Tunnel label: imp-null, next hop point2point
    Output interface: Se2/0, imposed label stack {18}
  Create time: 3d15h, last status change time: 1d03h
  Signaling protocol: LDP, peer 10.2.2.2:0 up
    MPLS VC labels: local 18, remote 18
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 0, send 0
    byte totals:   receive 0, send 0
    packet drops:  receive 0, send 0
```

Example: Full-Mesh Configuration : Alternate Configuration

In a full-mesh configuration, each provider edge (PE) router creates a multipoint-to-multipoint forwarding relationship with all other PE routers in the Virtual Private LAN Services (VPLS) domain using a virtual forwarding interface (VFI). An Ethernet or virtual LAN (VLAN) packet received from the customer network can be forwarded to one or more local interfaces and/or emulated virtual circuits (VCs) in the VPLS domain. To avoid broadcasted packets looping in the network, no packet received from an emulated VC can be forwarded to any emulated VC of the VPLS domain on a PE router. That is, Layer 2 split horizon should always be enabled as the default in a full-mesh network.

Figure 3: VPLS Configuration Example



PE 1 Configuration

The following example shows how to create virtual switch instances (VSIs) and associated VCs and to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface gigabitethernet 0/0/0
 service instance 100 ethernet
 encap dot1q 100
 no shutdown
!
l2vpn vfi context PE1-VPLS-A
 vpn id 100
 neighbor 10.2.2.2 encapsulation mpls
 neighbor 10.3.3.3 encapsulation mpls
!
bridge-domain 100
 member gigabitethernet0/0/0 service-instance 100
 member vfi PE1-VPLS-A
```

PE 2 Configuration

The following example shows how to create VSIs and associated VCs and to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface gigabitethernet 0/0/0
 service instance 100 ethernet
 encap dot1q 100
 no shutdown
!
l2vpn vfi context PE2-VPLS-A
 vpn id 100
 neighbor 10.1.1.1 encapsulation mpls
 neighbor 10.3.3.3 encapsulation mpls
!
bridge-domain 100
 member gigabitethernet0/0/0 service-instance 100
 member vfi PE2-VPLS-A
```

PE 3 Configuration

The following example shows how to create of the VSIs and associated VCs and to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface gigabitethernet 0/0/0
 service instance 100 ethernet
 encap dot1q 100
 no shutdown
!
l2vpn vfi context PE3-VPLS-A
 vpn id 100
 neighbor 10.1.1.1 encapsulation mpls
 neighbor 10.2.2.2 encapsulation mpls
!
bridge-domain 100
 member gigabitethernet0/0/0 service-instance 100
 member vfi PE3-VPLS-A
```

The following sample output from the **show mpls l2 vc** command provides information on the status of the VC:

```
Device# show mpls l2 vc

Local intf       Local circuit  Dest address    VC ID      Status
-------------    -------------- --------------- ---------- ----------
VFI PE3-VPLS-A   VFI            10.2.2.2        100        UP
VFI PE3-VPLS-A   VFI            10.3.3.3        100        UP
```

The following sample output from the **show l2vpn vfi** command provides information about the VFI:

```
Device# show l2vpn vfi VPLS-2

Legend: RT= Route-target

VFI name: serviceCore1, State: UP, Signaling Protocol: LDP
  VPN ID: 100, VPLS-ID: 9:10, Bridge-domain vlan: 100
  RD: 9:10, RT: 10.10.10.10:150
  Pseudo-port Interface: Virtual-Ethernet1000

  Neighbors connected via pseudowires:
  Interface     Peer Address    VC ID      Discovered Router ID   Next Hop
  Pw2000        10.0.0.1        10         10.0.0.1               10.0.0.1
  Pw2001        10.0.0.2        10         10.1.1.2               10.0.0.2
  Pw2002        10.0.0.3        10         10.1.1.3               10.0.0.3
  Pw5           10.0.0.4        10         -                      10.0.0.4
```

The following sample output from the **show l2vpn atom vc** command provides information on the virtual circuits:

```
Device# show l2vpn atom vc

Local intf    Local circuit           Dest address    VC ID      Status
------------- ----------------------- --------------- ---------- ----------
Et0/0.1       Eth VLAN 101            10.0.0.2        101        UP
Et0/0.1       Eth VLAN 101            10.0.0.3        201        DOWN
```

Flow Aware Transport (FAT) Pseudowire (PW) over VPLS

A Pseudowire (PW) load-balances traffic, between the ingress and egress PE routers, by using the Equal Cost Multiple Path (ECMP) routing technique to route packets along multiple PWs of equal cost, based on the VC label. Using multiple PWs results in wasted resources because the technique does not load balance within a PW. The distribution of multiple flows within a PW over ECMPs is a new functionality provided by FAT-PW over VPLS.

FAT-PW over VPLS uses a flow label, which is a unique identifier to distinguish a flow within the PW, and is derived from the payload of a packet. The flow label contains the End of Label Stack (EOS) bit set and inserted after the VC label and before the control word (if any). Calculation and pushing of the flow label are done by an ingress PE, which is enabled with the FAT PW configuration. The egress PE discards the flow label. For more information, see Flow-Aware Transport (FAT) Load Balancing.

Note The FAT-PW over VPLS is supported on Cisco IOS XE 16.9.1 and later. It is supported on the Cisco RSP3 module.

You can use the following commands to configure the FAT-PW over VPLS feature:

- **load-balance flow-label both**—Between two PE routers (that is, its head starts at the imposition PE router and its tail terminates on the disposition PE router), PE1 and PE2, when FAT-PW is enabled, L2 traffic is load balanced in both transmit and receive directions.
- **load-balance flow-label receive**—Between two PE routers (that is, its head starts at the imposition PE router and its tail terminates on the disposition PE router), PE1 and PE2, when FAT-PW is enabled, L2 traffic is load balanced only in the receive direction.
- **load-balance flow-label transmit**—Between two PE routers (that is, its head starts at the imposition PE router and its tail terminates on the disposition PE router), PE1 and PE2, when FAT-PW is enabled, L2 traffic is load balanced only in the transmit direction.

Configuring FAT-PW over VPLS

Step 1 Configure the Bridge Domain.

Example:

```
(config)# bridge-domain 100
(config-bdomain)# member GigabitEthernet0/0/1 service-instance 123
(config-bdomain)# member vfi 100
(config-bdomain)# exit
(config)# exit
```

Step 2 Configure the service instance.

Example:

```
(config)# service instance 100 ethernet
(config)# encapsulation dot1q 100
(config)# rewrite ingress tag pop 1 symmetric
(config-if)# exit
```

Step 3 Configure the L2VPN.

Example:

```
(config)# interface pseudowire20
(config-if)# encapsulation mpls
(config-if)# neighbor 20.20.20.20 123
```

Step 4 Configure the steps to enable FAT-PW over VPLS.

Example:

```
(config-if)# load-balance flow-label ?
    both      Enable FATPW in both directions
    receive   Enable FATPW in the receive direction
    transmit  Enable FATPW in the transmit direction
(config-if)# exit
(config)# l2vpn vfi context 100
(config-cross-connect)# vpn id 100
(config-cross-connect)# member pseudowire20
(config-cross-connect)# exit
```

Restrictions for FAT-PW over VPLS

- By default, the load balance selects the **src-dst-mac-ip four-tuple** (Source MAC Address, Destination MAC Address, Source IP, and Destination IP) hash method to generate a unique flow label in the VPLS implementation.
- The RSP3 module cannot control the selection of the tuple hash method using the **load-balance flow ethernet** *both* command.
- FAT-PW cannot be enabled if VPLS is in autodiscovery mode because the load balance CLI is available only on the pseudowire interface, and it cannot be configured with autodiscovery.
- FAT-PW is not supported with the **l2 vfi** *name* manual model.

- If one of the nodes runs the 16.7.1 image that supports FAT-PW over EoMPLS, the FAT-PW negotiation is enabled for VPLS, however, there will be a considerable traffic drop. Therefore, it is recommended to run the 16.9.1 image or later for FAT-PW over VPLS.
- FAT-PW over VPLS is not supported with BGP signaling.
- Due to the existing design limitation, load balancing based on a flow-label does not work when RSP3 is deployed at the P node where rLFA/LFA configurations are present.
- Routed FAT-PW is not supported.
- Load balancing is not supported with DHCP packets.
- There is no change in the existing 4k scale number with respect to VPLS.
- The FAT-PW feature configuration is available only under the new configuration model. Therefore, all restrictions that are applicable for the new configuration model are also applicable for this feature.

Verifying FAT-PW over VPLS

Use the **show platform hardware pp active pw vpls** command to verify if the FAT flow label has been signaled and the direction of the load balancing—imposition or disposition.

```
pw    : VFI1            bdomain         : 50          vsi       : 0x15
peer_ip        : 4.4.4.4         vc_id           : 1           has_cw    : 0
STP            : FWD             status          : Disabled    sh_group  : 0
local_label    : 18              remote_label    : 19          sh_type   : Hub
imp_oce        : 0x23DBECE4      disp_oce        : 0x23DBEDC4  label_oce : 0x23DBF19C
pwe_lif        : 0x8000          psn_fec         : 0x20000410  encap_id  : 0x8000
dest_gport     : 0x6C0000D1      ing_gport       : 0x18908000  egr_gport : 0x18A08000
imp_flow_label : Yes             disp_flow_label : Yes
```

Layer 2 Protocol Tunneling

Customers at different sites connected across a service-provider network need to use various Layer 2 protocols to scale their topologies to include all remote sites, as well as the local sites. STP must run properly, and every VLAN should build a proper spanning tree that includes the local site and all remote sites across the service-provider network. Cisco Discovery Protocol (CDP) must discover neighboring Cisco devices from local and remote sites.

VLAN Trunking Protocol (VTP) must provide consistent VLAN configuration throughout all sites in the customer network that are participating in VTP. Similarly, DTP, LACP, LLDP, PAgP, and UDLD can also run across the service-provider network.

When protocol tunneling is enabled, edge switches on the inbound side of the service-provider network encapsulate Layer 2 protocol packets with a special MAC address (0100.0CCD.CDD0) and send them across the service-provider network. Core switches in the network do not process these packets but forward them as normal (unknown multicast data) packets. Layer 2 protocol data units (PDUs) for the configured protocols cross the service-provider network and are delivered to customer switches on the outbound side of the service-provider network. Identical packets are received by all customer ports on the same VLANs with these results:

- Users on each of a customer's sites can properly run STP, and every VLAN can build a correct spanning tree based on parameters from all sites and not just from the local site.

- CDP discovers and shows information about the other Cisco devices connected through the service-provider network.
- VTP provides consistent VLAN configuration throughout the customer network, propagating to all switches through the service provider that support VTP.

Customers use Layer 2 protocol tunneling to tunnel BPDUs through a service-provider network without interfering with internal provider network BPDUs.

Note Layer 2 protocol tunneling is supported on EFPs, but not on switchports. Layer 2 protocol tunneling is not supported on cross-connect EFPs.

In figure below, Customer X has four switches in the same VLAN, which are connected through the service-provider network. If the network does not tunnel PDUs, switches on the far ends of the network cannot properly run STP, CDP, and other Layer 2 protocols. For example, STP for a VLAN on a switch in Customer X, Site 1, will build a spanning tree on the switches at that site without considering convergence parameters based on Customer X's switch in Site 2. This could result in the topology shown in figure below.

Figure 4: Layer 2 Protocol Tunneling

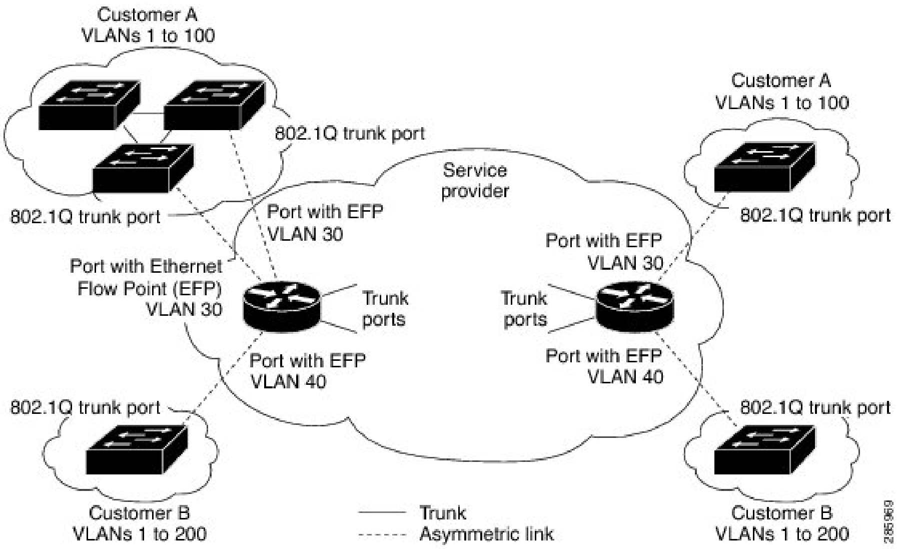
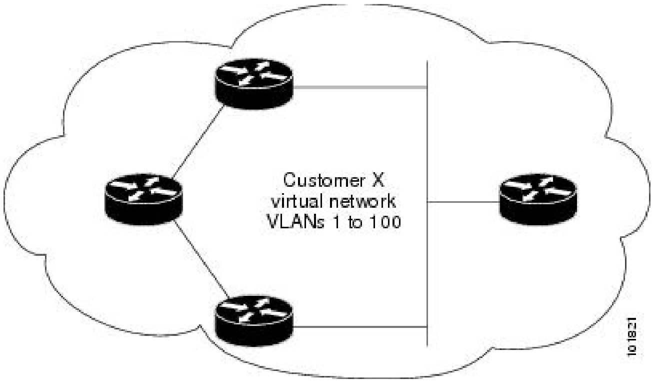


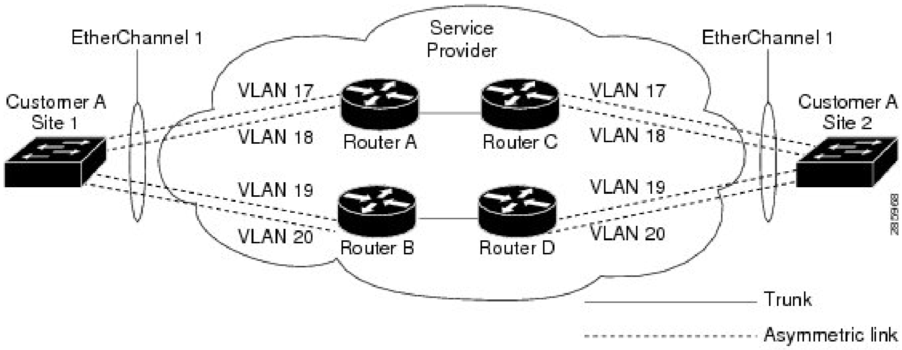
Figure 5: Layer 2 Network Topology without Proper Convergence



In a service-provider network, you can use Layer 2 protocol tunneling to enhance the creation of EtherChannels by emulating a point-to-point network topology. When you enable protocol tunneling (PAgP or LACP) on the service-provider switch, remote customer switches receive the PDUs and can negotiate the automatic creation of EtherChannels.

For example, in figure below, Customer A has two switches in the same VLAN that are connected through the SP network. When the network tunnels PDUs, switches on the far ends of the network can negotiate the automatic creation of EtherChannels without needing dedicated lines.

Figure 6: Layer 2 Protocol Tunneling for EtherChannels



Use the **l2protocol tunnel** *protocol* service-instance configuration command to enable Layer 2 protocol tunneling on a service instance:

Valid protocols include CDP, LACP, LLDP, PAgP, STP, UDLD, and VTP. If a protocol is not specified for a service instance, the protocol frame is dropped at the interface.

This is an example of Layer 2 protocol tunneling configuration:

```
Router (config)# interface gigabitethernet0/0/2
Router (config-if)# service instance 10 Ethernet
Router (config-if-srv)# encapsulation untagged, dot1q 200 second-dot1q 300
Router (config-if-srv)# l2protocol tunnel cdp stp vtp dtp pagp lacp
Router (config-if-srv)# bridge-domain 10
```

Note To enable tunneling of most Layer 2 protocol, you must configure **encapsulation untagged** because Layer 2 protocol PDUs are usually untagged.

Layer 2 protocol tunneling statistics

The following command is used to view the Layer 2 protocol tunneling statistics:

show ethernet service instance id *service-instance id*interface *interface* platform.

This is an example of Layer 2 protocol tunneling statistics:

```
2020#sh run int gi0/0/9
Building configuration...

Current configuration : 228 bytes
interface GigabitEthernet0/0/9
 no ip address
 media-type auto-select
 negotiation auto
 no keepalive
 service instance 200 ethernet
  encapsulation untagged
  l2protocol tunnel
  xconnect 2.2.2.2 1 encapsulation mpls
end


2020#show ethernet service instance id 200 inter gig 0/0/9 platform

Service Instance (EFP) L2 PDU Handing Info

EFP               CDP   STP   VTP   DTP   PAGP  LLDP  LACP  UDLD  LOAM  ESMC  ELMI  PTPPD
 RES4  RES5  RES6  RES8  RES9  RESA  RESB  RESC  RESD  RESF  CFG  NH
-----------------------------------------------------------------------------------------------------------------------------------------------
Gi0/0/9.Efp200    TUNL  TUNL  TUNL  DROP  TUNL  TUNL  TUNL  TUNL  TUNL  TUNL  TUNL  TUNL
 TUNL  TUNL  TUNL  TUNL  TUNL  TUNL  TUNL  TUNL  TUNL  TUNL  Y    N

EFP L2PT Tunnel statistics
----------------------------------------
L2protocol        Encapped      Decapped
----------------------------------------
CDP:              0             0
STP:              4059          13661
VTP:              0             0
DTP:              0             0
PAGP:             0             0
LLDP:             0             0
LACP:             0             0
UDLD:             0             0
LOAM:             0             0
ESMC:             0             0
ELMI:             0             0
PTPPD:            0             0
```

Note Layer 2 Protocol Tunnel decap statistics increments on core port for Layer 2 Protocol Tunnel over BD/VPLS scenario and Layer 2 Protocol Tunnel.