Configuring Ethernet Virtual Circuit

This chapter describes Ethernet Virtual Circuit (EVC), EVC types, Ethernet Flow Point (EFP), and bridge domain. This chapter also describes procedures to configure EVC.

This chapter includes the following topics:

- Understanding Carrier Ethernet, page 2
- Understanding Ethernet Virtual Circuit, page 2
- Understanding Ethernet Flow Point, page 3
- Understanding Bridge Domain, page 3
- EVC Features, page 4
- EVC Types, page 5
- Counters, page 8
- EVC and EFP Limitations and Restrictions in CPT, page 9
- Configuration Procedures, page 9
- NTP-J1 Configure an EVC Circuit, page 10
- DLP-J216 Configure a Bridge Domain Using Cisco IOS Commands, page 10
- DLP-J1 Configure an Ethernet Service Instance Using Cisco IOS Commands, page 12
- DLP-J2 Create an EVC Circuit Using CTC, page 13
- DLP-J3 Edit an EVC Circuit Using CTC, page 17
- DLP-J4 Query an EVC Circuit Using CTC, page 20
- Interactions of EVC with Other Features, page 21
- DLP-J212 Configure Layer 2 Protocol Tunneling Using Cisco IOS Commands, page 24
- DLP-J213 Configure Layer 2 Protocol Tunneling Using CTC, page 25
- Supported Encapsulation and Rewrite Operations, page 25

Understanding Carrier Ethernet

The Carrier Ethernet uses a high bandwidth Ethernet technology to deliver dedicated connectivity. It provides network connectivity by connecting to the customer site through a private Layer 2 Ethernet circuit. The available interfaces are normally 10Mbps and 100Mbps Fast Ethernet and 1000Mb/s Gigabit Ethernet.

The Carrier Ethernet enables you to run different services over a single connection. Next Generation Networks, VoIP, Storage, and Managed Security are some of the services that can run over a single Carrier Ethernet connection.

The Metro Ethernet Forum (MEF) defines the following five attributes to define an Ethernet as Carrier class:

- Standardized Services
- Quality of Service
- Scalability
- Service Management
- Reliability

Carrier Ethernet can be deployed in many ways:

- Ethernet over SDH/SONET
- Ethernet over MPLS
- Native Ethernet

Note Carrier Packet Transport (CPT) system does not support Ethernet over SDH/SONET.

Understanding Ethernet Virtual Circuit

The Ethernet Virtual Circuit (EVC) represents a logical relationship between Ethernet User–Network interfaces (UNI) in a provider–based Ethernet service. The EVC represents the service offered and is carried through the provider network. Each EVC is configured by its unique name across the provider network.

An EVC is an end–to–end representation of a single instance of a Layer 2 service that a service provider offers. It embodies the different parameters based on which the service is offered. EVC prevents data transfer between sites that are not part of the same EVC.

In simple terms, EVC is the A–Z circuit that enables you to pass customer VLANs from one port on a node to another port on another node in the network.

In the CPT system, the EVC represents a Carrier Ethernet Service and is an entity that provides end–to–end connection between two or more customer end points.

EVC Attributes

Some of the global EVC attributes are:

- EVC ID—An unique identifier that identifies the EVC

- EVC Type—E-LINE, E-LAN, or E-TREE
- List of associated EFPs that belong to an EVC

Understanding Ethernet Flow Point

The traffic for the service needs to pass through several switches in the provider network to connect customer sites across the provider network. The instance of a specific EVC service on the physical interface of each network device through which the EVC passes through is called an Ethernet Flow Point (EFP). An EFP is a logical demarcation point of an EVC on an interface. An EFP can be associated with a bridge domain.

In simple terms, an EFP is defined as an end point of an EVC within a node. Because multiple EVCs can pass through one physical interface, the main purpose of an EFP configuration is to recognize the traffic belonging to a specific EVC on that interface and to apply the forwarding behavior and features specific to that EVC.

The EFPs on CPT can be on all the ports of the fabric card, line card, or the CPT 50 panel.

The possible EFP administrative states are UP and DOWN. This administrative state maps to the EFP administrative state in IOS.

EFP Attributes

The key attributes of an EFP are:

- Encapsulation string—Defines the classification criteria for an incoming packet.
- Forwarding operations—Defines the forwarding operation to be applied on frames that belong to this EFP.
- Ingress rewrite operation—Defines the rewrites to be performed on the frames that belong to this EFP before proceeding with the forwarding operations.

Note CPT supports only the Ingress rewrite operation and all the rewrite operations are symmetric in nature.

For a list of supported encapsulation and ingress rewrite operations on point–to–point (P2P) EVC and point–to–multipoint (P2MP) EVC, see Supported Encapsulation and Rewrite Operations, on page 25.

Understanding Bridge Domain

The bridge domain is an Ethernet broadcast domain internal to the device. The bridge domain enables you to decouple the VLAN from the broadcast domain. The bridge domain has one to many mapping with EFPs.

All the EFPs in a node for a specific EVC are grouped using the bridge domain. If EFPs belong to the same bridge domain and have the same bridge domain number, the EFPs receive traffic even if they have different VLAN numbers.

The bridge domain number is local to the node. Different nodes that are part of an EVC can have the same or different bridge domain number. However, the bridge domain number is unique for an EVC within a node.

For EVC, the bridge domain number is from 1 to 16384.

Configuration Constraint

The encapsulation and rewrite operations are not allowed if the bridge domain is configured on the EFP. Remove the existing bridge domain from the EFP and then change the encapsulation and rewrite operations.

Types of Bridge Domains

The bridge domain can be configured to operate in point–to–pointand point–to–multipoint modes. The default configuration mode of the bridge domain is point–to–multipoint.

The types of bridge domains supported are:

Bridge Domain Type	Description
Point–to–point	This bridge domain can be used for Ethernet Private Line (EPL) and Ethernet Virtual Private Line (EVPL). CPT supports up to 16384 point–to–point bridge domains. MAC learning is not supported for point–to–point bridge domains. REP does not block EFPs that are in point–to–point bridge domain. In a ring scenario, the point–to–multipoint bridge domain is needed.
Point–to–multipoint	This bridge domain can be used for Ethernet Private LAN (EPLAN) and Ethernet Virtual Private LAN (EVPLAN). CPT supports up to 4000 point–to–multipoint bridge domains. MAC learning is supported for point–to–multipoint bridge domains. The point–to–multipoint bridge domain is supported over REP.

EVC Features

EVC in CPT supports the following features:

- Create, delete, or modify EFPs
- Add EFPs as members of a bridge domain
- Map Traffic to EFPs based on:
  - 802.1q VLANS (Single VLAN, list, range)
  - Cisco Q-in-Q VLANS (Single outer and single inner VLAN)
  - Proprietary Q-in-Q VLANS (9100, 9200)
  - 802.1ad Provider Bridges (encapsulation and rewrite)
- Map VLAN—Push, Pop, Translate Single VLAN tag
- Support for rewriting single or double VLAN tags
- Support for grouping VLANs from several UNI to a single EVC
- Support for Ethernet UNI with dual VLAN tag (Cisco-QinQ or IEEE 802.1ad)

- Support for 802.1Q VLAN ID translation on the 802.1q tagged traffic on the UNI
- Support for point–to–point EVC, multipoint–to–multipoint EVC, and rooted multipoint EVC
- Support for Ethernet over MPLS
- Support for 1:2 VLAN translation and 2:2 VLAN translation
- EVC MAC address aging
- Flex Service Mapping (Advanced VLAN translations).
  - Support for dot1ad and Cisco Q-in-Q etype for S-tag
- Support for Layer 2 Protocol Tunneling (L2PT) for each port

EVC Types

CPT supports the following categories of EVCs:

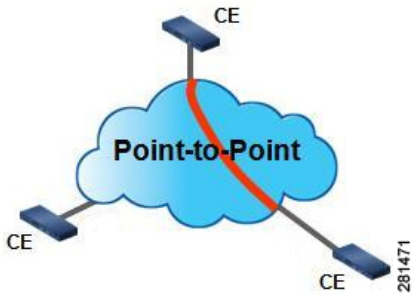
- Point–to–point EVCs (E-LINE services)
- Multipoint–to–multipoint EVCs (E-LAN services)
- Rooted Multipoint EVCs (E-TREE services). CTC handles E-TREE services as a special case of EPLAN/EVPLAN.

CPT supports the following types of EVCs:

Ethernet Private Line

An Ethernet Private Line (EPL) is a point–to–point EVC. EPL is an EVC that supports communication between two UNIs. In EPL, only one EVC can exist on a port and the port can have only one EFP. See Figure 1: Ethernet Private Line, on page 5.

Figure 1: Ethernet Private Line

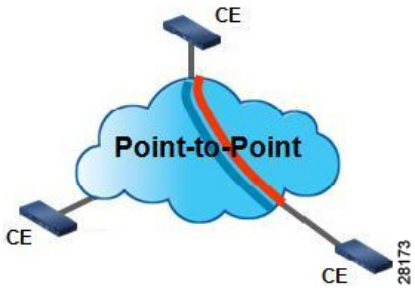


Ethernet Virtual Private Line

An Ethernet Virtual Private Line (EVPL) is a point–to–point EVC. EVPL is an EVC that supports communication between two UNIs. In EVPL, multiple EVCs can exist on a port and the port can have multiple

EFPs. Each EFP is associated with a different bridge domain. See Figure 2: Ethernet Virtual Private Line, on page 6.

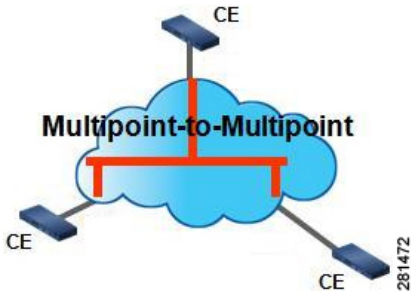
Figure 2: Ethernet Virtual Private Line



Ethernet Private LAN

An Ethernet Private LAN (EPLAN) is a multipoint–to–multipoint EVC. EPLAN is an EVC that supports communication between two or more UNIs. In EPLAN, only one EVC can exist on a port and the port can have only one EFP. See Figure 3: Ethernet Private LAN, on page 6.

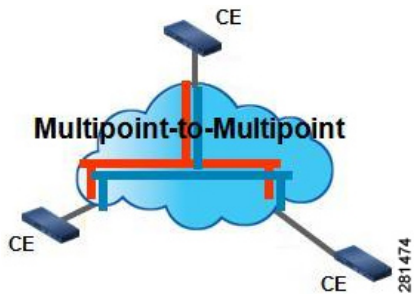
Figure 3: Ethernet Private LAN



Ethernet Virtual Private LAN

An Ethernet Virtual Private LAN (EVPLAN) is a multipoint–to–multipoint EVC. EVPLAN is an EVC that supports communication between two or more UNIs. In EVPLAN, multiple EVCs can exist on a port and the port can have multiple EFPs. Each EFP is associated with a different bridge domain. See Figure 4: Ethernet Virtual Private LAN, on page 6.

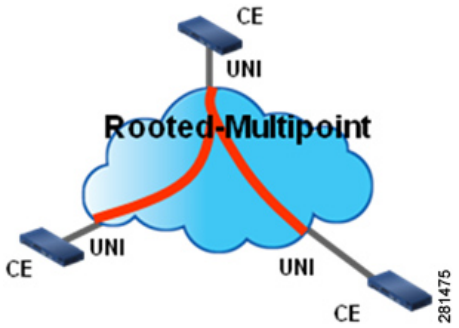
Figure 4: Ethernet Virtual Private LAN



Rooted Multipoint EVC

A rooted Multipoint EVC is a multipoint–to–multipoint EVC. In this EVC, the split horizon is configured on the access side EFPs. Figure 5: Rooted Multipoint EVC, on page 7 shows a rooted multipoint EVC with a split horizon configured between the two UNIs.

Figure 5: Rooted Multipoint EVC



CTC handles E-TREE services as a special case of EPLAN or EVPLAN.

Split Horizon

The rooted multipoint EVC supports split horizon. A split horizon is a subset of the members of a bridge domain.

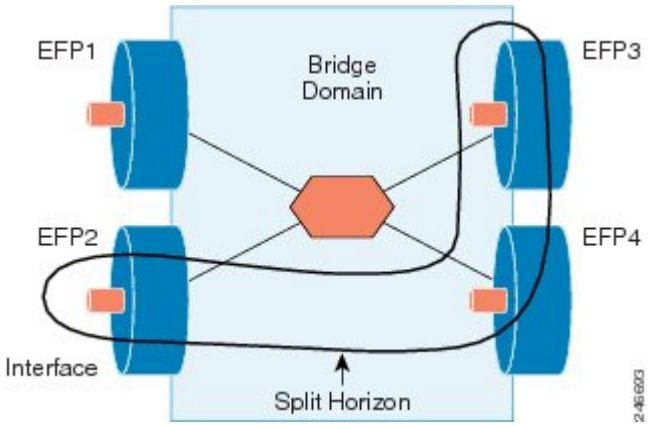
The split horizon can be enabled for the service instances that are members of the multipoint bridge domain. This disables traffic among all the members of the bridge domain where split horizon is configured.

Note CPT does not support split horizon groups.

Figure 6: Bridge Domain with Four EFPs, on page 8 shows an example of a multipoint bridge domain with four EFPs. The split horizon is configured on the three EFPs (EFP2, EFP3, and EFP4). The traffic is disabled among these three EFPs. Therefore, these three EFPs can only pass traffic to EFP1 and receive traffic from EFP1.

If the split horizon is configured on a Link Aggregation Group (LAG), the configurations apply to the entire LAG and not to the individual member ports.

Figure 6: Bridge Domain with Four EFPs



The following example configures a split horizon through Cisco IOS commands.

```
Router> enable
Router# configure terminal
Router(config)# interface TenGigabitEthernet 4/1
Router(config-if)# service instance 101 ethernet
Router(config-if-srv)# encapsulation  dot1q 100
Router(config-if-srv)# rewrite ingress tag push dot1q 20 symmetric
Router(config-if-srv)# bridge-domain 12 split-horizon
Router(config-if-srv)# exit
```

Counters

The following counters are supported for EFPs:

- Ingress packet counts
- Egress packet counts
- Ingress bytes
- Egress bytes

For point–to–multipoint bridge domains, all the counters are enabled by default. For point–to–point bridge domains, the ingress counters are enabled by default and the egress counters are disabled by default.

Note The hardware resources for point–to–point EFPs and point–to–multipoint EFPs are shared across Multiprotocol Label Switching (MPLS) and Quality of Service (Qos) and therefore, scalability of these counters for these EFPs are subject to availability of resources. A single 64-bit counter cannot be split among two byte counters or two packets counters.

Note The egress statistics for point–to–point EFPs is collected using a shared hardware resource. You can enable the egress statistics collection for each point–to– point EFP using the **evc-enable-stats** command.

EVC and EFP Limitations and Restrictions in CPT

These limitations and restrictions apply to EVC and EFP in CPT:

- The EFP point–to–point service does not support MAC learning and rewrite egress operations. It supports only the symmetric rewrite operation.
- The EFP multipoint–to–multipoint service supports the rewrite ingress option with the symmetric option. It does not support the rewrite egress operation.
- Different EFPs cannot have encapsulation for the same VLAN ID on a single interface. For example, dot1q 10 and dot1q 1-20 are not supported on a single interface because both include the same VLAN 10.
- Two different Ethernet types are not supported on a single interface. For example, encapsulations dot1q and dot1ad are not supported on the same interface.
- Rewrite Push 1 tag operation is not supported for encapsulations with double tag.
- Rewrite Push 2 tag operation is not supported for encapsulations with single or double tag.
- Translate rewrite operations are not supported for encapsulations such as untagged, any, default, and encapsulations involving VLAN range and list.
- If encapsulation default is configured on an EFP, no other encapsulation match on a EFP can be configured.
- Two EFPs on the same bridge domain and on the same interface is not supported.
- Encapsulation range limits—only up to 4 ranges are allowed for each EFP and only up to 8 VLAN ranges are allowed for each port.
- The point–to–point traffic flow is limited to 99% because of the 4 byte overhead that is added to the frames carrying point–to–point traffic.

Configuration Procedures

The following procedures can be performed using Cisco IOS commands to configure EVC and Layer 2 protocol tunneling:

- DLP-J216 Configure a Bridge Domain Using Cisco IOS Commands, on page 10
- DLP-J1 Configure an Ethernet Service Instance Using Cisco IOS Commands, on page 12
- DLP-J212 Configure Layer 2 Protocol Tunneling Using Cisco IOS Commands, on page 24

The following procedures can be performed using CTC to configure EVC and Layer 2 protocol tunneling:

- DLP-J2 Create an EVC Circuit Using CTC, on page 13

- DLP-J3 Edit an EVC Circuit Using CTC, on page 17
- DLP-J4 Query an EVC Circuit Using CTC, on page 20
- DLP-J213 Configure Layer 2 Protocol Tunneling Using CTC, on page 25

NTP-J1 Configure an EVC Circuit

Purpose	This procedure configures an EVC circuit.
Tools/Equipment	None
Prerequisite Procedures	None
Required/As Needed	As needed
Onsite/Remote	Onsite or remote
Security Level	Provisioning or higher

Procedure

Perform any of the following procedures as needed:

- DLP-J216 Configure a Bridge Domain Using Cisco IOS Commands, on page 10
- DLP-J1 Configure an Ethernet Service Instance Using Cisco IOS Commands, on page 12
- DLP-J2 Create an EVC Circuit Using CTC, on page 13
- DLP-J3 Edit an EVC Circuit Using CTC, on page 17
- DLP-J4 Query an EVC Circuit Using CTC, on page 20
- DLP-J212 Configure Layer 2 Protocol Tunneling Using Cisco IOS Commands, on page 24
- DLP-J213 Configure Layer 2 Protocol Tunneling Using CTC, on page 25

Stop. You have completed this procedure.

DLP-J216 Configure a Bridge Domain Using Cisco IOS Commands

Purpose	This procedure configures a bridge domain using Cisco IOS commands.
Tools/Equipment	None
Prerequisite Procedures	None

Required/As Needed	As needed
Onsite/Remote	Onsite or remote
Security Level	Provisioning or higher

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** Router> enable	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** Router# configure terminal	Enters global configuration mode.
Step 3	**bridge-domain** *bridge-id* [**split-horizon**] **Example:** Router(config)# bridge-domain 12	Creates a bridge domain where *bridge-id* is the identifier for the bridge domain instance.
Step 4	**mode p2p** **Example:** Router(config)# mode p2p	Configures the p2p (point–to–point) bridge domain. The default mode of the bridge domain is p2mp (point–to–multipoint).
Step 5	**no mode p2p** **Example:** Router(config)# no mode p2p	Configures the point–to–multipoint bridge domain.
Step 6	**exit** **Example:** Router(config)# exit	Exits global configuration mode.
Step 7	Return to your originating procedure (NTP). **Example:** —	

DLP-J1 Configure an Ethernet Service Instance Using Cisco IOS Commands

Purpose	This procedure configures an Ethernet service instance using Cisco IOS commands.
Tools/Equipment	None
Prerequisite Procedures	DLP-J216 Configure a Bridge Domain Using Cisco IOS Commands, on page 10
Required/As Needed	As needed
Onsite/Remote	Onsite or remote
Security Level	Provisioning or higher

Perform these steps to configure an EVC service instance under the point–to–multipoint bridge domain. To configure an EVC service instance under the point–to–point bridge domain, perform the DLP-J216 Configure a Bridge Domain Using Cisco IOS Commands, on page 10 procedure first and then perform this procedure.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** Router> enable	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** Router# configure terminal	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** Router(config)# interface TenGigabitEthernet 4/1	Specifies the interface to configure and enters interface configuration mode.
Step 4	**service instance** *id* **ethernet** [*evc-id*] **Example:** Router(config-if)# service instance 101 ethernet	Configures an Ethernet service instance on an interface and enters service instance configuration mode.
Step 5	**encapsulation dot1q** {*any* | *vlan-id* [*vlan-id* [*-vlan-id*]]} **second-dot1q** {*any* | *vlan-id* [*vlan-id* [*-vlan-id*]]} **Example:**	Defines the matching criteria that maps the ingress dot1q, QinQ, or untagged frames on an interface to the appropriate service instance.

	Command or Action	Purpose
	Router(config-if-srv)# encapsulation dot1q 100 second dot1q 200	
Step 6	**rewrite ingress tag** {**push** {**dot1q** *vlan-id* | **dot1q** *vlan-id* **second-dot1q** *vlan-id* | **dot1ad** *vlan-id* **dot1q** *vlan-id*} | **pop** {**1** | **2**} | **translate** {**1-to-1** {**dot1q** *vlan-id* | **dot1ad** *vlan-id*}| **2-to-1 dot1q** *vlan-id* | **dot1ad** *vlan-id*}| **1-to-2** {**dot1q** *vlan-id* **second-dot1q** *vlan-id* | **dot1ad** *vlan-id* **dot1q** *vlan-id*} | **2-to-2** {**dot1q** *vlan-id* **second-dot1q** *vlan-id* | **dot1ad** *vlan-id* **dot1q** *vlan-id*}} {**symmetric**} **Example:** Router(config-if-srv)# rewrite ingress tag push dot1q 20	Specifies the rewrite operation to be applied on the frame ingress to the service instance.
Step 7	**bridge-domain** *bridge-id* [**split-horizon**] **Example:** Router(config-if-srv)# bridge-domain 12	Binds the Ethernet service instance to a bridge domain instance where *bridge-id* is the identifier for the bridge domain instance.
Step 8	**exit** **Example:** Router(config-if-srv)# exit	Exits the service instance configuration mode.
Step 9	Return to your originating procedure (NTP). **Example:** —	

Configure an Ethernet Service Instance

The following example shows how to configure an Ethernet service instance using Cisco IOS commands.

```
Router> enable
Router# configure terminal
Router(config)# interface TenGigabitEthernet 4/1
Router(config-if)# service instance 101 ethernet
Router(config-if-srv)# encapsulation  dot1q 100
Router(config-if-srv)# rewrite ingress tag push dot1q 20 symmetric
Router(config-if-srv)# bridge-domain 12
Router(config-if-srv)# exit
```

DLP-J2 Create an EVC Circuit Using CTC

Purpose	This procedure creates an EVC circuit using CTC.

Tools/Equipment	None
Prerequisite Procedures	None
Required/As Needed	As needed
Onsite/Remote	Onsite or remote
Security Level	Provisioning or higher

Note The layer 2 services on CPT can be created on top of layer 2 PPCs or OCHTrails.

CPT supports Client/Trunk to Client/Trunk (layer 2) PPC for topology discovery and layer 2 service routing. If CPT is used in the GNE-ENE mode of configuration, then generic communications channels (GCC) can be used for topology discovery and layer 2 PPCs for layer 2 service routing. Both layer 2 PPC and GCC can be created between the same set of ports. However, this is not mandatory.

CPT also supports OCH Trunk to OCH Filter PPCs to connect CPT to MSTP nodes. If you want to route the traffic from a non co-located CPT node to a DWDM network, use OCH Trunk to OCH Filter PPCs to connect CPT and MSTP nodes. The OCHTrail can be created on top of this and the layer 2 services can be created on top of OCHTrails.

Procedure

Step 1 Complete the NTP-J22 Log into CTC procedure at a node where you want to create an EVC circuit.

Step 2 From the View menu, choose **Go to Home View**.

Step 3 Click the **Layer2+** tab.

Step 4 Click **Carrier Ethernet**.

Step 5 Click **Create**. The Circuit Creation wizard appears.

Step 6 In the Circuit Attributes screen of the wizard:

a) Enter the name of the service that you want to provision in the Name field.
When a name is not specified, CTC automatically creates a name with a random number at the end.

b) Enter the description of the service in the Description field.

c) From the EVC Type drop-down list, choose the service type that you want to provision.
The supported EVC types are:

- Ethernet Private Line
- Ethernet Virtual Private Line
- Ethernet Private LAN
- Ethernet Virtual Private LAN

d) From the EFP State drop-down list, choose UP or DOWN. The default value is UP. This EFP state maps to the EFP admin state in IOS.

e) Specify the bandwidth of the EVC in Kbps, Mbps, or Gbps and click **Next**.

Note The bandwidth value is used only for accounting purposes. The bandwidth value is not reserved by the CPT system for the actual traffic. The bandwidth value is not enforced on the EVC services. For example, if the actual traffic exceeds the bandwidth value, the CPT system carries the traffic and does not raise an alarm. The bandwidth value is not mapped to the values specified in the QoS policy.

Step 7 In the Source screen of the wizard:

a) From the Node drop-down list, choose the source node where you want to provision the EVC. The EFP A section appears. The Shelf field displays the shelf automatically assigned to the EVC.

b) To choose a port, complete the following:

1. From the Fabric/Line/Satellite Slot drop-down list, choose a slot.
2. From the Port drop-down list, choose a port to serve as the source EFP.

c) If you want to choose a channel group to serve as the source EFP:

1. Check the **CHGRP as EFP** check box.
2. From the CHGRP drop-down list, choose a channel group to serve as the source EFP.
3. Click **Manual Load Balancing** to configure manual load balancing on the ports of the channel group. The Manual Load Balancing dialog box appears.
4. From the Primary Loadbalanced Link list, choose a port.
5. Click **Apply**.

d) Click **Next**.

Step 8 In the Destination screen of the wizard:

a) From the Node drop-down list, choose the destination node that you want to provision the EVC. The EFP Z section appears.

b) If you want to choose a port to serve as the destination EFP, complete the following:

1. From the Fabric/Line/Satellite Slot drop-down list, choose a slot.
2. From the Port drop-down list, choose a port to serve as the destination EFP.

c) If you want to choose a channel group to serve as the destination EFP:

1. Check the **CHGRP as EFP** check box.
2. From the CHGRP drop-down list, choose a channel group to serve as the destination EFP.
3. Click **Manual Load Balancing** to configure manual load balancing on the ports of the channel group. The Manual Load Balancing dialog appears.
4. From the Primary Loadbalanced Link list, choose a port.
5. Click **Apply**.

d) Click **Next**.

Step 9 In the EVC Circuit Routing Preview screen of the wizard, CTC displays the shortest route of the EVC circuit. You can specify the nodes that need to be included or excluded in the EVC circuit.

a) Click the **Constraints** tab.

b) Choose the nodes appropriately and click **Include** or **Exclude**.
c) Click **Apply** to apply the constraints.
d) Click **Next**.

Step 10 In the EFP Configuration Preview screen of the wizard, specify the VLAN configuration for EFPs.

a) Select an EFP in the EVC path.
The node and ports are populated in the EFP Selection area.

b) In the Outer VLAN Configuration area, choose the type of VLAN tagging:

- Double Tagged
- Single Tagged
- Untagged
- Default
- Any

Note The VLAN tagging type chosen for Ethernet Private Line and Ethernet Private LAN is Default. Do not change this option for the source EFP.

c) From the TPID drop-down list, choose a TPID—dot1q, dot1ad, 0x9100, or 0x9200.
d) Enter a VLAN tag in the VLAN Tag field. For example, enter 10,20,30-50 without white spaces in the VLAN Tag field.
e) Check the **exact** check box to specify the exact encapsulation value.
f) In the Inner VLAN Configuration area, enter the VLAN tag. You cannot enter VLAN range for inner VLANs. The TPID is dot1q and cannot be changed.
g) In the Rewrite Ingress Operation area, choose the rewrite operation:

- PUSH 1
- PUSH 2
- POP 1
- POP 2
- TRANSLATE 1-to-1
- TRANSLATE 1-to-2
- TRANSLATE 2-to-1
- TRANSLATE 2-to-2

h) From the Outer VLAN TPID drop-down list, choose a TPID—dot1q, dot1ad, 0x9100, or 0x9200.
i) Enter the outer VLAN tag in the Outer VLAN Tag field.
j) Enter the inner VLAN tag in the Inner VLAN Tag field. The Inner VLAN TPID is dot1q and cannot be changed.
k) (Only for Ethernet Private LAN and Ethernet Virtual Private LAN EVC types) In the Split Horizon area, check the **Enable Split Horizon** check box to enable the split horizon for the EFPs.
l) In the Enable Statistics area, check **Ingress** or **Egress** as appropriate.
m) Click **Save** to apply this configuration to the selected EFP.
n) To create a configuration for another EFP, select the node from the map and select the EFP from the Available Ports drop-down list.

If you have selected the Ethernet Private Line or Ethernet Virtual Private Line as the EVC type in the Circuit Attributes screen, you can configure only two EFPs, namely EFP A and EFP Z. If you have selected Ethernet Private LAN and Ethernet Virtual Private LAN in the Circuit Attributes screen, you can configure more than two EFPs.

You cannot configure same VLAN ID for different services.

o) Click **Apply to All** to derive the EFP configuration on all the intermediate EFPs in the EVC path from the source UNI and NNI EFP configuration.
The source node UNI and NNI EFP configuration that is specified are copied to the destination node UNI and NNI respectively. You can choose to copy the source node NNI configuration without rewrite to all the other EFPs. If the EVC has only one node, the **Apply to All** button does not function.

Step 11 Click **Finish** to create a EVC circuit. The created EVC circuit appear in the list of EVC circuits.

Step 12 Return to your originating procedure (NTP).

DLP-J3 Edit an EVC Circuit Using CTC

Purpose	This procedure edits an EVC circuit using CTC.
Tools/Equipment	None
Prerequisite Procedures	DLP-J2 Create an EVC Circuit Using CTC, on page 13
Required/As Needed	As needed
Onsite/Remote	Onsite or remote
Security Level	Provisioning or higher

This procedure allows you to perform the following:

- Create new end point EFPs for the EVC
- View the configurations of the EFPs
- Specify the QoS policies to apply on individual EFPs
- Specify the IGMP Snooping settings for the bridge domain
- Specify the MAC learning settings for the bridge domain
- Specify the multicast settings for the bridge domain

Procedure

Step 1 Complete the NTP-J22 Log into CTC procedure at a node where you want to edit an EVC circuit.

Step 2 From the View menu, choose **Go to Home View**.

Step 3 Click the **Layer2+** tab.

Step 4 Click **Carrier Ethernet**.

Step 5 From the list of EVC circuits, select an EVC circuit to edit.

Step 6 Click **Edit**. The Edit Circuit dialog box appears.

Step 7 In the General tab, view the name, service ID, description, EVC type, and bandwidth of the EVC circuit.

Step 8 In the Endpoint EFPs tab, view the EFPs that are part of this EVC. You can create new end points only for Ethernet Private LAN and Ethernet Virtual Private LAN. To create a new end point EFP for this EVC:

a) Click **Create**. The Define New Drop wizard appears.

b) In the New Drop screen of the wizard, choose a node from the Node drop-down list.

c) To choose a port to serve as the EFP:

1. From the Fabric/Line/Satellite Slot drop-down list, choose a slot.
2. From the Port drop-down list, choose a port to serve as the EFP.

d) To choose a channel group to serve as the EFP:

1. Check the **CHGRP as EFP** check box.
2. From the CHGRP drop-down list, choose a channel group to serve as the EFP.
3. Click **Manual Load Balancing** to configure manual load balancing on the ports of the channel group. The Manual Load Balancing dialog box appears.
4. From the Primary Loadbalanced Link list, choose a port.
5. Click **Apply**.

e) Click **Next**.

f) In the EVC Circuit Routing Preview screen of the wizard, CTC displays the route of the EVC circuit. Specify the nodes that need to be included or excluded in the EVC circuit.

g) Click the **Constraints** tab.

h) Choose the nodes appropriately to include or exclude.

i) Click **Apply** to apply the constraints.

j) Click **Next**.

k) In the EFP Configuration Preview screen of the wizard, specify the VLAN configuration for this EFP.

l) Click **Finish** to create a new EFP for this EVC.

Step 9 In the EFP Configuration tab, view the configurations of the EFPs:

a) From the EFP drop-down list, choose an EFP to view its configuration. The VLAN configurations of the EFP cannot be changed.

b) From the EFP State drop-down list, choose UP or Down to change the up or down status of EFP.

c) Click **Apply**.

Step 10 In the QoS tab, specify the QoS policies to apply on the individual EFPs:

a) Check the **Enable/Disable Ingress QoS** check box as appropriate.

All the configured ingress QoS policies are populated in the Ingress Policy drop-down list.

b) From the Ingress Policy drop-down list, choose the required policy.
c) Check the **Enable/Disable Egress QoS** check box as appropriate.
All the configured egress QoS policies are populated in the Egress Policy drop-down list.
d) From the Egress Policy drop-down list, choose the required policy.
e) Click **Apply**.

Step 11 (Only for Ethernet Virtual Private LAN EVC type) In the IGMP Snooping tab, specify the settings for the bridge domain.
a) Check the **IGMP Snooping** check box to enable IGMP snooping on this bridge domain.
b) Check the **Immediate Leave** check box. When you enable IGMP immediate leave, IGMP snooping immediately removes a port when it detects an IGMP version 2 leave message on that port.
c) Check the **Report Suppression** check box. When you enable report suppression, the bridge domain forwards only one IGMP report for each multicast query.
d) Check the **IGMP Static Router Port** check box to add a static router to the EFP.
e) Click **Apply**.

Step 12 (Only for Ethernet Private LAN and Ethernet Virtual Private LAN EVC types) In the Mac Learning tab, specify the MAC learning settings for the bridge domain.
a) Check the **MAC Learning** check box to enable MAC learning on this bridge domain. MAC learning is enabled by default for Ethernet Private LAN and Ethernet Virtual Private LAN.
b) Enter the upper limit on the number of MAC addresses that reside in a bridge domain in the Limit field. The maximum MAC address limit on a bridge domain is 128000.
c) Click **Apply**.
d) Click **EFP Static MAC Address Configuration** to enter static MAC addresses for each EFP. The EFP Static MAC Address Configuration dialog box appears.
e) From the EFP drop-down list, choose an EFP.
f) Enter one or more static MAC addresses for this EFP in the MAC Address field. The added MAC addresses appear in the Entered MAC Addresses area.
g) Click **Apply** and close the EFP Static MAC Address Configuration dialog box.
h) Click **Clear MAC Address(es)** to remove a specific MAC addresses from the MAC address table. The Clear MAC Addresses dialog box appears.
i) Select the CPT System where you want to clear the MAC address from.
j) Enter the MAC address in the MAC Address field and click **Add**.
k) Click **Clear** to clear all the MAC addresses in the MAC Addresses to clear area.
l) Click **Clear All** to clear all the MAC addresses learned on the CPT system.
m) Close the Clear MAC Addresses dialog box.
n) Click **Display MAC Address(es)** to display the configured static MAC addresses for each EFP. The Configured EFP Static MAC Addresses dialog box appears.
o) From the EFP drop-down list, choose an EFP.
The MAC addresses configured on the EFP appear in the Configured MAC Addresses area.
p) Close the Configured EFP Static MAC Addresses dialog box.

Step 13 (Only for Ethernet Virtual Private LAN EVC type) In the MVR tab, specify the multicast settings for the bridge domain:
a) Check the **MVR** check box to enable MVR for this bridge domain.
b) Click **Apply**.

c) Click **Multicast IP Address Configuration** to add multicast IP addresses for this bridge domain. The Multicast IP Addresses dialog box appears.
d) Enter one or more multicast IP address in the IP Address field and click **Add**. The added multicast addresses appear in the IP Addresses area.
e) Click **Apply** and close the Multicast IP Addresses dialog box.
f) From the MVR Type drop-down list, choose **None**, **Source** or **Receiver** for each EFP.
g) Click the Source Service ID field and select a MVR enabled service.
h) Check the **Immediate Leave** check box. When you enable immediate leave, MVR immediately removes a port when it detects a leave message on that port.
i) Click **Apply**.

Step 14 Return to your originating procedure (NTP).

DLP-J4 Query an EVC Circuit Using CTC

Purpose	This procedure allows you to discover the EVC services using CTC.
Tools/Equipment	None
Prerequisite Procedures	DLP-J2 Create an EVC Circuit Using CTC, on page 13
Required/As Needed	As needed
Onsite/Remote	Onsite or remote
Security Level	Provisioning or higher

Note When the discovered nodes are disconnected, the circuits move to Partial state. When the disconnected nodes become online in CTC, re-query the circuits to move the circuits to Discovered state.

Procedure

Step 1 Complete the NTP-J22 Log into CTC procedure at a node where you want to query for an EVC circuit.
Step 2 From the View menu, choose **Go to Home View**.
Step 3 Click the **Layer2+** tab.
Step 4 From the left pane, click **Carrier Ethernet**.
Step 5 Click **Query**. The L2 Services Query dialog box appears.
Step 6 From the Existing/New Query drop-down list, choose an existing query or a new query.
Step 7 In the Equipment Termination area, choose **Port** or **Query Group**.
Step 8 If you choose Port, specify the following:

a) Click **Port**. The Port/Channel Group Selection dialog box appears.
b) Choose the node, card, and port/channel group and click **OK**.
c) Close the Port/Channel Group Selection dialog box.

Step 9 If you choose Query Group, specify the following:
a) Click **Query Group**. The User Query Group Chooser dialog box appears.
b) From the Group drop-down list, choose a query group.
c) Add the nodes that can be grouped for the query from the Available Nodes area to the Grouped Nodes area.
d) Click **Save** to save the query group and close the User Query Group Chooser dialog box.

Step 10 In the L2 Services Query dialog box, click **Save**. The Store a Set of Query Criteria dialog box appears.

Step 11 Enter the query name in the Name field and click **Save** to save the query.

Step 12 In the L2 Services Query dialog box, click **Run Query**.
The results of the query appear in the Service Query Results area.

Step 13 Click **Discover All** to discover all the EVC services or click **Discover Selected** to discover the selected EVC services.
Close the L2 Services Query dialog box. The discovered EVC services appear in the Carrier Ethernet Circuits area.

Step 14 Return to your originating procedure (NTP).

Interactions of EVC with Other Features

EVC interacts with the following features.

- LAG
- REP
- MPLS
- Dot1ad and Layer 2 Protocol Tunneling
- MAC learning and MAC address limiting
- QoS
- High Availability
- MVR
- IGMP Snooping

EVC with LAG

EFPs can be configured on a channel group. The traffic, carried by the EFPs, is load balanced across the member links. Ingress traffic for a single EVC can arrive on any member of the bundle. All egress traffic for an EFP uses only one of the member links. The load balancing is achieved by distributing EFPs between the member links. The EFPs on a channel group are grouped and each group is associated with a member link. In the default load balancing mechanism, there is no control over how the EFPs are distributed together, and

sometimes the EFP distribution is not ideal. The manual load balancing mechanism can be alternatively used to control the EFP grouping.

When you configure a physical port as part of a channel group, you cannot configure EVCs under that physical port.

The number of LAGs supported is 128 with 8 member links for each LAG.

LACP protocol is supported on the LAG.

EVC with REP

EVC supports up to 32 segments. You can configure REP over EVC using the cross–connect or using the bridge domain at the service instance level. REP is not supported on service instances configured with encapsulation untagged or default type.

REP is not supported for Ethernet Private Line and Ethernet Virtual Private Line services.

EVC with MPLS

MPLS pseudowire circuits can be configured over the service instance infrastructure using xconnect commands to bind the EFPs.

All the encapsulation and rewrite operations are supported for MPLS pseudowire EFPs except the following:

- Symmetric rewrites do not support push 2 tag operations.
- Symmetric 1–to–2 and 2–to–1 translate operations are not supported.

EVC with Layer 2 Protocol Tunneling

CPT supports Layer 2 protocol tunneling only at the interface level. Configurations applied at the interface level are applicable to all the EFPs configured on that interface.

The following port actions are supported in this release:

- Forward—Forwards the unmodified ingress BPDUs on the data path.
- Drop—Drops the ingress BPDUs on the interface.
- Peer—Punts BPDUs to the local instance of the protocol.

Note The pass option, tunnel option, and Layer 2 protocol tunneling at the EFP are not supported in this release.

The following protocols are supported for each port action:

Port Action	Supported Protocols
Peer	LACP, CDP
Drop	STP, VTP, DTP, PAGP, DOT1X, LACP, CDP
Forward	STP, VTP, DTP, PAGP, DOT1X, LACP, CDP

The following table lists the default port action for each protocol:

Protocol	Default Port Action
CDP	Peer
VTP	Forward
DTP	Forward
STP	Forward
PAGP	Forward
LACP	Peer
DOT1X	Forward

See DLP-J212 Configure Layer 2 Protocol Tunneling Using Cisco IOS Commands, on page 24 and DLP-J213 Configure Layer 2 Protocol Tunneling Using CTC, on page 25 to configure Layer 2 protocol tunneling.

EVC with MAC Learning and MAC Address Limiting

MAC learning is supported and enabled (by default) only for point–to–multipoint bridge domains. MAC learning can be enabled or disabled for point–to–multipoint bridge domains.

The MAC address limiting for bridge domains provides the capability to control the MAC address learning behavior at the bridge domain level. You can configure an upper limit on the number of MAC addresses that reside in a bridge domain. The remaining MAC addresses are flooded because they are not learned.

The MAC address limiting commands are configured under the bridge domain.

The default MAC address limit on a bridge domain is 1024. The maximum MAC address limit on a bridge domain is 128000.

EVC with QoS

See EVCS QoS Support.

EVC with High Availability

All the EFP configurations are synchronized between the active and standby fabric cards.

EVC with MVR

Multicast VLAN Registration (MVR) is supported only for point–to–multipoint services. Twenty bridge domains can be configured for MVR. The multicast traffic flows from MVR source EFP to multiple MVR receiver EFPs.

EVC with IGMP Snooping

Internet Group Management Protocol Snooping (IGMP snooping) is supported only for point–to–multipoint services. IGMP snooping can be enabled only at the bridge domain level.

DLP-J212 Configure Layer 2 Protocol Tunneling Using Cisco IOS Commands

Purpose	This procedure configures Layer 2 protocol tunneling using Cisco IOS commands.
Tools/Equipment	None
Prerequisite Procedures	None
Required/As Needed	As needed
Onsite/Remote	Onsite or remote
Security Level	Provisioning or higher

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** Router> enable	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** Router# configure terminal	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** Router(config)# interface TenGigabitEthernet 4/1	Specifies the interface to configure and enters interface configuration mode.
Step 4	**l2protocol** [**drop**|**forward**|**peer**] [**cdp**|**dot1x**|**dtp**|**lacp**|**pagp**|**stp**|**vtp**] **Example:** Router(config-if)# l2protocol forward cdp	Configures Layer 2 protocol tunneling actions for each interface.
Step 5	**exit** **Example:** Router(config-if)# exit	Returns to privileged EXEC mode.
Step 6	Return to your originating procedure (NTP).	—

DLP-J213 Configure Layer 2 Protocol Tunneling Using CTC

Purpose	This procedure configures Layer 2 protocol tunneling actions for each port using CTC.
Tools/Equipment	None
Prerequisite Procedures	None
Required/As Needed	As needed
Onsite/Remote	Onsite or remote
Security Level	Provisioning or higher

Procedure

Step 1 Complete the NTP-J22 Log into CTC procedure at a node where you want to configure Layer 2 protocol tunneling.

Step 2 In node view, right-click the fabric or line card and choose **Open Packet Transport System View**. The Packet Transport System View dialog box appears.

Step 3 Double-click a fabric card, line card, or CPT50 panel.

Step 4 Click the **Provisioning > Ether Ports > Ethernet** tabs.

Step 5 Click the **Configure/Edit L2PT** link under L2PT Config field for each port. The L2PT Config dialog box appears.

Step 6 From the Action drop-down list, choose **Drop**, **Forward**, or **Peer** for each Layer 2 protocol.

Step 7 Click **Apply**.
See DLP-J15 Create a Channel Group Using CTC to configure Layer 2 protocol tunneling actions for the member interfaces of the channel group.

Step 8 Return to your originating procedure (NTP).

Supported Encapsulation and Rewrite Operations

The following table lists the supported encapsulation and rewrite operations for point–to–point (P2P) EVC.

Table 1: Supported Encapsulation and Rewrite Operations for P2P EVC

Encapsulation Criterion	Ingress Rewrite Action
encapsulation default	No rewrite
encapsulation dot1q any	No rewrite
encapsulation dot1q range	No rewrite

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1q list	No rewrite
encapsulation untagged	No rewrite
encapsulation untagged, dot1q range, list	No rewrite
encapsulation dot1q vlan<id>	No rewrite
encapsulation dot1q vlan<id> exact	No rewrite
encapsulation dot1q range, list exact	No rewrite
encapsulation dot1q range exact	No rewrite
encapsulation dot1q list exact	No rewrite
encapsulation dot1q any second-dot1q vlan <id>	No rewrite
encapsulation dot1q range second-dot1q vlan<id>	No rewrite
encapsulation dot1q vlan <id> second-dot1q vlan<id>	No rewrite
encapsulation dot1q vlan<id> second-dot1q list	No rewrite
encapsulation dot1ad any	No rewrite
encapsulation dot1ad range	No rewrite
encapsulation dot1ad vlan<id>	No rewrite
encapsulation dot1ad vlan<id> exact	No rewrite
encapsulation dot1ad any dot1q vlan<id>	No rewrite
encapsulation dot1ad range dot1q vlan <id>	No rewrite
encapsulation dot1ad vlan <id> dot1q list	No rewrite
encapsulation dot1q any vlan-type <type value>	No rewrite
encapsulation dot1q range vlan-type <type value>	No rewrite
encapsulation dot1q vlan<id> vlan-type <type value>	No rewrite
encapsulation dot1q vlan<id> vlan-type <type value> exact	No rewrite
encapsulation dot1q any vlan-type <type value> dot1q vlan<id>	No rewrite
encapsulation dot1q range vlan-type <type value> dot1q vlan <id>	No rewrite
encapsulation dot1q vlan <id> vlan-type <type value> dot1q vlan<id>	No rewrite
encapsulation dot1q vlan <id> vlan-type <type value> dot1q list	No rewrite
encapsulation default	rewrite ingress tag push dot1q vlan<id>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1q any	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q range	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q list	rewrite ingress tag push dot1q vlan<id>
encapsulation untagged	rewrite ingress tag push dot1q vlan<id>
encapsulation untagged, dot1q range, list	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q vlan<id>	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q range, list exact	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q range exact	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q list exact	rewrite ingress tag push dot1q vlan<id>
encapsulation default	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q any	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q range	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q list	rewrite ingress tag push dot1ad vlan<id>
encapsulation untagged	rewrite ingress tag push dot1ad vlan<id>
encapsulation untagged, dot1q range, list	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q vlan<id>	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q range, list exact	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q range exact	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q list exact	rewrite ingress tag push dot1ad vlan<id>
encapsulation default	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q any	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q range	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q list	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation untagged	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation untagged, dot1q range, list	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1q vlan<id>	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q vlan<id> exact	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q range, list exact	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q range exact	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q list exact	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation untagged	rewrite ingress tag push dot1q vlan<id> second-dot1q vlan<id>
encapsulation untagged	rewrite ingress tag push dot1ad vlan <id> dot1q vlan<id>
encapsulation untagged	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>second-dot1q vlan<id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan <id> vlan-type <type value> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1ad vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1ad vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1ad vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id> vlan-type <type value>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id> vlan-type <type value>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id> vlan-type <type value>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> second-dot1q vlan <id>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> vlan-type <type value> second-dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> vlan-type <type value> second-dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> vlan-type <type value> second-dot1q vlan <id>

The following table lists the supported encapsulation and rewrite operations for point–to–multipoint (P2MP) EVC :

Table 2: Supported Encapsulation and Rewrite Operations for P2MP EVC

Encapsulation Criterion	Ingress Rewrite Action
encapsulation default	No rewrite
encapsulation dot1q any	No rewrite
encapsulation dot1q range	No rewrite
encapsulation dot1q list	No rewrite
encapsulation untagged	No rewrite
encapsulation untagged, dot1q range, list	No rewrite
encapsulation dot1q vlan<id>	No rewrite
encapsulation dot1q vlan<id> exact	No rewrite
encapsulation dot1q range, list exact	No rewrite
encapsulation dot1q range exact	No rewrite
encapsulation dot1q list exact	No rewrite
encapsulation dot1q any second-dot1q vlan <id>	No rewrite

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1q range second-dot1q vlan<id>	No rewrite
encapsulation dot1q vlan <id> second-dot1q vlan<id>	No rewrite
encapsulation dot1q vlan<id> second-dot1q list	No rewrite
encapsulation dot1ad any	No rewrite
encapsulation dot1ad range	No rewrite
encapsulation dot1ad vlan<id>	No rewrite
encapsulation dot1ad vlan<id> exact	No rewrite
encapsulation dot1ad any dot1q vlan<id>	No rewrite
encapsulation dot1ad range dot1q vlan <id>	No rewrite
encapsulation dot1ad vlan <id> dot1q vlan<id>	No rewrite
encapsulation dot1ad vlan <id> dot1q list	No rewrite
encapsulation dot1q any vlan-type <type value>	No rewrite
encapsulation dot1q range vlan-type <type value>	No rewrite
encapsulation dot1q vlan<id> vlan-type <type value>	No rewrite
encapsulation dot1q vlan<id> vlan-type <type value> exact	No rewrite
encapsulation dot1q any vlan-type <type value> dot1q vlan<id>	No rewrite
encapsulation dot1q range vlan-type <type value> dot1q vlan <id>	No rewrite
encapsulation dot1q vlan <id> vlan-type <type value> dot1q vlan<id>	No rewrite
encapsulation dot1q vlan <id> vlan-type <type value> dot1q list	No rewrite
encapsulation default	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q any	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q range	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q list	rewrite ingress tag push dot1q vlan<id>
encapsulation untagged	rewrite ingress tag push dot1q vlan<id>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation untagged, dot1q range, list	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q vlan<id>	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q range, list exact	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q range exact	rewrite ingress tag push dot1q vlan<id>
encapsulation dot1q list exact	rewrite ingress tag push dot1q vlan<id>
encapsulation default	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q any	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q range	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q list	rewrite ingress tag push dot1ad vlan<id>
encapsulation untagged	rewrite ingress tag push dot1ad vlan<id>
encapsulation untagged, dot1q range, list	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q vlan<id>	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q range, list exact	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q range exact	rewrite ingress tag push dot1ad vlan<id>
encapsulation dot1q list exact	rewrite ingress tag push dot1ad vlan<id>
encapsulation default	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q any	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q range	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q list	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation untagged	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation untagged, dot1q range, list	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q vlan<id>	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q vlan<id> exact	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1q range, list exact	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q range exact	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation dot1q list exact	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>
encapsulation untagged	rewrite ingress tag push dot1q vlan<id> second-dot1q vlan<id>
encapsulation untagged	rewrite ingress tag push dot1ad vlan <id> dot1q vlan<id>
encapsulation untagged	rewrite ingress tag push dot1q vlan<id> vlan-type <type value>second-dot1q vlan<id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value>	rewrite ingress tag translate 1-to-1 dot1q vlan <id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1q vlan <id> vlan-type <type value> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1ad vlan <id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan <id> vlan-type <type value> dot1q vlan<id>	rewrite ingress tag translate 1-to-1 dot1q vlan <id> vlan-type <type-value>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan<id>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1q vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1ad vlan<id>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1ad vlan<id> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value>	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag translate 1-to-2 dot1q vlan <id> vlan-type <type-value> second-dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value>	rewrite ingress tag translate 2-to-1 dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1ad vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1ad vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1ad vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id> vlan-type <type value>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id> vlan-type <type value>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-1 dot1q vlan <id> vlan-type <type value>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> second-dot1q vlan <id>

Encapsulation Criterion	Ingress Rewrite Action
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> second-dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1ad vlan <id> dot1q vlan <id>
encapsulation dot1q vlan <id> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> vlan-type <type value> second-dot1q vlan <id>
encapsulation dot1ad vlan <id> dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> vlan-type <type value> second-dot1q vlan <id>
encapsulation dot1q vlan <id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag translate 2-to-2 dot1q vlan <id> vlan-type <type value> second-dot1q vlan <id>
encapsulation dot1q <vlan> second-dot1q <vlan>	rewrite ingress tag pop 2
encapsulation dot1ad <vlan> dot1q <vlan>	rewrite ingress tag pop 2
encapsulation dot1q <vlan> vlan-type <type value> second-dot1q <vlan>	rewrite ingress tag pop 2
encapsulation dot1q vlan<id>	rewrite ingress tag pop 1
encapsulation dot1q vlan<id> exact	rewrite ingress tag pop 1
encapsulation dot1q vlan<id> second-dot1q vlan<id>	rewrite ingress tag pop 1
encapsulation dot1ad vlan<id>	rewrite ingress tag pop 1
encapsulation dot1ad vlan<id> exact	rewrite ingress tag pop 1
encapsulation dot1ad vlan<id> dot1q vlan<id>	rewrite ingress tag pop 1
encapsulation dot1q vlan<id>vlan-type <type value>	rewrite ingress tag pop 1
encapsulation dot1q vlan<id> vlan-type <type value> exact	rewrite ingress tag pop 1
encapsulation dot1q vlan<id> vlan-type <type value> second-dot1q vlan<id>	rewrite ingress tag pop 1