Configure Virtual LANs in Layer 2 VPNs

The Layer 2 Virtual Private Network (L2VPN) feature enables Service Providers (SPs) to provide L2 services to geographically disparate customer sites.

A virtual local area network (VLAN) is a group of devices on one or more LANs that are configured so that they can communicate as if they were attached to the same wire, when in fact they are located on a number of different LAN segments. The IEEE's 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames.

VLANs are very useful for user and host management, bandwidth allocation, and resource optimization. Using VLANs addresses the problem of breaking large networks into smaller parts so that broadcast and multicast traffic does not consume more bandwidth than necessary. VLANs also provide a higher level of security between segments of internal networks.

The 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames. Cisco IOS XR software supports VLAN sub-interface configuration on Gigabit Ethernet and 10-Gigabit Ethernet interfaces.

The configuration model for configuring VLAN Attachment Circuits (ACs) is similar to the model used for configuring basic VLANs, where the user first creates a VLAN sub-interface, and then configures that VLAN in sub-interface configuration mode. To create an Attachment Circuit, you need to include the **l2transport** keyword in the **interface** command string to specify that the interface is a L2 interface.

VLAN ACs support the following modes of L2VPN operation:

- Basic Dot1Q Attachment Circuit—The Attachment Circuit covers all frames that are received and sent with a specific VLAN tag.
- QinQ Attachment Circuit—The Attachment Circuit covers all frames received and sent with a specific outer VLAN tag and a specific inner VLAN tag. QinQ is an extension to Dot1Q that uses a stack of two tags.

Encapsulation

Encapsulation defines the matching criteria that maps a VLAN, a range of VLANs. Different types of encapsulations are default, dot1q, dot1ad. The following are the supported encapsulation types:

- **encapsulation default**: Configures the default service instance on a port.
- **encapsulation dot1q vlan-id** : Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance.

- **encapsulation dot1ad vlan-id** : Defines the matching criteria to map 802.1ad frames ingress on an interface to the appropriate service instance.
- **encapsulation dot1q second-dot1q**: Defines the matching criteria to map Q-in-Q ingress frames on an interface to the appropriate service instance.
- **encapsulation dot1ad dot1q**: Defines the matching criteria to be used in order to map single-tagged 802.1ad frames ingress on an interface to the appropriate service instance.

Restrictions and Limitations

To configure VLANs for Layer 2 VPNs, the following restrictions are applicable.

- In a point-to-point connection, the two Attachment Circuits do not have to be of the same type. For example, a port mode Ethernet Attachment Circuit can be connected to a Dot1Q Ethernet Attachment Circuit.
- Pseudowires can run in VLAN mode or in port mode. A pseudowire running in VLAN mode always carries Dot1Q or Dot1ad tag(s), while a pseudowire running in port mode may or may NOT carry tags. To connect these different types of circuits, popping, pushing, and rewriting tags is required.
- The Attachment Circuits on either side of an MPLS pseudowire can be of different types. In this case, the appropriate conversion is carried out at one or both ends of the Attachment Circuit to pseudowire connection.
- When receiving single or double Dot1Q tagged traffic on an L2VPN pseudowire, the egress rewrite action Push 1 configured in an attachment circuit is not supported. The egress rewrite action Push 1 configured in an attachment circuit is supported only for untagged traffic received on an L2VPN pseudowire.

- Configure VLAN Subinterfaces, on page 2
- Introduction to Ethernet Flow Point, on page 5
- Configure VLAN Header Rewrite, on page 7
- VLAN Switch, on page 14

Configure VLAN Subinterfaces

Subinterfaces are logical interfaces created on a hardware interface. These software-defined interfaces allow for segregation of traffic into separate logical channels on a single hardware interface as well as allowing for better utilization of the available bandwidth on the physical interface.

Subinterfaces are distinguished from one another by adding an extension on the end of the interface name and designation. For instance, the Ethernet subinterface 23 on the physical interface designated TenGigE 0/1/0/0 would be indicated by TenGigE 0/1/0/0.23.

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

The subinterface Maximum Transmission Unit (MTU) is inherited from the physical interface with 4 bytes allowed for the 802.1Q VLAN tag.

The following modes of VLAN subinterface configuration are supported:

- Basic dot1q Attachment Circuit
- Basic dot1ad Attachment Circuit
- Q-in-Q Attachment Circuit

To configure a basic dot1q Attachment Circuit, use this encapsulation mode:

encapsulation dot1q *vlan extra-id*

To configure a basic dot1ad Attachment Circuit, use this encapsulation mode:

encapsulation dot1ad *vlan-id*

To configure a Q-in-Q Attachment Circuit, use the following encapsulation modes:

- **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id*
- **encapsulation dot1ad** *vlan-id* **dot1q** *vlan-id*

Restrictions and Limitations

To configure VLAN subinterface, the following restrictions are applicable.

- At least 64 VLAN-IDs in a VLAN list is required to overcome the limitation of only 9 VLAN ranges per NPU.
- For double-tagged packet, the VLAN range is supported only on the inner tag.
- VLANs separated by comma are called a VLAN list. VLAN list isn’t supported on the router.
- If 0x9100/0x9200 is configured as tunneling ether-type, then dot1ad (0x88a8) encapsulation isn’t supported.
- If any subinterface is already configured under a main interface, modifying the tunneling ether-type isn’t supported.
- Following limitations are applicable to both outer and inner VLAN ranges:
  - 32 unique VLAN ranges are supported per NPU.
  - The overlap between outer VLAN ranges on subinterfaces of the same Network Processor Unit (NPU) isn’t supported. A subinterface with a single VLAN tag that falls into a range configured on another subinterface of the same NPU is also considered an overlap.
  - The overlap between inner VLAN ranges on subinterfaces of the same NPU isn’t supported.
  - Range 'any' doesn’t result in explicit programming of a VLAN range in hardware and therefore doesn’t count against the configured ranges.

Configuration Example

Configuring a VLAN subinterface involves:

- Creating a Ten Gigabit Ethernet subinterface
- Enabling L2 transport mode on the interface

- Defining the matching criteria (encapsulation mode) to be used in order to map ingress frames on an interface to the appropriate service instance.

Configuration of Basic dot1q Attachment Circuit

```
Router# configure
Router(config)# interface TenGigE 0/0/0/10.1 l2transport
Router(config-if)# encapsulation dot1q 10
Router(config-if)# no shutdown
```

Running Configuration

```
configure
 interface TenGigE 0/0/0/10.1
  l2transport
  encapsulation dot1q 10
 !
!
```

Verification

Verify that the VLAN subinterface is active:

```
Router# show interfaces TenGigE 0/0/0/10.1

...
TenGigE0/0/0/10.1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 0011.1aac.a05a
  Layer 2 Transport Mode
  MTU 1518 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation 802.1Q Virtual LAN,
    Outer Match: Dot1Q VLAN 10
    Ethertype Any, MAC Match src any, dest any
  loopback not set,
...
```

```
Router#show interfaces TenGigE 0/0/0/1.101
TenGigabitEthernet0/0/0/1.101 is down, line protocol is down
   Interface state transitions: 0
   Hardware is VLAN sub-interface(s), address is 008a.9678.0c04
   Layer 2 Transport Mode
   MTU 1518 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
      reliability Unknown, txload Unknown, rxload Unknown
   Encapsulation 802.1Q Virtual LAN,
      Outer Match: Dot1Q VLAN 66-67,68-69,70-71,118-119,120-121,122-123,229,230,231
      Ethertype Any, MAC Match src any, dest any
   loopback not set,
   Last input never, output never
   Last clearing of "show interface" counters never
      0 packets input, 0 bytes
      0 input drops, 0 queue drops, 0 input errors
      0 packets output, 0 bytes
```

```
0 output drops, 0 queue drops, 0 output errors
```

Associated Commands

- encapsulation dot1ad dot1q
- encapsulation dot1q
- encapsulation dot1q second-dot1q
- l2transport (Ethernet)
- encapsulation dot1ad

Introduction to Ethernet Flow Point

An Ethernet Flow Point (EFP) is a Layer 2 logical sub-interface used to classify traffic under a physical or a bundle interface. An EFP is defined by a set of filters (a set of entries) that are applied to all the ingress traffic to classify the frames that belong to a particular EFP. Each entry usually contains 0, 1 or 2 VLAN tags. You can specify a VLAN or QinQ tagging to match against on ingress. A packet that starts with the same tags as an entry in the filter is said to match the filter; if the start of the packet does not correspond to any entry in the filter, then the packet does not match the filter.

All traffic on ingress are processed by that EFP if a match occurs, and this can in turn change VLAN IDs, add or remove VLAN tags, and change ethertypes. After the frames are matched to a particular EFP, any appropriate feature (such as, any frame manipulations specified by the configuration as well as things such as QoS and ACLs) can be applied.

The benefits of EFP include:

- Identifying all frames that belong to a particular flow on a given interface
- Performing VLAN header rewrites

  (See, Configure VLAN Header Rewrite, on page 7)
- Adding features to the identified frames
- Optionally defining how to forward the identified frames in the data path

Limitations of EFP

Egress EFP filtering is not supported on Cisco IOS XR.

Identify Frames of an EFP

The EFP identifies frames belonging to a particular flow on a given port, independent of their Ethernet encapsulation. An EFP can flexibly map frames into a flow or EFP based on the fields in the frame header. The frames can be matched to an EFP using VLAN tags.

The frames can’t be matched to an EFP through this:

- Any information outside the outermost Ethernet frame header and its associated tags such as

- IPv4, IPv6, or MPLS tag header data
- C-DMAC, C-SMAC, or C-VLAN

VLAN Tag Identification

Below table describes the different encapsulation types and the EFP identifier corresponding to each.

Encapsulation Type	EFP Identifier
Single tagged frames	802.1Q customer-tagged Ethernet frames
Double tagged frames	802.1Q (ethertype 0x8100) double tagged frames 802.1ad (ethertype 0x88a8) double tagged frames

You can use wildcards while defining frames that map to a given EFP. EFPs can distinguish flows based on a single VLAN tag, a stack of VLAN tags or a combination of both (VLAN stack with wildcards). It provides the EFP model, a flexibility of being encapsulation agnostic, and allows it to be extensible as new tagging or tunneling schemes are added.

Apply Features

After the frames are matched to a particular EFP, any appropriate features can be applied. In this context, “features” means any frame manipulations specified by the configuration as well as things such as QoS and ACLs. The Ethernet infrastructure provides an appropriate interface to allow the feature owners to apply their features to an EFP. Hence, IM interface handles are used to represent EFPs, allowing feature owners to manage their features on EFPs in the same way the features are managed on regular interfaces or sub-interfaces.

The only L2 features that can be applied on an EFP that is part of the Ethernet infrastructure are the L2 header encapsulation modifications. The L2 features are described in this section.

Encapsulation Modifications

EFP supports these L2 header encapsulation modifications on both ingress and egress:

- Push 1 or 2 VLAN tags
- Pop 1 or 2 VLAN tags

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

- Rewrite 1 or 2 VLAN tags:
  - Rewrite outer tag
  - Rewrite outer 2 tags
  - Rewrite outer tag and push an additional tag
  - Rewrite outer tag and pop inner tag

  For each of the VLAN ID manipulations, these can be specified:

- The VLAN tag type, that is, C-VLAN, S-VLAN, or I-TAG. The ethertype of the 802.1Q C-VLAN tag is defined by the dot1q tunneling type command.
- The VLAN ID. 0 can be specified for an outer VLAN tag to generate a priority-tagged frame.

Note For tag rewrites, the CoS bits from the previous tag should be preserved in the same way as the DEI bit for 802.1ad encapsulated frames.

Define Data-Forwarding Behavior

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

- L2 Switched Service (Bridging)—The EFP is mapped to a bridge domain, where frames are switched based on their destination MAC address. This includes multipoint services:
  - Ethernet to Ethernet Bridging
  - Multipoint Layer 2 Services
- L2 Stitched Service (AC to AC xconnect)—This covers point-to-point L2 associations that are statically established and do not require a MAC address lookup.
  - Ethernet to Ethernet Local Switching—The EFP is mapped to an S-VLAN either on the same port or on another port. The S-VLANs can be identical or different.
- Tunneled Service (xconnect)—The EFP is mapped to a Layer 3 tunnel. This covers point-to-point services, such as EoMPLS.

Configure VLAN Header Rewrite

EFP supports the following VLAN header rewrites on both ingress and egress ports:

- Push 1 VLAN tag
- Pop 1 VLAN tag

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

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

Various combinations of ingress, egress VLAN rewrites with corresponding tag actions during ingress and egress VLAN translation, are listed in the following sections:

Configuration Example

This topic covers VLAN header rewrites on various attachment circuits, such as:

- L2 single-tagged sub-interface
- L2 double-tagged sub-interface

Configuring VLAN header rewrite involves:

- Creating a TenGigabit Ethernet sub-interface
- Enabling L2 transport mode on the interface
- Defining the matching criteria (encapsulation mode) to be used in order to map single-tagged frames ingress on an interface to the appropriate service instance
- Specifying the encapsulation adjustment that is to be performed on the ingress frame

Configuration of VLAN Header Rewrite (single-tagged sub-interface)

```
Router# configure
Router(config)# interface TenGigE 0/0/0/10.1 l2transport
Router(config-if)# encapsulation dot1q 10
Router(config-if)# rewrite ingress tag push dot1q 20 symmteric
```

Running Configuration

```
/* Configuration without rewrite */

configure
 interface TenGigE0/0/0/0.1 l2transport
  encapsulation dot1q 10
 !
!

/* Configuration with rewrite */

/* PUSH 1 */
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
 rewrite ingress tag push dot1q 20 symmteric
 !
!

/* POP 1 */
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
  rewrite ingress tag pop 1
 !
!

/* TRANSLATE 1-1 */

interface TenGigE0/0/0/0.1 l2transport
```

```
 encapsulation dot1q 10
  rewrite ingress tag translate 1-to-1 dot1q 20
 !
!

/* TRANSLATE 1-2 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
  rewrite ingress tag translate 1-to-2 dot1q 20 second-dot1q 30
 !
!
```

Running Configuration (VLAN header rewrite on double-tagged sub-interface)

```
/* Configuration without rewrite */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
 !
!

/* Configuration with rewrite */

/* PUSH 1 */
interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag push dot1q 20 symmteric
 !
!

/* TRANSLATE 1-1 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 1-to-1 dot1q 20
 !
!

/* TRANSLATE 1-2 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 1-to-2 dot1q 20 second-dot1q 30
 !
!

/* TRANSLATE 2-1 */
interface TenGigE0/0/0/0.1 l2transport
encapsulation dot1q 10 second-dot1q 11
rewrite ingress tag translate 2-to-1 dot1q 20

/* TRANSLATE 2-2 */

interface TenGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10 second-dot1q 11
  rewrite ingress tag translate 2-to-2 dot1q 20 second-dot1q 30
 !
!
```

Associated Commands

- encapsulation dot1ad dot1q
- encapsulation dot1q
- encapsulation dot1q second-dot1q
- l2transport (Ethernet)
- rewrite ingress tag

Rewrite Encapsulation Combinations

The following table lists the supported and unsupported rewrite combinations:

Table 1: Rewrite Encapsulation Combinations

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
No rewrite	• untagged • default • dot1q range • dot1ad range • dot1q priority tagged • dot1ad priority tagged • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag	• dot1q any • dot1ad any

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
Pop 1	• dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag	• untagged • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q priority tagged • dot1ad priority tagged
Pop 2	• dot1q double tag • dot1ad double tag • custom 9100/9200 double tag	• untagged • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q priority tagged • dot1ad priority tagged • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
Push 1	• untagged • default • dot1q range • dot1ad range • dot1q priority tagged • dot1ad priority tagged • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag	• dot1q any • dot1ad any
Push 2	• untagged • dot1q priority tagged • dot1ad priority tagged • dot1q • dot1ad	• default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag • custom 9100/9200 double tag

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
Translate 1 to 1	• dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag	• untagged • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q priority tagged • dot1ad priority Tagged • custom 9100/9200 double tag
Translate 1 to 2	• dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any • dot1q double tag • dot1ad double tag	untagged Default dot1q range dot1q any dot1ad any dot1ad range dot1q priority tagged dot1ad priority Tagged Custom 9100/9200 double tag

Rewrite Action	Supported Encapsulation Type	Unsupported Encapsulation
Translate 2 to 2	• dot1q double tag • dot1ad double tag • custom 9100/9200 double tag	• untagged • default • dot1q range • dot1q any • dot1ad any • dot1ad range • dot1q priority tagged • dot1ad priority Tagged • dot1q • dot1ad • dot1q double inner tag range • dot1ad double inner tag range • dot1q double Inner tag any • dot1ad double inner tag any
translate 2-to-1	Not Supported	
dot1ad push 1	Not Supported	
dot1ad push 2	Not Supported	
dot1ad translate 1-to-1	Not Supported	
dot1ad translate 1-to-2	Not Supported	
dot1ad translate 2-to-2	Not Supported	
dot1ad translate 2-to-1	Not Supported	

VLAN Switch

The VLAN Switch feature enables L2 VLAN switching with minimal configuration. This feature allows you to configure L2 bridging without having to configure and manage separate bridge instances and sub-interfaces for each per VLAN L2 forwarding domain.

Prior to this feature, numerous sub-interfaces were required to configure and manage L2 bridging. Using separate sub-interfaces for each VLAN on a port overloads the system scalability and consumes hardware resources, slows down provisioning, and makes the device harder to manage.

Prior to Cisco IOS XR Software Release 6.5.1, an L2 VLAN sub-interface was required to be configured to either classify L2 traffic based on the VLAN tags included in the frame header, or to perform a rewrite on the frame header. This is achieved by adding, removing, or translating the VLAN tags in the frame header.

The Trunk level VLAN encapsulation provides a simpler configuration where you can directly configure a VLAN on a trunk interface, a physical interface, or LAG interface. You do not need a separate sub-interface to be configured.

A trunk is a point-to-point link between the device and another networking device. Trunks carry the traffic of multiple VLANs over a single link and allow you to extend VLANs across an entire network.

To deliver the traffic on a trunk port with several VLANs, the device uses the IEEE 802.1Q encapsulation (tagging) method that uses a tag that is inserted into the frame header. This tag carries information about the specific VLAN to which the frame and packet belong. This method allows packets that are encapsulated for several different VLANs to traverse the same port and maintain traffic separation between the VLANs. The encapsulated VLAN tag also allows the trunk to move traffic end-to-end through the network on the same VLAN.

Restrictions

The following restrctions are applicable on the Cisco NCS 5000 Router:

- dot1ad native VLAN is not supported.
- VLAN switched trunk cannot be configured on the same trunk (physical or LAG) as trunk l2 transport mode.
- To change the list of VLAN bridges, L2VPN vlan-switch instance must be deleted and recreated.
- A trunk interface (physical or LAG) cannot use a mix of 802.1Q outer tags 802.1ad outer tags.
- If dot1q native VLAN and VLAN switched trunk are both configured then the Native VLAN must be contained in the set of VLAN IDs matched by the **vlan-switched trunk** configuration.
- When VLAN switched trunk is configured, IP addresses must not be configured on a trunk.

Configure VLAN Switch

Perform this task to configure VLAN Switch.

Configuration Example

```
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#vlan-switch gl1
Router(config-l2vpn-vs)#vlan 1-200
Router(config-l2vpn-vs)#vni 501-700
Router(config-l2vpn-vs)#interface GigabitEthernet0/0/0/0
Router(config-l2vpn-vs)#interface GigabitEthernet0/0/0/1
Router(config-l2vpn-vs)#interface GigabitEthernet0/0/0/2
/* Binds the VLAN switch to three trunk interfaces GE/0/0/0-2 */
Router(config-l2vpn-vs)#commit


/* GE/0/0/0 accepts 802.1ad tagged VLAN traffic with VLAN IDs between 1 and 100.
Traffic is mapped (without outermost VLAN tag) to the per-VLAN bridge with the same VLAN
ID. */
```

```
Router#configure
Router(config)#interface GigabitEthernet0/0/0/0
Router(config)#vlan-switched trunk dot1ad 1-100
!

/* GE/0/0/1 accepts 802.1Q tagged VLAN traffic with VLAN IDs between 50 and 150.
Tagged traffic is mapped (without outermost VLAN tag) to the per-VLAN bridge with the same
 VLAN ID.
Also accepts untagged traffic that is assigned to per-VLAN bridge gl_vlan50. */

Router(config)#interface GigabitEthernet0/0/0/1
Router(config)#vlan-switched trunk dot1q 50-150 dot1q native vlan 50
!

/* GE/0/0/2 accepts all traffic and is statically assigned to the VLAN bridge gl_vlan100.
*/

Router(config)#interface GigabitEthernet0/0/0/2
Router(config)#vlan-switched access 100
!

Router(config-if)#commit
```

Running Configuration

```
l2vpn
 vlan-switch gl1
   vlan 1-200
   vni 501-700
   interface GigabitEthernet0/0/0/0
   interface GigabitEthernet0/0/0/1
   interface GigabitEthernet0/0/0/2
 !
!
interface GigabitEthernet0/0/0/0
 vlan-switched trunk dot1ad 1-100
!
interface GigabitEthernet0/0/0/1
 vlan-switched trunk dot1q 50-150 dot1q native vlan 50
!
interface GigabitEthernet0/0/0/2
 vlan-switched access 100
```

Verification

Verify the VLAN switch configuration. Reports the interface as being in VLAN switched trunk mode and gives the encapsulation and VLAN IDs that are matched.

```
Router#show interfaces GigabitEthernet 0/0/0/0
Wed May 23
08:07:39.869 PDT
GigabitEthernet0/0/0/0 is up, line protocol is up
    Interface state transitions: 1
    Hardware is GigabitEthernet, address is 02fe.08cb.26c5 (bia
02fe.08cb.26c5)
    Internet address is Unknown
    MTU 1518 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
       reliability 255/255, txload 0/255, rxload 0/255
    Encapsulation 802.1Q Virtual LAN,
    vlan-switched trunk port: Dot1ad VLAN 1-100
```

```
    Full-duplex, 1000Mb/s, unknown, link type is force-up
    output flow control is off, input flow control is off
    Carrier delay (up) is 10 msec
    loopback not set,
    Last link flapped 00:05:32
    Last input never, output never
    Last clearing of "show interface" counters never
    5 minute input rate 0 bits/sec, 0 packets/sec
    5 minute output rate 0 bits/sec, 0 packets/sec
       0 packets input, 0 bytes, 0 total input drops
       0 drops for unrecognized upper-level protocol
       Received 0 broadcast packets, 0 multicast packets
                0 runts, 0 giants, 0 throttles, 0 parity
       0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
       0 packets output, 0 bytes, 0 total output drops
       Output 0 broadcast packets, 0 multicast packets
       0 output errors, 0 underruns, 0 applique, 0 resets
       0 output buffer failures, 0 output buffers swapped out
       1 carrier transitions


Router#show interfaces GigabitEthernet 0/0/0/1
Wed May 23
08:07:49.049 PDT
GigabitEthernet0/0/0/1 is up, line protocol is up
    Interface state transitions: 1
    Hardware is GigabitEthernet, address is 024f.88a0.8c9d (bia
024f.88a0.8c9d)
    Internet address is Unknown
    MTU 1518 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
       reliability 255/255, txload 0/255, rxload 0/255
    Encapsulation 802.1Q Virtual LAN, Native VLAN Id 50,
    vlan-switched trunk port: Dot1Q VLAN 50-150
    Full-duplex, 1000Mb/s, unknown, link type is force-up
    output flow control is off, input flow control is off
    Carrier delay (up) is 10 msec
    loopback not set,
    Last link flapped 00:05:08
    Last input never, output never
    Last clearing of "show interface" counters never
    5 minute input rate 0 bits/sec, 0 packets/sec
    5 minute output rate 0 bits/sec, 0 packets/sec
       0 packets input, 0 bytes, 0 total input drops
       0 drops for unrecognized upper-level protocol
       Received 0 broadcast packets, 0 multicast packets
                0 runts, 0 giants, 0 throttles, 0 parity
       0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
       0 packets output, 0 bytes, 0 total output drops
       Output 0 broadcast packets, 0 multicast packets
       0 output errors, 0 underruns, 0 applique, 0 resets
       0 output buffer failures, 0 output buffers swapped out
       1 carrier transitions

Router#show interfaces GigabitEthernet 0/0/0/2
Wed May 23
08:07:57.681 PDT
GigabitEthernet0/0/0/2 is up, line protocol is up
    Interface state transitions: 1
    Hardware is GigabitEthernet, address is 0246.c822.daae (bia
0246.c822.daae)
    Internet address is Unknown
    MTU 1514 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
       reliability 255/255, txload 0/255, rxload 0/255
    Encapsulation 802.1Q Virtual LAN,
```

```
vlan-switched access port: VLAN 100
Full-duplex, 1000Mb/s, unknown, link type is force-up
output flow control is off, input flow control is off
Carrier delay (up) is 10 msec
loopback not set,
Last link flapped 00:05:16
Last input never, output never
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
   0 packets input, 0 bytes, 0 total input drops
   0 drops for unrecognized upper-level protocol
   Received 0 broadcast packets, 0 multicast packets
            0 runts, 0 giants, 0 throttles, 0 parity
   0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
   0 packets output, 0 bytes, 0 total output drops
   Output 0 broadcast packets, 0 multicast packets
   0 output errors, 0 underruns, 0 applique, 0 resets
   0 output buffer failures, 0 output buffers swapped out
   1 carrier transitions
```

Associated Commands

- vlan-switched trunk
- vlan-switched access
- show interfaces