Configuring IEEE 802.1Q Tunneling

Information About IEEE 802.1Q Tunneling

The IEEE 802.1Q Tunneling feature is designed for service providers who carry traffic of multiple customers across their networks and are required to maintain the VLAN and Layer 2 protocol configurations of each customer without impacting the traffic of other customers.

IEEE 802.1Q Tunnel Ports in a Service Provider Network

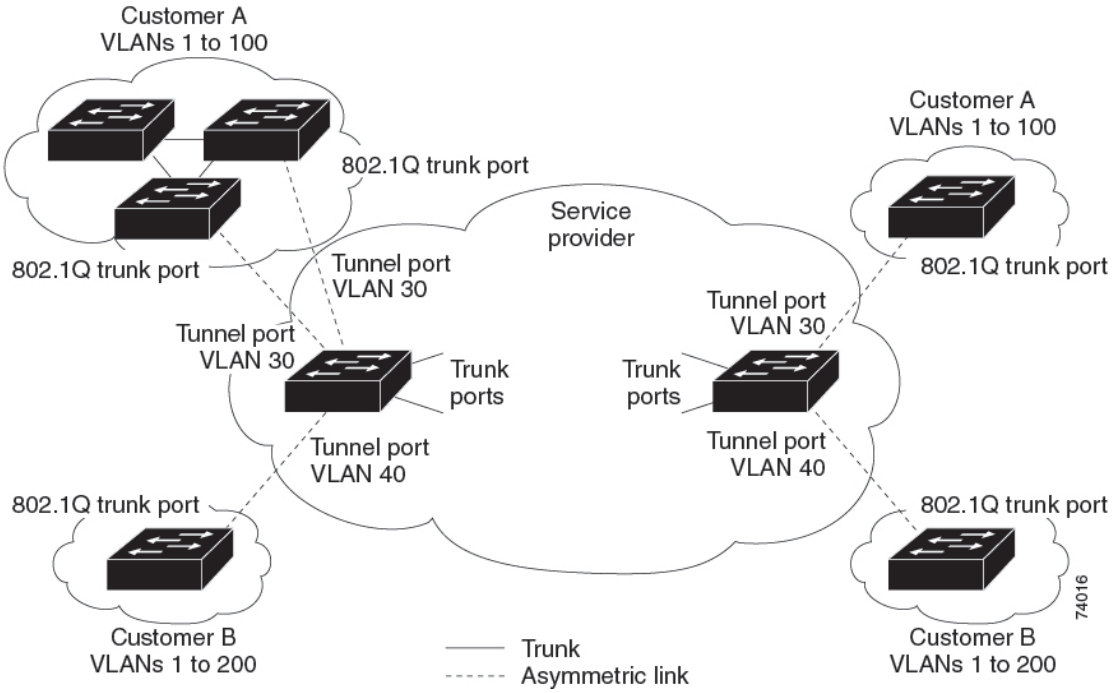
Business customers of service providers often have specific requirements for VLAN IDs and the number of VLANs to be supported. The VLAN ranges required by different customers in the same service-provider network might overlap, and traffic of customers through the infrastructure might be mixed. Assigning a unique range of VLAN IDs to each customer would restrict customer configurations and could easily exceed the VLAN limit (4096) of the IEEE 802.1Q specification.

Using the IEEE 802.1Q tunneling feature, service providers can use a single VLAN to support customers who have multiple VLANs. Customer VLAN IDs are preserved, and traffic from different customers is segregated within the service-provider network, even when they appear to be in the same VLAN. Using IEEE 802.1Q tunneling expands VLAN space by using a VLAN-in-VLAN hierarchy and retagging the tagged packets. A port configured to support IEEE 802.1Q tunneling is called a tunnel port. When you configure tunneling, you assign a tunnel port to a VLAN ID that is dedicated to tunneling. Each customer requires a separate service-provider VLAN ID, but that VLAN ID supports all of the customer's VLANs.

Customer traffic that is tagged in the normal way with appropriate VLAN IDs comes from an IEEE 802.1Q trunk port on the customer device and into a tunnel port on the service-provider edge device. The link between the customer device and the edge device is asymmetric because one end is configured as an IEEE 802.1Q

trunk port, and the other end is configured as a tunnel port. You assign the tunnel port interface to an access VLAN ID that is unique to each customer.

Figure 1: IEEE 802.1Q Tunnel Ports in a Service-Provider Network

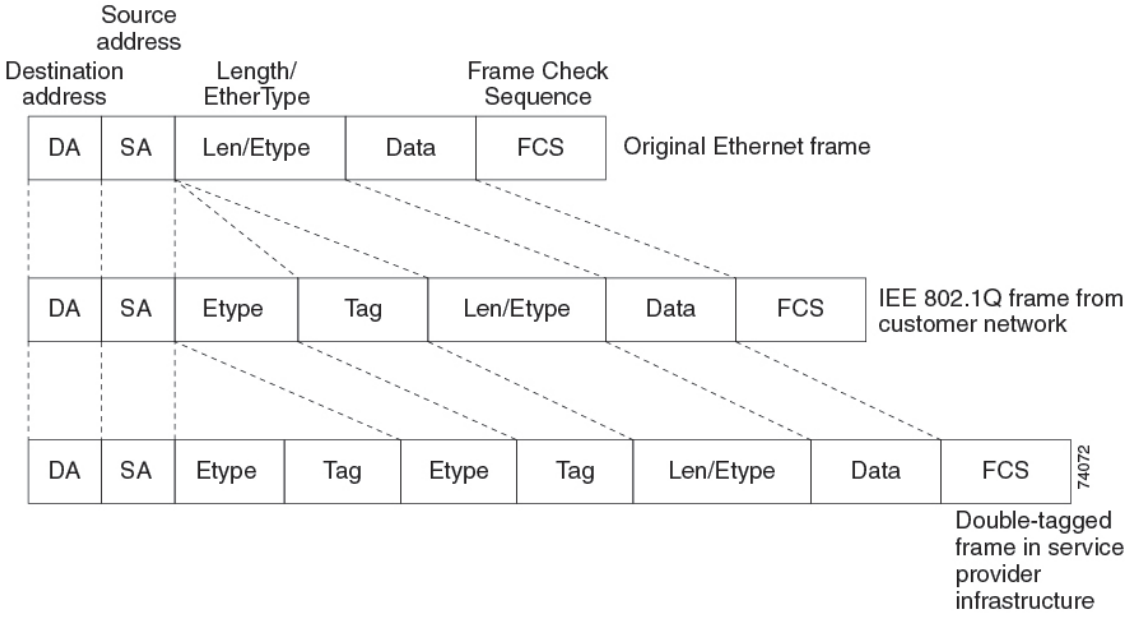


Packets coming from the customer trunk port into the tunnel port on the service-provider edge device are normally IEEE 802.1Q-tagged with the appropriate VLAN ID. The tagged packets remain intact inside the device and when they exit the trunk port into the service-provider network, they are encapsulated with another layer of an IEEE 802.1Q tag (called the metro tag) that contains the VLAN ID that is unique to the customer. The original customer IEEE 802.1Q tag is preserved in the encapsulated packet. Therefore, packets entering the service-provider network are double-tagged, with the outer (metro) tag containing the customer's access VLAN ID, and the inner VLAN ID being that of the incoming traffic.

When the double-tagged packet enters another trunk port in a service-provider core device, the outer tag is stripped as the device processes the packet. When the packet exits another trunk port on the same core device, the same metro tag is again added to the packet.

Figure 2: Original (Normal), IEEE 802.1Q, and Double-Tagged Ethernet Packet Formats

This figure shows the tag structures of the double-tagged packets.



When the packet enters the trunk port of the service-provider egress device, the outer tag is again stripped as the device internally processes the packet. However, the metro tag is not added when the packet is sent out the tunnel port on the edge device into the customer network. The packet is sent as a normal IEEE 802.1Q-tagged frame to preserve the original VLAN numbers in the customer network.

In the above network figure, Customer A was assigned VLAN 30, and Customer B was assigned VLAN 40. Packets entering the edge device tunnel ports with IEEE 802.1Q tags are double-tagged when they enter the service-provider network, with the outer tag containing VLAN ID 30 or 40, appropriately, and the inner tag containing the original VLAN number, for example, VLAN 100. Even if both Customers A and B have VLAN 100 in their networks, the traffic remains segregated within the service-provider network because the outer tag is different. Each customer controls its own VLAN numbering space, which is independent of the VLAN numbering space that is used by other customers and the VLAN numbering space that is used by the service-provider network.

At the outbound tunnel port, the original VLAN numbers on the customer's network are recovered. It is possible to have multiple levels of tunneling and tagging, but the device supports only one level in this release.

If traffic coming from a customer network is not tagged (native VLAN frames), these packets are bridged or routed as normal packets. All packets entering the service-provider network through a tunnel port on an edge device are treated as untagged packets, whether they are untagged or already tagged with IEEE 802.1Q headers. The packets are encapsulated with the metro tag VLAN ID (set to the access VLAN of the tunnel port) when they are sent through the service-provider network on an IEEE 802.1Q trunk port. The priority field on the metro tag is set to the interface class of service (CoS) priority configured on the tunnel port. (The default is zero if none is configured.)

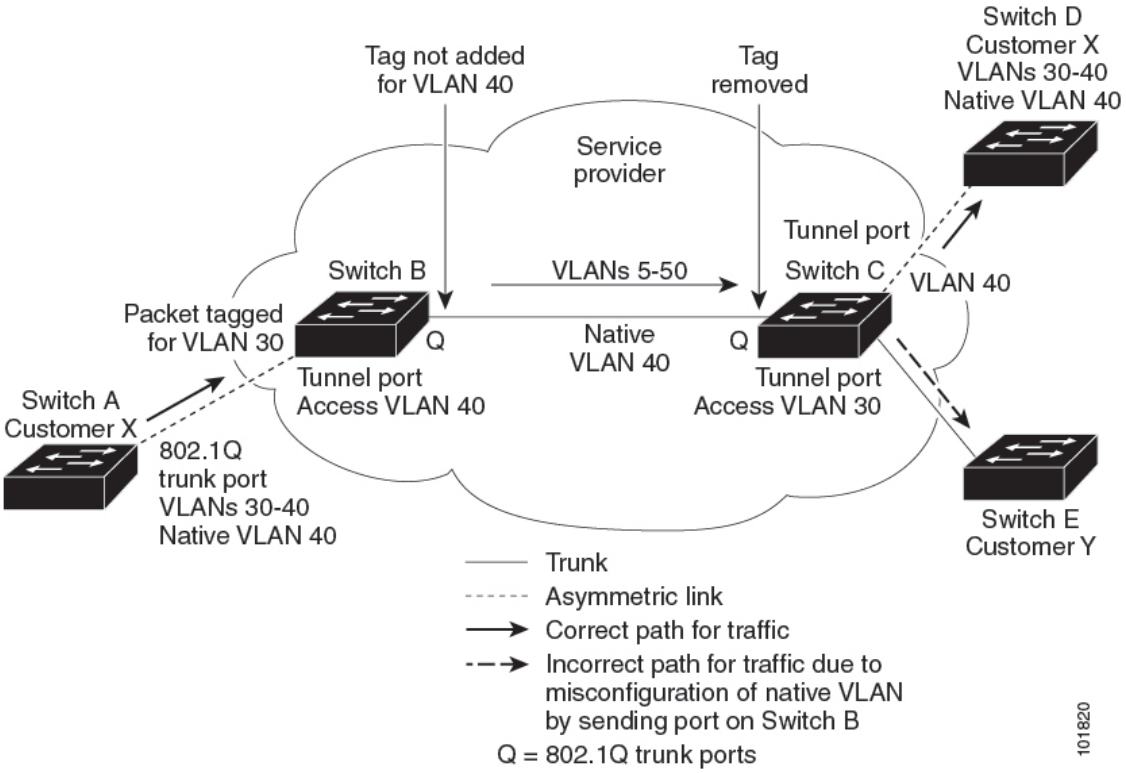
On switches, because 802.1Q tunneling is configured on a per-port basis, it does not matter whether the switch is a standalone device or a member switch. All configuration is done on the active switch.

Native VLANs

When configuring IEEE 802.1Q tunneling on an edge device, you must use IEEE 802.1Q trunk ports for sending packets into the service-provider network. However, packets going through the core of the service-provider network can be carried through IEEE 802.1Q trunks, ISL trunks, or nontrunking links. When IEEE 802.1Q trunks are used in these core devices, the native VLANs of the IEEE 802.1Q trunks must not match any native VLAN of the nontrunking (tunneling) port on the same device because traffic on the native VLAN would not be tagged on the IEEE 802.1Q sending trunk port.

In the following network figure, VLAN 40 is configured as the native VLAN for the IEEE 802.1Q trunk port from Customer X at the ingress edge switch in the service-provider network (Switch B). Switch A of Customer X sends a tagged packet on VLAN 30 to the ingress tunnel port of Switch B in the service-provider network, which belongs to access VLAN 40. Because the access VLAN of the tunnel port (VLAN 40) is the same as the native VLAN of the edge switch trunk port (VLAN 40), the metro tag is not added to tagged packets received from the tunnel port. The packet carries only the VLAN 30 tag through the service-provider network to the trunk port of the egress-edge switch (Switch C) and is misdirected through the egress switch tunnel port to Customer Y.

Figure 3: Potential Problems with IEEE 802.1Q Tunneling and Native VLANs



These are some ways to solve this problem:

- Use the **switchport trunk native vlan tag** per-port command and the **vlan dot1q tag native** global configuration command to configure the edge switches so that all packets going out an IEEE 802.1Q trunk, including the native VLAN, are tagged. If the switch is configured to tag native VLAN packets on all IEEE 802.1Q trunks, the switch drops untagged packets, and sends and receives only tagged packets.

Note **vlan dot1q tag native** global command needs to be enabled to execute the **switchport trunk native vlan tag** command.

- Ensure that the native VLAN ID on the edge switches trunk port is not within the customer VLAN range. For example, if the trunk port carries traffic of VLANs 100 to 200, assign the native VLAN a number outside that range.

System MTU

The default system MTU for traffic on the device is 1500 bytes.

You can configure 10-Gigabit and Gigabit Ethernet ports to support frames larger than 1500 bytes by using the **system mtu** *bytes* global configuration command.

The system MTU and system jumbo MTU values do not include the IEEE 802.1Q header. Because the IEEE 802.1Q tunneling feature increases the frame size by 4 bytes when the metro tag is added, you must configure all devices in the service-provider network to be able to process maximum frames by adding 4 bytes to the system MTU size.

For example, the device supports a maximum frame size of 1496 bytes with this configuration: The device has a system MTU value of 1500 bytes, and the **switchport mode dot1q tunnel** interface configuration command is configured on a 10-Gigabit or Gigabit Ethernet device port.

IEEE 802.1Q Tunneling and Other Features

Although IEEE 802.1Q tunneling works well for Layer 2 packet switching, there are incompatibilities between some Layer 2 features and Layer 3 switching.

- A tunnel port cannot be a routed port.
- IP routing is not supported on a VLAN that includes IEEE 802.1Q tunnel ports. Packets that are received from a tunnel port are forwarded based only on Layer 2 information. If routing is enabled on a switch virtual interface (SVI) that includes tunnel ports, untagged IP packets received from the tunnel port are recognized and routed by the switch. Customers can access the Internet through its native VLAN. If this access is not needed, you should not configure SVIs on VLANs that include tunnel ports.
- Fallback bridging is not supported on tunnel ports. Because all IEEE 802.1Q-tagged packets that are received from a tunnel port are treated as non-IP packets, if fallback bridging is enabled on VLANs that have tunnel ports that are configured, IP packets would be improperly bridged across VLANs. Therefore, you must not enable fallback bridging on VLANs with tunnel ports.
- Tunnel ports do not support IP access control lists (ACLs).
- Layer 3 quality of service (QoS) ACLs and other QoS features related to Layer 3 information are not supported on tunnel ports. MAC-based QoS is supported on tunnel ports.
- EtherChannel port groups are compatible with tunnel ports as long as the IEEE 802.1Q configuration is consistent within an EtherChannel port group.
- Port Aggregation Protocol (PAgP), Link Aggregation Control Protocol (LACP), and UniDirectional Link Detection (UDLD) are supported on IEEE 802.1Q tunnel ports.

- Dynamic Trunking Protocol (DTP) is not compatible with IEEE 802.1Q tunneling because you must manually configure asymmetric links with tunnel ports and trunk ports.
- VLAN Trunking Protocol (VTP) does not work between devices that are connected by an asymmetrical link or devices that communicate through a tunnel.
- Loopback detection is supported on IEEE 802.1Q tunnel ports.
- When a port is configured as an IEEE 802.1Q tunnel port, spanning-tree bridge protocol data unit (BPDU) filtering is automatically enabled on the interface. Cisco Discovery Protocol (CDP) is automatically disabled on the interface.

Note When you are configuring IEEE 802.1Q tunneling, the BPDU filtering configuration information is not displayed as spanning-tree BPDU filter is automatically enabled. You can verify the BPDU filter information using the **show spanning tree interface** command.

- When an IEEE 802.1Q tunnel port is configured as SPAN source, span filter must be applied for SVLAN to avoid packet loss.
- IGMP/MLD packet forwarding can be enabled on IEEE 802.1Q tunnels. This can be done by disabling IGMP/MLD snooping on the service provider network.

Default IEEE 802.1Q Tunneling Configuration

By default, IEEE 802.1Q tunneling is disabled because the default switchport mode is dynamic auto. Tagging of IEEE 802.1Q native VLAN packets on all IEEE 802.1Q trunk ports is also disabled.

Information About Custom EtherType for Cisco Catalyst 9600 Series Supervisor 2 Module

Cisco Catalyst 9600 Series Supervisor 2 Module uses the default ether type 0x8100 for dot1q and Q-in-Q encapsulations.

Custom EtherType allows configuration of the EtherType per port. The EtherTypes 0x9100 and 0x88a8 are supported using the custom EtherType feature by configuring the **switchport dot1q ether type** command on a Layer 2 interface. 802.1q (0x8100) EtherType is the default EtherType.

With the custom dot1q EtherType, you can select a nonstandard (0x9100 and 0x88a8) 2-byte EtherType in order to identify 802.1Q-tagged frames. The switch is allowed to interoperate with third-party vendors' switches that do not use the standard 0x8100 EtherType to identify 802.1Q-tagged frames. For instance, if 0x9100 EtherType is used as the custom dot1q EtherType on a particular port, incoming frames containing the EtherType 0x9100 are forwarded to the VLAN contained in the tag. Frames that arrive with the default EtherType 0x8100 are treated as untagged.

The interface can be configured with the following EtherTypes:

- 0x9100
- 0x88a8

When configuring a custom EtherType field value, consider the following:

- To use a custom EtherType field value, all the network devices in the traffic path across the network must support the custom EtherType field value.
- You can configure a custom EtherType field value on trunk ports and tunnel ports.
- You can configure a custom EtherType field value on an EtherChannel port.

Restrictions for Custom Ethertypes

- If a custom ethertype 0x9100 or 0x88a8 is configured on a port, all the custom ethertypes are accepted by the RX filter. However, only the configured ethertype will be used during TX. During egress, the configured ethertype 0x9100 or 0x88a8 is inserted and packets received with 8100 ethertypes are treated as untagged (current limitation of untagged packet on trunk port is observed).
- If a custom ethertype is configured under a physical port, all the tagged service instances under the physical port are forced to use that particular ethertype.
- Custom ethertype is not supported on IP configured or routed interfaces.
- Custom ethertype dynamic update from dot1q to tunneling, or from tunneling to dot1q is not supported.
- dot1q tunneling ethertype Canonical Frame Identifier (CFI) preservation is not supported.
- Connectivity Fault Management (CFM) with custom ethertype is not supported.
- 802.1ad is not supported for custom ethertype.
- DHCP snooping is not supported for custom ethertype.
- Each port supports only one EtherType field value. A port that is configured with a custom EtherType field value does not recognize frames that have any other EtherType field value as tagged frames. For example, a trunk port that is configured with a custom EtherType field value does not recognize the standard 0x8100 EtherType field value on 802.1Q-tagged frames and cannot put the frames into the VLAN to which they belong.

How to Configure IEEE 802.1Q Tunneling

Follow these steps to configure a port as an IEEE 802.1Q tunnel port:

Before you begin

- Always use an asymmetrical link between the customer device and the edge device, with the customer device port configured as an IEEE 802.1Q trunk port and the edge device port configured as a tunnel port.
- Assign tunnel ports only to VLANs that are used for tunneling.
- Observe configuration requirements for native VLANs and for and maximum transmission units (MTUs).

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet2/0/1`	Enters interface configuration mode for the interface to be configured as a tunnel port. This should be the edge port in the service-provider network that connects to the customer device. Valid interfaces include physical interfaces and port-channel logical interfaces (port channels 1 to 48).
Step 4	**switchport access vlan** *vlan-id* **Example:** `Device(config-if)# switchport access vlan 2`	Specifies the default VLAN, which is used if the interface stops trunking. This VLAN ID is specific to the particular customer.
Step 5	**switchport mode dot1q-tunnel** **Example:** `Device(config-if)# switchport mode dot1q-tunnel`	Sets the interface as an IEEE 802.1Q tunnel port. **Note** Use the **no switchport mode dot1q-tunnel** interface configuration command to return the port to the default state of dynamic desirable.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 7	**vlan dot1q tag native** **Example:** `Device(config)# vlan dot1q tag native`	(Optional) Sets the device to enable tagging of native VLAN packets on all IEEE 802.1Q trunk ports. When not set, and a customer VLAN ID is the same as the native VLAN, the trunk port does not apply a metro tag, and packets could be sent to the wrong destination. **Note** Use the**no vlan dot1q tag native** global configuration command to disable tagging of native VLAN packets.

	Command or Action	Purpose
Step 8	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Step 9	Use one of the following:
	• **show dot1q-tunnel**
	• **show running-config interface**
	Example:
	`Device# show dot1q-tunnel`
	or
	`Device# show running-config interface`
		Displays the ports that are configured for IEEE 802.1Q tunneling.
		Displays the ports that are in tunnel mode.

Step 10	**show vlan dot1q tag native** **Example:** `Device# show vlan dot1q native`	Displays IEEE 802.1Q native VLAN tagging status.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

How to Configure Custom EtherTypes

Custom EtherType allows configuration of the EtherType per port. Use the following procedure to configure custom EtherTypes on Cisco Catalyst 9600 Series Supervisor 2 Module.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode, enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface name* **Example:**	Enters interface configuration mode.

	Command or Action	Purpose
	`Device(config)# interface FiftyGigE1/1/0/1`	
Step 4	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Configures the interface to operate in trunk mode.
Step 5	**switchport trunk allowed vlan all** **Example:** `Device(config-if)# switchport trunk allowed vlan all`	Specifies all the VLANs defined on the switch to be allowed on the interface.
Step 6	**switchport dot1q EtherType** *EtherType value* **Example:** `Device(config-if)# switchport dot1q ethertype 9100`	Configures a custom EtherType. Supported custom EtherTypes are 0x9100 and 0x88a8. The EtherType value should be in hex *<600-FFFF>*.
Step 7	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 8	**show platform software fed switch active ifm mappings** **Example:** `Device(config-if)# show platform software fed switch active ifm mappings`	Displays mapping information about the switch.
Step 9	**show platform software fed switch active ifm if-id** *IF-ID* **Example:** `Device(config-if)# show platform software fed switch active ifm if-id 0x4b6`	Displays mapping information about the specified interface.

Monitoring Tunneling Status

The following table describes the commands used to monitor tunneling status.

Table 1: Commands for Monitoring Tunneling

Command	Purpose
show dot1q-tunnel	Displays IEEE 802.1Q tunnel ports on the device.
show dot1q-tunnel interface *interface-id*	Verifies if a specific interface is a tunnel port.
show vlan dot1q tag native	Displays the status of native VLAN tagging on the device.

Example: Configuring an IEEE 802.1Q Tunneling Port

The following example shows how to configure an interface as a tunnel port, enable tagging of native VLAN packets, and verify the configuration. In this configuration, the VLAN ID for the customer connected to Gigabit Ethernet interface 7 on stack member 1 is VLAN 22.

```
Device(config)# interface gigabitethernet1/0/7
Device(config-if)# switchport access vlan 22
% Access VLAN does not exist. Creating vlan 22
Device(config-if)# switchport mode dot1q-tunnel
Device(config-if)# exit
Device(config)# vlan dot1q tag native
Device(config)# end
Device# show dot1q-tunnel interface gigabitethernet1/0/7
Port
-----
Gi1/0/1Port
-----
Device# show vlan dot1q tag native
dot1q native vlan tagging is enabled
```

Example: Configuring a Custom Ethertype

The following example shows how to configure a custom ethertype under an interface.

```
Device> enable
Device# configure terminal
Device(config)# interface Hu2/1/0/1
Device(config-if)# switchport mode trunk
Device(config-if)# switchport  trunk allowed vlan all
Device(config-if)# switchport dot1q ethertype 88a8
Device(config-if)# end
Device# show platform software fed switch active ifm mappings

Interface                IF_ID  Inst Asic Core IFG_ID Port SubPort Mac  First Serdes Last
 Serdes  Cntx LPN  GPN   Type Active
HundredGigE2/1/0/1       0x3      0   0    0      0    0      0     129  0            1
          0    1    1153 NIF    Y
HundredGigE2/1/0/2       0x4b4    0   0    0      0    0      0     130  2            3
          0    2    1154 NIF    Y
HundredGigE2/1/0/3       0x4b5    0   0    0      0    0      0     131  4            5
          0    3    1155 NIF    Y
HundredGigE2/1/0/4       0x4b6    0   0    0      0    0      0     132  6            7
          0    4    1156 NIF    Y
```

```
HundredGigE2/1/0/5        0x4b7    0   0    0      1    0      0     133  8            9
          0    5    1157 NIF    Y
HundredGigE2/1/0/6        0x4b8    0   0    0      1    0      0     134  10           11
          0    6    1158 NIF    Y
HundredGigE2/1/0/7        0x4b9    0   0    0      1    0      0     135  12           13
          0    7    1159 NIF    Y
HundredGigE2/1/0/8        0x4ba    0   0    0      1    0      0     136  14           15
          0    8    1160 NIF    Y
HundredGigE2/1/0/9        0x4bb    0   0    1      0    0      0     137  16           17
          0    9    1161 NIF    Y
HundredGigE2/1/0/10       0x4bc    0   0    1      0    0      0     138  18           19
          0    10   1162 NIF    Y
HundredGigE2/1/0/11       0x4bd    0   0    1      0    0      0     139  20           21
          0    11   1163 NIF    Y

Device# show platform software fed switch active ifm if-id 0x4b6

Interface IF_ID         : 0x00000000000004b6
Interface Name          : HundredGigE2/1/0/4
Interface Block Pointer : 0x7ba00bbb5328
Interface Block State   : Ready
Interface State         : Enabled
Interface Admin mode    : Admin Up
Interface Status        : NPD
Interface Ref-Cnt       : 1
Interface Type          : ETHER
 Port Type         : SWITCH PORT
 Port Location     : LOCAL
 Slot              : 15
 Unit              : 0
 Slot Unit         : 4
 SNMP IF Index     : 183
 GPN               : 1156
 EC Channel        : 0
 EC Index          : 0
 QoS Trust Type    : 3 (DSCP)
Ref Count : 1 (feature Ref Counts + 1)
No Feature Reference count Present
IFM Feature Sub block information
 Port Physical Subblock
               Affinity .......... [local]
  LPN ................ [4]
  GPN ................ [1156]
  Speed .............. [40GB]
  type ............... [IFM_PORT_TYPE_L2]
  MTU ................ [9222]
  ac profile ........ [IFM_AC_PROFILE_DEFAULT_CUST_ETHER]
 Port Subblock [0]
  Mac port oid..................... [2426]
  System port oid.................. [2430]
  System port gid.................. [1359]
  Ethernet port oid................ [2441]
  Voq oid.......................... [2428]
 Platform Subblock
  Asic............... [0]
  Core............... [0]
  Asic Port......... [0]
  Asic Sub Port..... [65535]
  Ifg Id............. [0]
  Mac Num............ [132]
  First Serdes...... [6]
  Last Serdes....... [7]
  FC Mode........... [0]
               FEC Mode.......... [0]
```

```
  Context Id........ [0]
 Port L2 Subblock
  L2 Port Mode ................ [port_mode_trunk]
  L2 Port Mode set............. [Yes]
  Default vlan ................ [0]
  Ethertype.................... [88a8]
  untagged port bd vlan ....... [0]
  status....................... [0]
  ac profile .................. [IFM_AC_PROFILE_DEFAULT_CUST_ETHER]
Events Log
[2023/02/23 14:36:45.121] XCVR enable
[2023/02/23 14:36:45.122] Port mode enable
[2023/02/23 14:36:49.710] link state up
[2023/02/23 14:36:49.712] Mode Dynamic
[2023/02/23 14:36:52.720] Mode Trunk
[2023/02/23 14:36:52.725] Mode Trunk
[2023/02/23 14:44:55.090] STATE DISABLE link down
[2023/02/23 14:44:55.129] link state down
[2023/02/23 14:44:55.132] XCVR disable
[2023/02/23 14:44:55.132] Port mode disable
[2023/02/23 14:44:55.132] link state down
[2023/02/23 14:45:03.992] XCVR enable
 --More--

Device# show interface Hu2/1/0/1 switchport
Name: Hu2/1/0/4
Switchport: Enabled
Administrative Mode: tunnel
Operational Mode: tunnel
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: native
Administrative Dot1q Ethertype: 0x9100
Operational Dot1q Ethertype: 0x9100
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan: none
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Vepa Enabled: false
App Interface: false
Appliance trust: none

Device#
```

Feature History for IEEE 802.1Q Tunneling

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	IEEE 802.1Q Tunneling	The IEEE 802.1Q tunneling feature is designed for service providers who carry traffic of multiple customers across their networks and are required to maintain the VLAN and Layer 2 protocol configurations of each customer without impacting the traffic of other customers.
Cisco IOS XE Dublin 17.11.1	Custom Ethertype	This feature was introduced on Cisco Catalyst 9600X-SUP-2 module .

Use Cisco Feature Navigator to find information about platform and software image support.