

Configure QinQ

- Feature History for QinQ, on page 1
- Information About IEEE 802.1Q Tunneling, on page 1
- Configure Custom EtherTypes, on page 5
- Example to configure an IEEE 802.1Q tunneling port, on page 6
- Example to configure a custom ethertype on an interface, on page 7

Feature History for QinQ

This table provides release and platform support information for the features explained in this module.

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

Table 1:

Release	Feature Name and Description	Supported Platform
Cisco IOS XE 17.18.1	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 C9350 Series Smart Switches

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

IEEE 802.1Q tunnel ports are a type of network port used in service-provider networks that:

- preserve customer VLAN IDs through a VLAN-in-VLAN hierarchy,
- segregate traffic for multiple customers within the same service-provider network, and
- expand VLAN space by retagging customer-tagged packets with a metro tag.

Metro tag:

An outer IEEE 802.1Q tag added to encapsulate packets entering the service-provider network, uniquely identifying customer traffic.

Tunnel port:

A port configured to use IEEE 802.1Q tunneling, connecting customer devices to the service-provider edge device.

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 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.

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.

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 used by other customers and the VLAN numbering space 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.

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.

IEEE 802.1Q tunneling features and compatibility

IEEE 802.1Q tunneling is utilized for Layer 2 packet switching but certain incompatibilities exist with Layer 3 features. Below is a structured reference for IEEE 802.1Q tunneling:

Incompatibility Issues:

- Routing Support: Tunnel ports cannot be routed. IP routing is unsupported on VLANs with tunnel ports.
- Fallback Bridging: Unsupported on tunnel ports, treating IP packets improperly.
- IP ACLs & Layer 3 QoS: Not applicable on tunnel ports; MAC-based QoS is supported.

Compatibility with Other Protocols:

- EtherChannel Port Groups: Compatible if IEEE 802.1Q is consistent across groups.
- PAgP, LACP, UDLD: Supported on tunnel ports.
- Dynamic Trunking Protocol (DTP): Not compatible; requires manual configuration of links.
- VLAN Trunking Protocol (VTP): Ineffective across asymmetrical links and tunnels.

Additional Features and Protocol Support:

- BPDU Filtering: Enabled automatically on tunnel ports.
- Cisco Discovery Protocol (CDP): Disabled automatically.
- Loopback Detection: Supported feature.
- IGMP/MLD Forwarding: Can be enabled by disabling snooping on the provider network.
- SPAN Configuration: Span filter application needed when tunnel ports are SPAN sources.
- Routing Support: Tunnel ports cannot be routed; IP routing is unsupported on VLANs with tunnel ports.
- Fallback Bridging: Unsupported on tunnel ports, treating IP packets improperly.
- IP ACLs & Layer 3 QoS: Not applicable on tunnel ports; MAC-based QoS is supported.

Compatibility with Other Protocols:

- EtherChannel Port Groups: Compatible if IEEE 802.1Q is consistent across groups.
- PAgP, LACP, UDLD: Supported on tunnel ports.
- Dynamic Trunking Protocol (DTP): Not compatible; requires manual configuration of links.
- VLAN Trunking Protocol (VTP): Ineffective across asymmetrical links and tunnels.

Additional Features and Protocol Support:

- BPDU Filtering: Enabled automatically on tunnel ports.

- Cisco Discovery Protocol (CDP): Disabled automatically.
- Loopback Detection: Supported feature.
- IGMP/MLD Forwarding: Can be enabled by disabling snooping on the provider network.
- SPAN Configuration: Span filter application needed when tunnel ports are SPAN sources.

Configure Custom EtherTypes

Follow these steps to configure a custom EtherTypes on Cisco Catalyst Switches:

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	interfaceinterface name **Example:** `Device(config)# interface FiftyGigE1/1/0/1`	Enters interface configuration mode.
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 EtherTypeEtherType 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.

	Command or Action	Purpose
Step 8	show platform software fed switch active ifm if-idIF-ID **Example:** `Device(config-if)# show platform software fed switch active ifm interface_name HundredGig2/1/0/4`	Displays mapping information about the specified interface.

The switch interface is configured with the custom EtherType.

What to do next

Verify that the EtherType configuration works as expected by observing the network traffic and connectivity.

Commands for monitoring tunneling status

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

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 to configure an IEEE 802.1Q tunneling port

Configure an interface as a tunnel port with IEEE 802.1Q tunneling.

This procedure demonstrates configuring a tunneling port and verifying the setup using CLI commands.

Follow these steps to configure the IEEE 802.1Q tunneling port:

```
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
```

Result:

The tunneling port is configured, and native VLAN tagging is enabled, as verified by the show commands.

Example to configure a custom ethertype on an interface

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 interface_name HundredGig2/1/0/4
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#
```