Configuring 802.1Q VLAN Interfaces

This module describes the configuration and management of 802.1Q VLAN interfaces.

The IEEE 802.1Q specification establishes a standard method for tagging Ethernet frames with VLAN membership information. It defines the operation of VLAN bridges that permit the definition, operation, and administration of VLAN topologies within a bridged LAN infrastructure.

The 802.1Q standard is intended to address the problem of how to divide large networks into smaller parts so broadcast and multicast traffic does not use more bandwidth than necessary. The standard also helps provide a higher level of security between segments of internal networks.

Feature History for Configuring 802.1Q VLAN Interfaces

Release	Modification
Release 7.0.11	This feature was introduced.

Prerequisites for Configuring 802.1Q VLAN Interfaces

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Before configuring 802.1Q VLAN interfaces, ensure that you meet the following conditions:

- You must have configured a HundredGigE interface, a FourHundredGigE interface, or an Ethernet bundle interface.

Information About Configuring 802.1Q VLAN Interfaces

To configure 802.1Q VLAN interfaces, you must understand the following concepts:

802.1Q VLAN Overview

A VLAN is a group of devices on one or more LANs that you can configure so that the devices can communicate as if they were attached to the same wire. When in fact, they are located on several different LAN segments. Because VLANs are based on logical instead of physical connections, they are flexible for user and host management, bandwidth allocation, and resource optimization.

The IEEE 802.1Q protocol standard addresses the problem of dividing large networks into smaller parts so broadcast and multicast traffic does not consume more bandwidth than necessary. The standard also helps 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 subinterface configuration on 40Gigabit, HundredGig, FourHundredGig, and bundle interfaces.

802.1Q Tagged Frames

The IEEE 802.1Q tag-based VLAN uses an extra tag in the MAC header to identify the VLAN membership of a frame across bridges. This tag is used for VLAN and quality of service (QoS) priority identification. The VLANs can be created statically by manual entry or dynamically through Generic Attribute Registration Protocol (GARP) VLAN Registration Protocol (GVRP). The VLAN ID associates a frame with a specific VLAN and provides the information that switches must process the frame across the network. A tagged frame is four bytes longer than an untagged frame and contains two bytes of Tag Protocol Identifier (TPID) residing within the type and length field of the Ethernet frame and two bytes of Tag Control Information (TCI) which starts after the source address field of the Ethernet frame.

Subinterfaces

Subinterfaces are logical interfaces that you can create on a hardware interface. These software-defined interfaces allow the segregation of traffic into separate logical channels on a single hardware interface. It also allows for the better utilization of the available bandwidth on the physical interface.

You can distinguish subinterfaces from each other by adding an extension at the end of the interface name and designation. For instance, the system indicates Ethernet subinterface 23 on the physical interface designated TenGigE 0/1/0/0, by TenGigE 0/1/0/0.23.

Before the system allows a subinterface 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, you must explicitly define the VLAN identifier.

Subinterface MTU

The system inherits the subinterface maximum transmission unit (MTU)from the physical interface with an additional four bytes allowed for the 802.1Q VLAN tag.

Native VLAN

The router does not support a native VLAN. However, the equivalent functionality is accomplished using an **encapsulation** command as follows:

```
encapsulation dot1q TAG-ID
```

How to Configure 802.1Q VLAN Interfaces

This section contains the following procedures:

Configuring 802.1Q VLAN Subinterfaces

This task explains how to configure 802.1Q VLAN subinterfaces. To remove these subinterfaces, see the "Removing an 802.1Q VLAN Subinterface" section.

Tip You can programmatically configure and retrieve the VLAN interfaces and subinterfaces parameters using `openconfig-vlan.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

```
RP/0/RP0/CPU0:router# configure
```

/* Enter subinterface configuration mode and specifies the interface type, location, and subinterface number. */

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/2/0/4.10
```

- Replace the *interface-path-id* argument with one of the following instances:
- Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack*/*slot*/*module*/*port*, and a slash between values is required as part of the notation.
- Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.
- Naming notation is *interface-path-id.subinterface*, and a period between arguments is required as part of the notation.

/* Set the Layer 2 encapsulation of an interface. */

```
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 100
```

Note
- The **dot1q vlan** command is replaced by the **encapsulation dot1q** command on the Cisco 8000 Series Router. It is still available for backward-compatibility, but only for Layer 3 interfaces.

/* Assign an IP address and subnet mask to the subinterface. */

```
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 178.18.169.23/24
```

- Replace *ip-address* with the primary IPv4 address for an interface.

- Replace *mask* with the mask for the associated IP subnet. The network mask can be specified in either of two ways:
- The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address.
- The network mask can be indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are network address.

/* The **exit** command is not explicitly required. */

```
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# end
or
RP/0/RP0/CPU0:router(config)# commit
```

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?
  [cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Example

```
"RP/0/RP0/CPU0:S3(config)#interface fourHundredGigE 0/5/0/1.100
RP/0/RP0/CPU0:S3(config-subif)#ipv4 address 100.100.100.100/31
RP/0/RP0/CPU0:S3(config-subif)#encapsulation dot1q 100
RP/0/RP0/CPU0:S3(config-subif)#no shutdown
RP/0/RP0/CPU0:S3(config-subif)#commit
Mon Jul  8 23:05:01.979 PDT
RP/0/RP0/CPU0:S3(config-subif)#end
RP/0/RP0/CPU0:S3#show interfaces fourHundredGigE 0/5/0/1.100 brief
Mon Jul  8 23:05:08.784 PDT

               Intf       Intf        LineP              Encap  MTU        BW
               Name       State       State               Type (byte)    (Kbps)
--------------------------------------------------------------------------------
      FH0/5/0/1.100        up        up             802.1Q  1518  400000000

RP/0/RP0/CPU0:S3#show interfaces brief location 0/5/CPU0 | include 802.1Q
Mon Jul  8 23:07:43.929 PDT
      FH0/5/0/1.100        up        up             802.1Q  1518  400000000
RP/0/RP0/CPU0:S3#

RP/0/RP0/CPU0:S3#"
```

Removing an 802.1Q VLAN Subinterface

This task explains how to remove 802.1Q VLAN subinterfaces that have been previously configured using the Configuring 802.1Q VLAN subinterfaces section in this module.

```
RP/0/RP0/CPU0:router# configure
```

/* Remove the subinterface, which also automatically deletes all the configuration applied to the subinterface. */

```
RP/0/RP0/CPU0:router(config)# no interface TenGigE 0/2/0/4.10
```

- Replace the *instance* argument with one of the following instances:
- Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack*/*slot*/*module*/*port*, and a slash between values is required as part of the notation.
- Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.

Naming notation is *instance.subinterface*, and a period between arguments is required as part of the notation.

Note Repeat to remove other VLAN subinterfaces.

```
RP/0/RP0/CPU0:router(config)# end
or
RP/0/RP0/CPU0:router(config)# commit
```

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?
  [cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for VLAN Interfaces

This section contains the following example:

VLAN Subinterfaces: Example

The following example shows how to create three VLAN subinterfaces at one time:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/2/0/4.1

RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 10.0.10.1/24
RP/0/RP0/CPU0:router(config-subif)# interface TenGigE0/2/0/4.2

RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 101
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 10.0.20.1/24
RP/0/RP0/CPU0:router(config-subif)# interface TenGigE0/2/0/4.3

RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 102
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 10.0.30.1/24
RP/0/RP0/CPU0:router(config-subif)# commit
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# exit

RP/0/RP0/CPU0:router# show ethernet trunk bundle-Ether 1
Trunk                              Sub types        Sub states
VLAN trunks: 1,
  1 are 802.1Q (Ether)
Sub-interfaces: 3,
  3 are up.
802.1Q VLANs: 3,
  3 have VLAN Ids,

RP/0/RP0/CPU0:router# show vlan interfaceInterface     St Ly    MTU    Subs    L3
   Up    Down  Ad-Down
Te0/2/0/4.1         802.1Q             10  up
Te0/2/0/4.2         802.1Q             20  up
Te0/2/0/4.3         802.1Q             30  up
RP/0/RP0/CPU0:router# show vlan trunks briefBE1           Up L3   1514    1000
  0    1000     1000       0       0

Summary                       1000        0    1000     1000       0       0

Te0/2/0/4           802.1Q (Ether)        up
```

The following example shows how to create two VLAN subinterfaces on an Ethernet bundle:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface bundle-ether 2
RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.2.1/24
RP/0/RP0/CPU0:router(config-if)# exit
RP/0/RP0/CPU0:router(config)# interface bundle-ether 2.1

RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 192.168.100.1/24
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# interface bundle-ether 2.2
```

```
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 200
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 192.168.200.1/24
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# commit
```