Configuring 802.1Q VLAN Interfaces

A 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. VLANs are very flexible for user and host management, bandwidth allocation, and resource optimization because they are based on logical grouping instead of physical connections.

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.

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 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.

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.

SUMMARY STEPS

1. **configure**
2. **interface {TenGigE | FortyGigE | HundredGigE | Bundle-Ether}** *interface-path-id.subinterface*
3. **encapsulation dot1q**
4. **ipv4 address** *ip-address mask*

5. **exit**
6. Repeat Step 2 through Step 5 to define the rest of the VLAN subinterfaces.
7. **end** or **commit**
8. **show ethernet trunk bundle-ether** *instance*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **interface {TenGigE | FortyGigE | HundredGigE | Bundle-Ether}** *interface-path-id.subinterface*

Example:

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

Enters subinterface configuration mode and specifies the interface type, location, and subinterface number.

- 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 2147483647.
- Naming notation is *interface-path-id.subinterface*, and a period between arguments is required as part of the notation.

Step 3 **encapsulation dot1q**

Example:

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

Sets the Layer 2 encapsulation of an interface.

Step 4 **ipv4 address** *ip-address mask*

Example:

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

Assigns an IP address and subnet mask to the subinterface.

- 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.

Step 5 **exit**

Example:

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

(Optional) Exits the subinterface configuration mode.

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

Step 6 Repeat Step 2 through Step 5 to define the rest of the VLAN subinterfaces.

—

Step 7 **end** or **commit**

Example:

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

or

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

Saves configuration changes.

- 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.

Step 8 **show ethernet trunk bundle-ether** *instance*

Example:

```
RP/0/RP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Verification

This example shows how to verify the configuration of Ethernet interfaces :

```
# show ethernet trunk be 1020 Wed May 17 16:43:32.804 EDT

Trunk                                 Sub types         Sub states
Interface     St Ly    MTU    Subs       L2      L3       Up    Down  Ad-Down
BE1020       Up L3   9100       3        3       0        3       0        0

Summary                             3        3       0        3       0        0
```

Configuring an Attachment Circuit on a VLAN

Use the following procedure to configure an attachment circuit on a VLAN.

SUMMARY STEPS

1. **configure**
2. **interface [GigabitEthernet | TenGigE | Bundle-Ether | FortyGigE]** *interface-path*] *id.subinterface* **l2transport**
3. **encapsulation dot1q 100**
4. **end** or **commit**
5. **show interfaces** [**GigabitEthernet** |**FortyGigE**|**Bundle-Ether** | **TenGigE**] *interface-path-id.subinterface*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **interface [GigabitEthernet | TenGigE | Bundle-Ether | FortyGigE]** *interface-path*] *id.subinterface* **l2transport**

Example:

```
RP/0//CPU0:router(config)# interface TenGigE 0/0/0/1.1 l2transport
```

Enters subinterface configuration and specifies the interface type, location, and subinterface number.

- 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 *instance.subinterface*, and a period between arguments is required as part of the notation.
- You must include the **l2transport** keyword in the command string; otherwise, the configuration creates a Layer 3 subinterface rather that an AC.

Step 3 **encapsulation dot1q 100**

Example:

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

Sets the Layer 2 encapsulation of an interface.

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

Step 4 **end** or **commit**

Example:

```
RP/0//CPU0:router(config-if-l2)# end
```

or

```
RP/0//CPU0:router(config-if-l2)# commit
```

Saves configuration changes.

- 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.

Step 5 **show interfaces** [**GigabitEthernet** |**FortyGigE**|**Bundle-Ether** | **TenGigE**] *interface-path-id.subinterface*

Example:

```
RP/0//CPU0:router# show interfaces TenGigE 0/0/0/3.1
```

(Optional) Displays statistics for interfaces on the router.

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.

SUMMARY STEPS

1. **configure**
2. **no interface** {**TenGigE** | **FortyGigE** | **HundredGigE** | **Bundle-Ether**] *interface-path-id.subinterface*
3. Repeat Step 2 to remove other VLAN subinterfaces.
4. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **no interface** {**TenGigE** | **FortyGigE** | **HundredGigE** | **Bundle-Ether**] *interface-path-id.subinterface*

Example:

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

Removes the subinterface, which also automatically deletes all the configuration applied to the subinterface.

- 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 2147483647.

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

Step 3 Repeat Step 2 to remove other VLAN subinterfaces.

—

Step 4 **end** or **commit**

Example:

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

or

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

Saves configuration changes.

- 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.

Information About Configuring 802.1Q VLAN Interfaces

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

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 0/0/0/0 would be indicated by TenGigE 0/0/0/0.23.

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.

Subinterface MTU

The subinterface maximum transmission unit (MTU) is inherited from the physical interface with an additional four bytes allowed for the 802.1Q VLAN tag. By default subinterface inherits MTU of physical interface if the MTU is not configured. We can have maximum 3 different MTU for a subinterface per NPU. .

EFPs

An Ethernet Flow Point (EFP) is a Metro Ethernet Forum (MEF) term describing abstract router architecture. An EFP is implemented by an Layer 2 subinterface with a VLAN encapsulation. The term EFP is used synonymously with an VLAN tagged L2 subinterface. .

Layer 2 VPN on VLANs

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

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 subinterface, and then configures that VLAN in subinterface configuration mode. To create an AC, you need to include the **l2transport** keyword in the **interface** command string to specify that the interface is a Layer 2 interface.

VLAN ACs support these modes of L2VPN operation:

- Basic Dot1Q AC—The AC covers all frames that are received and sent with a specific VLAN tag.
- QinQ AC—The AC 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.

Each VLAN on a CE-to-PE link can be configured as a separate L2VPN connection (using either VC type 4 or VC type 5).

Layer 3 QinQ

Layer 3 QinQ is an extension of IEEE 802.1 QinQ VLAN tag stacking. This feature enables you to increase the number of VLAN tags in an interface and increments the number of sub-interfaces up to 4094. Hence, with dual tag, the number of VLANs can reach up to 4094*4094. With the L3 QinQ feature with dual tag, interfaces check for IP addresses along with MAC addresses.

This feature supports:

- 802.1Q standards like 0x8100, 0x9100, 0x9200 (used as outer tag ether-type) and 0x8100 (used as inner tag ether-type).
- L3 802.1ad VLAN sub-interfaces, with 0x88a8 as the outer S-tag ether-type.
- Co-existence of L2 and L3 single tagged and double tagged VLANs.
- QinQ and dot1ad over ethernet bundle sub-interfaces.
- Default VRF.

Note QinQ sub-interfaces support these IP features:

- QoS, with policy that matches outer VLAN (and COS) alone, and not both outer and inner VLANs together (2-level QoS/H-QoS support).

- ACL, Netflow, BFD, ARP.

- Routing protocols – static, BGP, OSFPv2.

- IPV4/IPV6 unicast/multicast.

Prerequisites:

1. Enable QinQ dual tag support on L3 sub-interfaces on the NSC 5500 and NCS 560 platforms.
2. Ensure the sub-interface scale is the same as what is supported per platform on single tag/802.1Q case. L3 QinQ feature is enabled on physical interfaces as well as on bundle interfaces.

Note Types of sub-interfaces:

Interface type	Outer tag	Inner tag
Dot1q sub-interface	0x8100	None
QinQ sub-interface	0x8100	0x8100
QinQ sub-interface	0x88a8	0x8100
QinQ sub-interface	0x9100	0x8100
QinQ sub-interface	0x9200	0x8100

Limitations:

MPLS is not supported.

Example:

```
Example 1:
interface TenGigE0/0/0/6.111
mtu 1400
ipv4 address 10.1.1.1 255.255.255.0
ipv6 address 10::1/64
encapsulation dot1q 100 second-dot1q 200
!

interface Bundle-Ether10.1
ipv4 address 10.1.2.1 255.255.255.0
ipv6 address 1002::1/64
encapsulation dot1ad 10 second-dot1q 20
!

Example 2:
Router(config)# interface gigabitethernet 1/0/0
Router(config-if)# dot1q tunneling ethertype 0x9100
Router(config-if)# interface gigabitethernet 1/0/0.1
Router(config-subif)# encapsulation dot1q 100 second-dot1q 200
Router(config-subif)# ipv4 address 172.16.1.2 255.255.255.0

Example 3:
interface GigabitEthernet0/7/0/2.100
 description ** Business Services over DOCSIS **
 encapsulation dot1q 100 second-dot1q 200-500
 ipv4 address 192.168.212.6 255.255.255.252

Example 4:
interface Bundle-Ether1.2
 description cliente: NUOVA JOLLY MARINE S.R.L. TD: null NUA: null TGU: 100213581081
 encapsulation dot1q 3200 second-dot1q 2
 ipv4 address 85.42.169.6 255.255.255.252
 service-policy input BIZDSLIP_HSIHYP_NOBP_96KBMG
```