

Configure Layer 3 VPNs

This chapter describes Layer 3 QinQ and the procedures to configure Layer 3 QinQ.

Table 1: Feature History

Feature Name	Release Information	Feature Description
Dual tag for L3VPN on dataplane for inBand management	Cisco IOS XR Release 6.5.31	The Layer 3 QinQ feature allows you to increase the number of VLAN tags in an interface and increment the number of subinterfaces up to 4094. Hence, with the dual tag, the number of VLANs can reach up to 4094*4094. You can enable this feature either on a physical interface or on a bundle interface. Commands modified: • encapsulation dot1q

- Layer 3 QinQ, on page 1
- QoS on Layer 3 VPN, on page 2
- Configure Layer 3 QinQ, on page 8
- Verify Layer 3 QinQ, on page 8
- VRF on Layer 3 VPN for management interface, on page 8

Layer 3 QinQ

The Layer 3 QinQ feature enables you to increase the number of VLAN tags in an interface and increment the number of subinterfaces up to 4094. Hence, with the dual tag, the number of VLANs can reach up to 4094*4094. You can enable this feature either on a physical interface or on a bundle interface. When you cofigure this feature with the dual tag, interfaces check for IP addresses along with MAC addresses. Layer 3 QinQ is an extension of IEEE 802.1 QinQ VLAN tag stacking.

A dot1q VLAN subinterface is a virtual interface that is associated with a VLAN ID on a routed physical interface or a bundle interface. Subinterfaces divide the parent interface into two or more virtual interfaces.

You can assign unique Layer 3 parameters on these virtual interfaces, such as IP addresses and dynamic routing protocols. The IP address for each subinterface must be in a different subnet from any other subinterface on the parent interface.

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 subinterfaces with 0x88a8 as the outer S-tag ether-type.
- Coexistence of Layer 2 and Layer 3 single tagged and double tagged VLANs.
- QinQ and dot1ad over Ethernet bundle subinterfaces.

The Layer 3 QinQ feature allows you to provision quality of service (QoS), access lists (ACLs), bidirectional forwarding detection (BFD), NetFlow, routing protocols, and IPv4 unicast and multicast.

Table 2: Types of Subinterfaces

Interface Type	Outer Tag	Inner Tag
Dot1q subinterface	0x8100	None
QinQ subinterface	0x8100	0x8100
QinQ subinterface	0x88a8	0x8100
QinQ subinterface	0x9100	0x8100
QinQ subinterface	0x9200	0x8100

QoS on Layer 3 VPN

Table 3: Feature History

Feature Name	Release Information	Feature Description
QoS on Layer 3 VPN.	Cisco IOS XR Release 6.5.33	The L3VPN QoS support on NCS4000 brings the Uniform and Pipe tunneling modes for DSCP/MPLS experimental bits, while the packet travels from one customer edge (CE) router to another across the MPLS core. The tunneling modes allow the customers to set the priority of the IP packets for the MPLS and the core networks.

QoS enables tunneling to be transparent from one edge of a network to the other edge of the network. A tunnel starts where there is label imposition and ends where the label is disposed off. Label disposition is where the

label is removed from the stack. The removed packet goes out as an MPLS packet with a different per-hop behavior (PHB) layer underneath or as an IP packet with the IP PHB layer.

Packets are forwarded in the following three ways through a network with respect to QoS.

- **Uniform Tunneling Mode** In the Uniform Tunneling mode, packets are treated uniformly in the IP and MPLS networks, that is, the IP Precedence value and the MPLS EXP bits always are identical. Whenever a router changes or recolors the PHB of a packet, that change must be propagated to all encapsulation markings. The propagation is performed by a router only when a PHB is added or exposed due to label imposition or disposition on any router in the packet's path. The color must be reflected everywhere, at all levels.
- **Pipe Tunneling Mode**
- **Short Pipe Tunneling Mode**

Pipe mode and Short Pipe mode provide QoS transparency. With QoS transparency, the customer's IP marking in the IP packet is preserved.



Note QoS transparency does not support short Pipe mode in this release.

Uniform mode is the default Tunneling Mode available on NCS4K. In this mode, the DiffServ Code Point/MPLS Experimental (EXP) bits as the packet travels from one customer edge (CE) router to another CE router across the MPLS core is as follows:

To implement the pipe tunneling mode, Conditional marking of MPLS experimental bits and DSCP preservation for L3VPN Traffic are enabled.

Conditional Marking of MPLS experimental bits for L3VPN Traffic

The conditional marking of MPLS experimental bits is achieved for Layer 3 VPN traffic by applying a combination of ingress and egress policy maps on the provider edge (PE) router. In the ingress policy map, the QoS-group or discard class is set either based on the result of the policing action or implicitly. The egress policy map matches on qos-group or discard-class and sets the MPLS experiment bits to the corresponding value.

Conditional marking of the MPLS experimental bits is done differently for in-contract and out-of-contract packets. In-contract packets are the confirmed packets with the color green and discard-class set to 0. Out-of-contract packets have exceeded the limit and have the color yellow and discard class set to 1.

Conditional marking of MPLS experimental bits for L3VPN can be done on physical and bundle main interfaces and subinterfaces.

The DSCP value of the IP packet is preserved for L3VPN networks when the experimental remarking is done on the imposition node. The original IP DSCP value is preserved on the disposition node, and egress queuing is done based on MPLS EXP.

Tunneling Pipe Mode

Tunneling Pipe Mode uses two layers of QoS:

- An underlying QoS for the data, which remains unchanged when traversing the core.
- A per-core QoS, which is separate from that of the underlying IP packets. This per-core QoS pre-hop behavior(PHB) remains transparent to end users.

When a packet reaches the edge of the MPLS core, the egress PE router (PE2) classifies the newly exposed IP packets for outbound queuing based on the MPLS PHB from the EXP bits of the recently removed label.

Additional capabilities across Imposition-PE, Core-P, and Disposition-PE nodes are enabled to enable the Functional requirements of Uniform and Pipe tunneling mode.

Imposition-PE

Following Capabilities are enabled in Imposition-PE:

- Supports implicit copy of precedence or DSCP to experimental bits (default mode)
- Supports explicit marking of experimental bits on ingress or egress
- Supports condition experimental bits marking on egress
- Classification based on L4 protocol

Core-P

Following Capabilities are enabled in Core-P:

- Supports no experimental bits remarking (default mode)
- Set MPLS top most at ingress

Disposition-PE

Following Capabilities are enabled in Disposition-PE:

- Experimental bits not copied to DSCP (Default mode)
- Classification on experimental bits in ingress for egress queuing support (pipe mode)

The following is a sample configuration of no experimental bits marking at ingress.

```
1.Configs at PE-1 (no EXP marking, dscp copied to EXP at imposition)
a.Ingress policy:
class-map prec3
    match precedence 3
end-class-map

policy-map ingressPE1
    class prec3
      set traffic-class 3
end-policy-map

b.Egress policy:
class-map tc3
   match traffic-class 3
end-class-map

policy-map egressPE1
   class tc3
      shape average percent 10
end-policy-map

2.Configs at P (swapping the label)
a.Ingress policy
class-map mpls-in-exp3
  match mpls experimental topmost 3
end-class-map
```

```
policy-map mpls-in-pol
  class mpls-in-exp3
   set mpls experimental topmost 2
end-policy-map

3.Configs at PE 2 (EXP not copied to dscp of exposed IP packet)
a.Ingress policy
class-map mpls-in-exp2
    match mpls experimental topmost 2
end-class-map

policy-map ing-PE2
   class mpls-in-exp2
     set traffic-class 2
end-policy-map

b.Egress policy
class-map tc2
match traffic-class 2
end-policy-map

policy-map egr-PE2
   class tc2
     shape average percent 10
end-policy-map
```

The following is a sample configuration of experimental bits marking at ingress.

```
1.Configs at PE-1 (setting the EXP label - imposition)
a.Ingress policy:
class-map dscpcs3
    match dscp cs3
end-class-map

policy-map ingressPE1
    class dscpcs3
      set qos-group 2
end-policy-map

b.Egress policy:
class-map qgrp2
   match qos-group 2
end-class-map

policy-map egressPE1
   class-map qgrp2
      set mpls experimental imposition 4
end-policy-map

2.Configs at P (swapping the label)
a.Ingress policy

class-map mpls-in-exp4
  match mpls experimental topmost 4
end-class-map

policy-map mpls-in-pol-P1
  class mpls-in-exp4
   set mpls experimental topmost 5
 end-policy-map

3.Configs at PE 2 (outgoing queueing policy)
a.Ingress policy
class-map mpls-in-exp5
    match mpls experimental topmost 5
```

```
end-class-map

policy-map ing-PE2
   class mpls-in-exp5
     set traffic-class 3
end-policy-map

b.Egress policy

class-map tc3
   match traffic-class 3
end-class-map

policy-map egr-PE2
   class tc3
     shape average percent 10
end-policy-map
```

The following is a sample configuration of conditional experimental bits marking at ingress.

```
1.Configs at PE-1 (conditional EXP marking at egress)
a.Ingress policy:
class-map dscpcs3
    match dscp cs3
end-class-map

policy-map ing-PE1
    class dscpcs3
      set qos-group 2
set traffic-class 2
police rate percent 5 peak-rate percent 10
end-policy-map

b.Egress policy:

class-map match-all qgrp2dc1
   match qos-group 2
   match discard-class 1
end-class-map

class-map match-all qgrp2dc0
  match qos-group 2
  match discard-class 0
end-class-map

class-map tc2
  match traffic-class 2
end-class-map

policy-map egr-mark-PE1
   class qgrp2dc1
      set mpls experimental imposition 4
   class qgrp2dc0
      set mpls experimental imposition 6
end-policy-map

policy-map egr-shape-PE1
  class tc2
     shape average percent 5
end-policy-map


2.Configs at P (swapping the label)
a.Ingress policy
```

```
class-map mpls-in-exp4
  match mpls experimental topmost 4
end-class-map

policy-map mpls-in-pol-P1
  class mpls-in-exp4
   set mpls experimental topmost 5
 end-policy-map

3.Configs at PE 2 (outgoing queueing policy)
a.Ingress policy

class-map mpls-in-exp5
    match mpls experimental topmost 5
end-class-map

class-map mpls-in-exp6
    match mpls experimental topmost 6
end-class-map

policy-map ing-PE2
   class mpls-in-exp5
     set traffic-class 3
   class mpls-in-exp6
     set traffic-class 4
end-policy-map

b.Egress policy:
class-map tc3
   match traffic-class 3
end-class-map

class-map tc4
   match traffic-class 4
end-class-map

policy-map egr-PE2
   class tc3
     shape average percent 10
   class tc4
     shape average percent 5
end-policy-map
```

Configuration and Software Restrictions

- In the ingress policy map, if qos-group is set for the incoming traffic packets, then the setting of DSCP and MPLS experimental bits will not work. In other words, set experimental bits/DSCP and set QoS-group cannot be used together. Setting MPLS experimental bits, precedence or DSCP, and QoS-group in the same class on ingress, the result is unpredictable on egress when used for the match.
- Both the ingress and egress policy maps must be applied to attain the expected behavior. If either one of them is not applied, then it might lead to undefined behavior.
- When access control list and QoS co-exist on an interface, the QoS statistics don’t work, whereas the ACL stats always works. When the ACL is removed from the interface, QoS statistics start working.
- All the control traffic hits Class Traffic 6 by default due to the DNX design, we cannot change the queuing.

Configure Layer 3 QinQ

Perform this task to configure the Layer 3 QinQ feature.

```
RP/0/RP0:hostname# configure
RP/0/RP0:hostname(config)# interface Bundle-Ether1000.3
RP/0/RP0:hostname(config-subif)# ipv4 address 192.0.2.1/24
RP/0/RP0:hostname(config-subif)# encapsulation dot1q 3 second-dot1q 4000
RP/0/RP0:hostname(config-subif)# commit
```

Verify Layer 3 QinQ

This section shows the verification of Layer 3 QinQ configuration.

```
RP/0/# show interfaces Bundle-Ether1000.3
Bundle-Ether1000.3 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 0c75.bd30.1c88
  Internet address is 192.0.2.1/24
  MTU 1522 bytes, BW 30000000 Kbit (Max: 30000000 Kbit)
     reliability 255/255, txload 0/255, rxload 6/255
  Encapsulation 802.1Q Virtual LAN, VLAN Id 3, 2nd VLAN Id 4000,
  loopback not set,
  Last link flapped 19:30:41
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 00:01:59
  Last clearing of "show interface" counters never
  5 minute input rate 797298000 bits/sec, 844605 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     59288018302 packets input, 6995904900380 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 2 broadcast packets, 516 multicast packets
     419 packets output, 54968 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
```

VRF on Layer 3 VPN for management interface

Table 4: Feature History

Feature Name	Release Information	Feature Description
VRF on Layer 3 VPN	Cisco IOS XR Release 6.5.35	The Virtual Routing and Forwarding (VRF) feature in Layer 3 VPN allows to optimize IP address usage within the management interface. You can perform an IP lookup check within VRF context and forward packets to correct destination.

VRF and VPN membership

A VRF (Virtual Routing and Forwarding) defines the VPN (Virtual Private Network) membership for a customer site connected to a Provider Edge (PE) router. A customer site can belong to multiple VPNs, but it can only be associated with one VRF.

A VRF contains all the routes available to the site from the VPNs it is part of. The distribution of VPN routing information is managed using VPN route target communities, which are implemented through BGP (Border Gateway Protocol) extended communities.

IPv4 VRF

IPv4 VRF technology allows multiple instances of IPv4 routing tables to exist on the same router or network device. VRF enables the creation of separate routing environments on a single physical network device, ensuring that customer traffic remains distinct and separate, even when using the same underlying infrastructure.

VRF limitations

- You can configure VRF on Layer 3 only on the management interface.
- The current supported VRF scale is 256.
- ACL and QoS cannot be applied on the same interface due to a general limitation across all DNX-based platforms.

Configure Layer 3 VRF on the management interface

The Layer 3 VRF import route target configuration enables exported VPN routes to be imported into the VPN, if the route targets of the exported route match any of the local VPN import route targets.

Use this task to configure a VRF instance with specified import and export route targets (2001:1601).

Procedure

Step 1 Run the **vrf vrf1601** command to configure a VRF instance on the router PE1.

Example:

```
RP/0/RP0:Router-PE1#configure
RP/0/RP0:Router-PE1(config)#vrf vrf1601
```

Step 2 Run the **address-family ipv4 unicast** command to configure the IPv4 unicast address family.

Example:

```
RP/0/RP0:Router-PE1(config-vrf)#address-family ipv4 unicast
```

Step 3 Run the **import route-target** command to configure the import route target and exit the configuration mode.

Example:

```
RP/0/RP0:Router-PE1(config-vrf-af)#import route-target
RP/0/RP0:Router-PE1(config-vrf-af-import-rt)#2001:1601
RP/0/RP0:Router-PE1(config-vrf-af-import-rt)#exit
```

Step 4 Run the **export route-target** command to configure the export route target and commit the changes.

Example:

```
RP/0/RP0:Router-PE1(config-vrf-af)#export route-target
RP/0/RP0:Router-PE1(config-vrf-af-export-rt)#2001:1601
RP/0/RP0:RP/0/RP0 Router-PE1(config-vrf-af-export-rt)#commit
```

The VRF instance is associated with the respective BGP instance.

Step 5 Run the **show runn vrf** command to display the running configuration.

Example:

```
RP/0/RP0:MM05#show runn vrf
vrf  vrf-group
RP/0/RP0:MM05#show runn vrf vrf1601
Tue Jan 28 12:22:36.892 UTC
vrf vrf1601
 address-family ipv4 unicast
  import route-target
   2001:1601
  !
  export route-target
   2001:1601
  !
 !
!
```

Step 6 Run the **show vrf vrf1601** command to verify the configured values.

Example:

```
RP/0/RP0:PE1#show vrf vrf1601
VRF                  RD          RT                  AFI   SAFI
vrf1601              2001:1601
                                 import  2001:1601   IPV4  Unicast
                                 export  2001:1601   IPV4  Unicast
```