# Modular QoS Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 7.6.x

**First Published:** 2022-03-30

**Americas Headquarters**
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2022 Cisco Systems, Inc. All rights reserved.

# CONTENTS


PREFACE

**Preface** ix

Changes to this Document ix

Communications, Services, and Additional Information ix

CHAPTER 1

**New and Changed QoS Features** 1

New and Changed QoS Features 1

CHAPTER 2

**Configuring Modular QoS Service Packet Classification** 3

Packet Classification Overview 4

Traffic Class Elements 6

Default Traffic Class 7

Create a Traffic Class 8

Traffic Policy Elements 9

Create a Traffic Policy 10

Scaling of Unique Ingress Policy Maps 12

Limitations and Restrictions 13

Attach a Traffic Policy to an Interface 13

Packet Marking 15

Handling QoS for Locally Originated Packets 17

Protecting Locally Originated BFD Packets 19

Prioritization of IS-IS and ARP Packets to Manage Transit Traffic 20

QoS Re-marking of IP Packets in Egress Direction 22

QoS Re-marking of Ethernet Packets in Egress Direction 23

QoS L2 Re-marking of Ethernet Packets in Egress Direction 23

QoS L2 Re-Marking of Ethernet Packets on L3 Flows in Egress Direction 24

QoS L2 Re-Marking of Ethernet Packets on L3 Flows in Egress Direction on L3 sub-interfaces 27

Layer 2 Ingress QoS Matching for IPv4 and IPv6 Destination Addresses 32
Overview 33
Guidelines and Limitations 33
Configure Layer 2 Ingress QoS Matching for IPv4 and IPv6 Destination Addresses 33
Bundle Traffic Policies 36
Shared Policy Instance 36
Restrictions and Guidelines 38
Attaching a Shared Policy Instance to Multiple Subinterfaces 38
Ingress Short-Pipe 40
Restrictions and Other Important Points 40
Configure Ingress Short-Pipe 41
Selective Egress Policy-Based Queue Mapping 42
Restrictions and Other Important Points 44
Configure Selective Egress Policy-Based Queue Mapping 44
Configuring QoS Groups with an ACL 47
Configuring an ACL with Fragment Match 50
Restrictions and Guidelines 51
Configuring an ACL with Fragment Match 51
Restrictions 53
In-Place Policy Modification 56
References for Modular QoS Service Packet Classification 57
Specification of the CoS for a Packet with IP Precedence 57
IP Precedence Bits Used to Classify Packets 57
IP Precedence Value Settings 57
IP Precedence Compared to IP DSCP Marking 58
Conditional Marking of MPLS Experimental bits for L3VPN Traffic 58
QoS DSCP Preservation 58
Policy-map for conditional marking of incoming IPv4 and IPv6 traffic 59
Policy-map for conditional marking of outgoing MPLS traffic 59
Conditional Marking of MPLS Experimental bits for L2VPN Traffic 60
Policy-map for conditional marking of incoming traffic 61
Policy-map for conditional marking of outgoing MPLS traffic 61
Conditional Marking of MPLS Experimental Bits for EVPN-VPWS Single-Homing Services 62
Classifying Packets Based On MPLS Experimental Bits in MPLS Over GRE 65

Behavioral Specifications 66
Restrictions 66
Set Up and Configuration for QoS Classification on MPLS Header 66
Terms Used for Classifying Packets Based On MPLS Experimental Bits in MPLS Over GRE 70
QPPB 71
QPPB: Guidelines and Limitations 72
Configuration Workflow 73
Configuring QPPB on an Interface 75
Egress Interface Configuration 75

CHAPTER 3 **Configuring Modular QoS Congestion Avoidance** 77
Modular QoS Congestion Avoidance 77
Tail Drop and the FIFO Queue 78
Tail Drop and the FIFO Queue: Tip to Optimize Hardware Resources 78
Configure Tail Drop 79
Random Early Detection and TCP 80
Configure Random Early Detection 81
Weighted Random Early Detection 83
Average Queue Size for WRED 83
Configure Weighted Random Early Detection 84
Configure WRED Counters by Class 86
Explicit Congestion Notification (ECN) 89
Configure ECN Maximum Marking Probability 93

CHAPTER 4 **Priority Flow Control Overview** 97
Restrictions 97
Configuring Priority Flow Control Thresholds 99
Priority Flow Control Watchdog Overview 105
Configure a Priority Flow Control Watchdog Interval 105
Monitoring and Logging Packet Drops on Lossless PFC-Enabled Queues 108
A Brief Background 109
Highlights of Logging and Monitoring Packet Drops 109
Restrictions for Monitoring and Logging Packet Drops 110

CHAPTER 5 **Configuring Modular QoS Congestion Management** 111

Congestion Management Overview 111

Ingress Traffic Management Model 112

Class-based Weighted Fair Queueing 114

Bandwidth Remaining 114

Configure Minimum Bandwidth and Bandwidth Remaining 114

Low-Latency Queuing with Strict Priority Queuing 116

Configuring Low Latency Queuing with Strict Priority queuing 116

Overhead Accounting 118

Traffic Shaping 121

Configure VOQ-Level Traffic Shaping 121

Port Shaping 123

Key Differences Between VOQ-Level and Port-Level Shapers 123

Burst Size for VOQ-Level Shaper 124

4K Pseudowire on Bundle with QoS Enhancement 124

Restrictions and Guidelines 125

Configure Two-Level Shaper Policy on Main Interface 125

Traffic Policing 127

Committed Bursts and Excess Bursts 128

Single-Rate Policer 128

Configure Traffic Policing (Single-Rate Two-Color) 129

Configure Traffic Policing (Single-Rate Three-Color) 131

Two-Rate Policer 132

Configure Traffic Policing (Two-Rate Three-Color) 133

Packets-Per-Second-Based Policer 135

Per-thousand and Per-million Units 138

Shared Policer 138

Policer Bucket Shared 139

Policer Bucket Referred 139

Shared Policer Statistics 139

Restrictions and Guidelines 140

Configuring Shared Policer 140

References for Modular QoS Congestion Management 142

Committed Bursts 142
Excess Bursts 143
Two-Rate Policer Details 143

CHAPTER 6 **Configuring Modular QoS on Link Bundles** 145
QoS on Link Bundles 145
Load Balancing 146
Configure QoS on Link Bundles 146

CHAPTER 7 **Configuring Hierarchical Modular QoS** 151
Overview of Hierarchical Modular QoS 151
Restrictions for Configuring H-QoS 152
Configuring Hierarchical Queuing 153
Conform Aware Hierarchical Policy Overview 158
Conform Aware Hierarchical Policy Restrictions 159
Configuring Conform Aware Hierarchy Policy 159

CHAPTER 8 **Configure Egress Traffic Management** 163
Egress Traffic Management 164
More About the Egress Traffic Management Model 166
QoS Policy Configuration Rules 167
Configure Egress Traffic Management 168
Low Rate Shaper 184

# Preface

This preface contains these sections:


- Changes to this Document, on page ix
- Communications, Services, and Additional Information, on page ix


## Changes to this Document

This table lists the changes made to this document since it was first published.

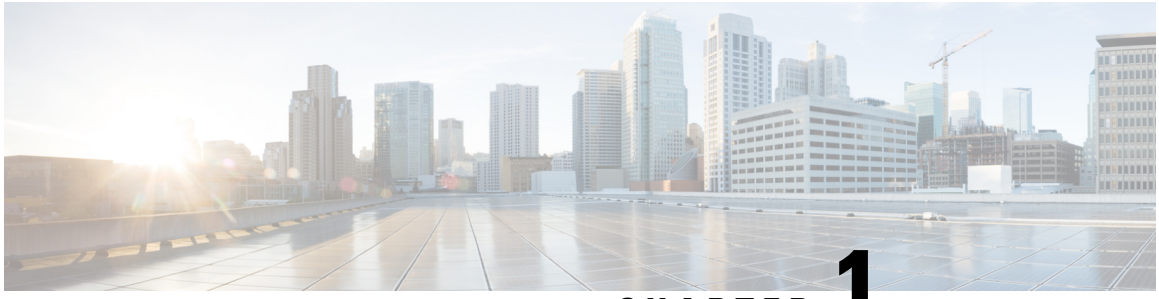
| Date | Summary |
|---|---|
| March 2022 | Initial release of this document. |

## Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

### Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



CHAPTER 1

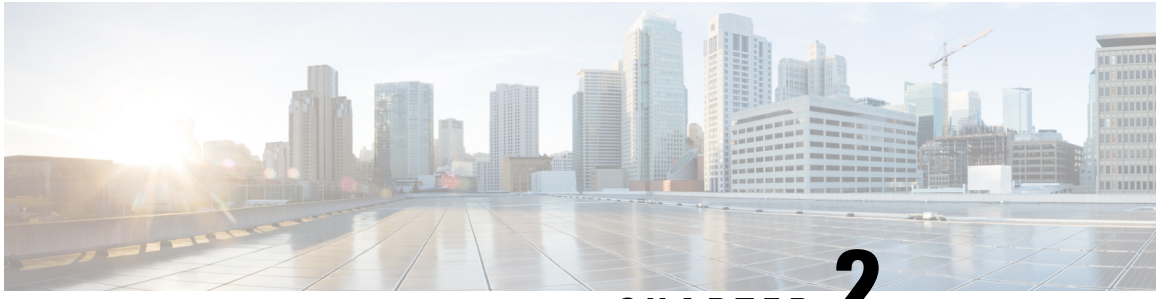
# New and Changed QoS Features


- New and Changed QoS Features, on page 1


## New and Changed QoS Features

*Table 1: QoS Features Added or Modified in IOS XR Release 7.6.x*

| Feature | Description | Changed in Release | Where Documented |
|---|---|---|---|
| Egress Traffic Management | The egress traffic management model employs a two-pass model per NPU core to get up to 50% of NPU core bandwidth as recycle bandwidth. VOQ lookup takes place in the second pass to get the outgoing interface or sub-interface VOQ. This lookup at the second pass enhances the egress queuing policy by making the feature-rich ingress pipeline available to the outgoing traffic. | Release 7.6.1 | Egress Traffic Management, on page 164 |
| Low Rate Shaper | This functionality allows your hardware to program low shaper bandwidth with accuracy. | Release 7.6.1 | Low Rate Shaper, on page 184 |



CHAPTER 2

# Configuring Modular QoS Service Packet Classification

This chapter covers these topics:


- Packet Classification Overview, on page 4
- Traffic Class Elements, on page 6
- Traffic Policy Elements, on page 9
- Ingress Short-Pipe, on page 40
- Selective Egress Policy-Based Queue Mapping, on page 42
- Configuring QoS Groups with an ACL, on page 47
- Configuring an ACL with Fragment Match, on page 50
- Restrictions , on page 53
- In-Place Policy Modification, on page 56
- References for Modular QoS Service Packet Classification, on page 57
- QPPB, on page 71

# Packet Classification Overview

*Table 2: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Cisco NC57 Compatibility Mode: QoS Enablement on Layer 2 MPLS/BGP | Release 7.3.1 | This feature is now supported on routers that have the Cisco NC57 line cards installed and operate in the compatibility mode.<br>The following Layer 2 services are supported:<br>• Local switching [xconnect or bridging]<br>• Layer 2 VPN – Virtual Private Wire Service (VPWS)<br>Apart from packet classification, this feature is available for the following QoS operations:<br>• Modular QoS Congestion Avoidance , on page 77<br>• Configuring Modular QoS Congestion Management, on page 111<br>• QoS on Link Bundles, on page 145<br>• Configuring Hierarchical Modular QoS, on page 151 |

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Cisco NC57 Native Mode: QoS Enablement on Layer 2 Services | Release 7.4.1 | This feature is now supported on routers that have the Cisco NC57 line cards installed and operate in the native mode.<br>The following Layer 2 services are supported:<br>• Local switching [xconnect or bridging]<br>• Layer 2 VPN – Virtual Private Wire Service (VPWS)<br>Apart from packet classification, this feature is available for the following QoS operations:<br>• Modular QoS Congestion Avoidance , on page 77<br>• Configuring Modular QoS Congestion Management, on page 111<br>• QoS on Link Bundles, on page 145<br>• Configuring Hierarchical Modular QoS, on page 151 |

Packet classification involves categorizing a packet within a specific group (or class) and assigning it a traffic descriptor to make it accessible for QoS handling on the network. The traffic descriptor contains information about the forwarding treatment (quality of service) that the packet should receive. Using packet classification, you can partition network traffic into multiple priority levels or classes of service. The source agrees to adhere to the contracted terms and the network promises a quality of service. Traffic policers and traffic shapers use the traffic descriptor of a packet to ensure adherence to the contract.

Traffic policers and traffic shapers rely on packet classification features, such as IP precedence, to select packets (or traffic flows) traversing a router or interface for different types of QoS service. After you classify packets, you can use other QoS features to assign the appropriate traffic handling policies including congestion management, bandwidth allocation, and delay bounds for each traffic class.

The Modular Quality of Service (QoS) CLI (MQC) defines the traffic flows that must be classified, where each traffic flow is called a class of service, or class. Later, a traffic policy is created and applied to a class. All traffic not identified by defined classes fall into the category of a default class.

You can classify packets at the ingress on L3 subinterfaces for (CoS, DEI) for IPv4, IPv6, and MPLS flows. IPv6 packets are forwarded by paths that are different from those for IPv4. To enable classification of IPv6 packets based on (CoS, DEI) on L3 subinterfaces, run the hw-module profile qos ipv6 short-l2qos-enable command and reboot the line card for the command to take effect.

Starting with Cisco IOS XR Release 7.4.1 systems with Cisco NC57 line cards running in native mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

# Traffic Class Elements

The purpose of a traffic class is to classify traffic on your router. Use the **class-map** command to define a traffic class.

A traffic class contains three major elements:

- A name
- A series of **match** commands - to specify various criteria for classifying packets.
- An instruction on how to evaluate these **match** commands (if more than one **match** command exists in the traffic class)

Packets are checked to determine whether they match the criteria that are specified in the **match** commands. If a packet matches the specified criteria, that packet is considered a member of the class and is forwarded according to the QoS specifications set in the traffic policy. Packets that fail to meet any of the matching criteria are classified as members of the default traffic class.

This table shows the details of match types that are supported on the router.

| Match Type Supported | Min, Max | Max Entries | Support for Match NOT | Support for Ranges | Direction Supported on Interfaces |
|---|---|---|---|---|---|
| IPv4 DSCP<br>IPv6 DSCP<br>DSCP | (0,63) | 64 | Yes | Yes | Ingress |
| IPv4 Precedence<br>IPv6 Precedence<br>Precedence | (0,7) | 8 | Yes | No | Ingress |
| MPLS Experimental Topmost | (0,7) | 8 | Yes | No | Ingress |
| Access-group | Not applicable | 8 | No | Not applicable | Ingress |
| QoS-group | (1,7)<br>(1,511) for peering profile | 7 | No | No | • Egress<br>• Ingress for QoS Policy Propagation Using Border Gateway Protocol (QPPB)<br>• Ingress for peering profile |
| Traffic-class | (1,7) | 7 | No | No | • Egress |

| Match Type Supported | Min, Max | Max Entries | Support for Match NOT | Support for Ranges | Direction Supported on Interfaces |
|---|---|---|---|---|---|
| Protocol | (0,255) | 1 | Yes | Not applicable | Ingress |

**Note** Egress queue statistics are displayed only for those classes which have a corresponding match criteria in the egress. Therefore, if you have a **set traffic-class** *x* configured in the ingress, you must have a corresponding **match traffic-class** *x* in the egress, in order to see the statistics in the egress side.

**Note** A maximum value of up to 64 unique queues is supported. Each unique queue-limit consumes one rate profile in the Traffic manager. Out of 64 unique queues, few are reserved for default configs and the remaining are usable.

Depending on the interface speeds, default configurations consume some of the rate profiles. The remaining rate profiles can be exhausted in the following scenarios:

- Different shape rates without configuring queue limits could exhaust the rate profiles as 10ms of guaranteed service rate converts to a different value in bytes based on the shape rate.
- Configuring queue limits in units of time could exhaust the rate profiles. For example, 20 ms of 50 Mbps and 20 ms of 100 Mbps are two different values in bytes.

**Tip** You can avoid exhausting rate profiles by configuring queue limits in absolute units (such as bytes, kilobytes, or megabytes) for class maps and sharing these limits with the policy maps.

# Default Traffic Class

Unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as belonging to the default traffic class.

If the user does not configure a default class, packets are still treated as members of the default class. However, by default, the default class has no enabled features. Therefore, packets belonging to a default class with no configured features have no QoS functionality. These packets are then placed into a first in, first out (FIFO) queue and forwarded at a rate determined by the available underlying link bandwidth. This FIFO queue is managed by a congestion avoidance technique called tail drop.

For egress classification, match on **traffic-class** (1-7) is supported. Match **traffic-class 0** cannot be configured. The class-default in the egress policy maps to **traffic-class 0**.

This example shows how to configure a traffic policy for the default class:

```
configure
 policy-map ingress_policy1
 class class-default
  police rate percent 30
```

!

# Create a Traffic Class

To create a traffic class containing match criteria, use the **class-map** command to specify the traffic class name, and then use the **match** commands in class-map configuration mode, as needed.

**Guidelines**

- Users can provide multiple values for a match type in a single line of configuration; that is, if the first value does not meet the match criteria, then the next value indicated in the match statement is considered for classification.
- Use the **not** keyword with the **match** command to perform a match based on the values of a field that are not specified.
- All **match** commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.
- If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. This is the default. If you specify **match-all**, the traffic must match all the match criteria.
- From Release 7.7.1 onwards, for the **match access-group** command, QoS classification based on the packet length field in the IPv4 and IPv6 headers is supported. Prior to this, support was not available for packet length and TTL (time to live) fields.
- Deny action in the ACL or an empty ACL (contains no rules, only remarks) is not supported for the **match access-group command** when an ACL list is used within a **class-map**.

  The configuration is rejected when you attach such a **policy-map** to an interface.
- The **traffic-class** and **discard-class** are supported only in egress direction, and these are the only match criteria supported in egress direction.
- The egress default class implicitly matches **qos-group** 0 for marking policy and **traffic-class** 0 for queuing policy.
- Multicast takes a system path that is different than unicast on router, and they meet later on the egress in a multicast-to-unicast ratio of 20:80 on a per interface basis. This ratio is maintained on the same priority level as that of the traffic.
- Egress QoS for multicast traffic treats traffic classes 0-5 as low-priority and traffic classes 6-7 as high priority. Currently, this is not user-configurable.
- Egress shaping does not take effect for multicast traffic in the high priority (HP) traffic classes. It only applies to unicast traffic.
- If you set a traffic class at the ingress policy and do not have a matching class at egress for the corresponding traffic class value, then the traffic at ingress with this class will not be accounted for in the default class at the egress policy map.
- Only traffic class 0 falls in the default class. A non-zero traffic class assigned on ingress but with no assigned egress queue, falls neither in the default class nor any other class.

### Configuration Example

You have to accomplish the following to complete the traffic class configuration:

1. Creating a class map
2. Specifying the match criteria for classifying the packet as a member of that particular class

   (For a list of supported match types, see Traffic Class Elements, on page 6.)

```
Router# configure
Router(config)# class-map match-any qos-1
Router(config-cmap)# match qos-group 1
Router(config-cmap)# end-class-map
Router(config-cmap)# commit
```

Use this command to verify the class-map configuration:

```
Router#show class-map qos-1
1) ClassMap: qos-1    Type: qos
    Referenced by 2 Policymaps
```

Also see, Running Configuration, on page 13.

Also see, Verification, on page 14.

### Related Topics

- Traffic Class Elements, on page 6
- Traffic Policy Elements, on page 9

### Associated Commands

- class-map
- match access-group
- match dscp
- match mpls experimental topmost
- match precedence
- match qos-group

# Traffic Policy Elements

A traffic policy contains three elements:

- Name
- Traffic class
- QoS policies

After choosing the traffic class that is used to classify traffic to the traffic policy, the user can enter the QoS features to be applied to the classified traffic.

The MQC does not necessarily require that the users associate only one traffic class to one traffic policy.

The order in which classes are configured in a policy map is important. The match rules of the classes are programmed into the TCAM in the order in which the classes are specified in a policy map. Therefore, if a packet can possibly match multiple classes, only the first matching class is returned and the corresponding policy is applied.

The router supports 32 classes per policy-map in the ingress direction and 8 classes per policy-map in the egress direction.

This table shows the supported class-actions on the router.

| Supported Action Types | Direction supported on Interfaces |
|---|---|
| minimum-bandwidth | egress |
| bandwidth-remaining* | egress |
| mark | (See Packet Marking, on page 15) |
| police | ingress |
| priority | egress (level 1 to level 7) |
| queue-limit | egress |
| shape | egress |
| wred | egress |

*Bandwidth and Bandwidth remaining configurations are not supported simultaneously within the same policy-map in H-QoS mode.

WRED supports **default** and **discard-class** options; the only values to be passed to the discard-class being 0 and 1.

# Create a Traffic Policy

The purpose of a traffic policy is to configure the QoS features that should be associated with the traffic that has been classified in a user-specified traffic class or classes.

**Note** Starting from Cisco IOS XR Release 7.6.1, the ingress QoS policy you configure, also applies to all the transit Ethernet Connectivity Fault Management (CFM) packets. With QoS features such as marking and policing now applied on the transit CFM packets, you ensure that these packets don't consume excess bandwidth and other QoS resources in the network.

To configure a traffic class, see Create a Traffic Class, on page 8.

After you define a traffic policy with the **policy-map** command, you can attach it to one, or more interfaces to specify the traffic policy for those interfaces by using the **service-policy** command in interface configuration

mode. With dual policy support, you can have two traffic policies, one marking and one queuing attached at the output. See, Attach a Traffic Policy to an Interface, on page 13.

### Configuration Example

You have to accomplish the following to complete the traffic policy configuration:

1. Creating a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Specifying the class-action(s) (see Traffic Policy Elements, on page 9)

```
Router# configure
Router(config)# policy-map  test-shape-1
Router(config-pmap)# class qos-1

/* Configure class-action ('shape' in this example).
Repeat as required, to specify other class-actions */
Router(config-pmap-c)# shape average percent 40
Router(config-pmap-c)# exit

/* Repeat class configuration as required, to specify other classes */

Router(config-pmap)# end-policy-map
Router(config)# commit
```

See, Running Configuration, on page 13.

See, Verification, on page 14.

### Related Topics

- Traffic Policy Elements, on page 9
- Traffic Class Elements, on page 6

### Associated Commands

- bandwidth
- bandwidth remaining
- class
- police
- policy-map
- priority
- queue-limit
- service-policy
- set discard-class
- set dscp
- set mpls experimental

- set precedence
- set qos-group
- shape

# Scaling of Unique Ingress Policy Maps

*Table 3: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Scaling of Unique Ingress Policy Maps | Release 7.4.1 | This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native mode. |
| Scaling of Unique Ingress Policy Maps | Release 7.3.1 | With this feature, unique policy maps associated to the same template are shared in TCAM, thus enabling scaling of — or creating more number of — policy maps. |

Traditionally, when unique policy maps were associated to the same template — that is, having the same match criteria and actions in the *same* order — each map was assigned a different TCAM entry. This resulted in inefficient TCAM entry management and also restricted the number of policy maps that could be created.

With this functionality, unique policy maps associated to the same template are shared in TCAM, thus enabling scaling of—in other words, creating more number of—policy maps. The other way to understand this functionality is that two policy maps with the same combination of criteria and actions use one template. This way, up to 250 templates are supported for association to policy map combinations.

As an example, consider the following policy maps (**policy-map ncs_input1** and **policy-map ncs_input2**) having the same class maps (**class COS7_DEI0** and class **COS7_DEI1**):

```
class-map match-all COS7_DEI0
 match cos 0
 end-class-map
class-map match-all COS7_DEI1
 match cos 1
 end-class-map

policy-map ncs_input1
 class COS7_DEI0
  set trafiic class 1
  police rate 10 mbps
!
 class COS7_DEI1
  set traffic class 2
  policer rate 20 mbps
!

policy-map ncs_input2
 class COS7_DEI0
  set traffic class 1
  police rate 30 mbps
!
```

```
 class COS7_DEI1
  set traffic class 2
  policer rate 40 mbps
!
```

Earlier, when the policy maps were attached to interface, they used different TCAM entries, although the match criteria and actions were the same, except for the policer action.

With this functionality, both policy maps share the TCAM entry instead of selecting different entries, thus freeing up TCAM entries for more policy maps.

## Limitations and Restrictions

- Policy Maps share TCAM entries only for the same match criteria and actions or template. However, the policer action can be different for the same template.
- For all unique policy maps the maximum number of templates supported is 250.

# Attach a Traffic Policy to an Interface

After the traffic class and the traffic policy are created, you must attach the traffic policy to interface, and specify the direction in which the policy should be applied.

**Note** When a policy-map is applied to an interface, the transmission rate counter of each class is not accurate. This is because the transmission rate counter is calculated based on the exponential decay filter.

### Configuration Example

You have to accomplish the following to attach a traffic policy to an interface:

1. Creating a traffic class and the associated rules that match packets to the class (see #unique_24 )
2. Creating a traffic policy that can be attached to one or more interfaces to specify a service policy (see Create a Traffic Policy, on page 10 )
3. Associating the traffic class with the traffic policy
4. Attaching the traffic policy to an interface, in the ingress or egress direction

```
Router# configure
Router(config)# interface HundredGigE 0/6/0/18
Router(config-int)# service-policy output test-shape-1
Router(config-int)# commit
```

### Running Configuration

```
/* Class-map configuration */

class-map match-any traffic-class-1
 match traffic-class 1
 end-class-map
```

```
!
- - -
- - -

/* Traffic policy configuration */
policy-map test-shape-1
 class traffic-class-1
  shape average percent 40
 !
 class class-default
 !
 end-policy-map
!
- - -
- - -

/* Attaching traffic policy to an interface in egress direction */
interface HundredGigE0/6/0/18
 service-policy output test-shape-1
 !
```

## Verification

```
Router# show qos interface hundredGigE 0/6/0/18 output

NOTE:- Configured values are displayed within parentheses Interface HundredGigE0/6/0/18 ifh
 0x30001f8  -- output policy
NPU Id:                        3
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
VOQ Base:                      11112
VOQ Stats Handle:              0x88430698
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   qos-1
Egressq Queue ID                          =   11113 (LP queue)
Queue Max. BW.                            =   40329846 kbps (40 %)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   40000000 kbps
TailDrop Threshold                        =   50069504 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                              =   class-default
Egressq Queue ID                          =   11112 (Default LP queue)
Queue Max. BW.                            =   101803495 kbps (default)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   50000000 kbps
TailDrop Threshold                        =   62652416 bytes / 10 ms (default)
WRED not configured for this class
```

## Related Topics

- Traffic Policy Elements, on page 9
- Traffic Class Elements, on page 6

### Associated Commands

- service-policy

# Packet Marking

**Note** L2 packet marking is not supported on NC57-24DD and NC57-18DD-SE line cards for Cisco IOS XR Release 7.0.2.

The packet marking feature provides users with a means to differentiate packets based on the designated markings. The router supports egress packet marking. match on **discard-class** on egress, if configured, can be used for a marking policy only.

The router also supports L2 ingress marking.

For ingress marking:

Ingress traffic— For the ingress pop operation, re-marking the customer VLAN tag (CoS, DEI) is not supported.

Egress traffic— The ingress 'pop VLAN' is translated to a 'push VLAN' for the egress traffic, and (CoS, DEI) marking is supported for newly pushed VLAN tags. If two VLAN tags are pushed to the packet header at the egress side, both inner and outer VLAN tags are marked. For example:

1. rewrite ingress tag pop 1 symmetric

2. rewrite ingress tag pop 2 symmetric

3. rewrite ingress tag translate 2-to-1 dot1q/dot1ad <> symmetric

In case of pop action, the outer VLAN tag (CoS, DEI) is retained on NC57 line cards.

Single tag— When symmetrical pop 1 action is performed, the outer tag (CoS, DEI) is retained as the original frame.

Double tag— When symmetrical pop 2 action is performed, the outer tag (CoS, DEI) is retained as the original frame and inner tag (CoS, DEI) is set to 0/0.

### Packet Marking Guidelines and Limitations

- While marking a packet, ensure you don't set the IP DSCP (using the **set dscp** command) and the MPLS experimental imposition values (using the **set mpls experimental imposition** command) for the same class map. Else, neither the DSCP remarking nor the MPLS EXP values may take effect at the ingress. This will cause, per default QoS behavior, the IP precedence values to be copied to the EXP bits on the imposed packets. Such an action could lead to unintended packets marked as high-priority by your customer being forwarded as high-priority MPLS packets in the network.

- The statistics and counters for the egress marking policy cannot be viewed on the router.

- QoS EXP matching for egress doesn't work for Layer 2 VPN and Layer 3 VPN traffic flowing from:

  - Cisco NCS 5700 series line cards at ingress to Cisco NCS 5500 series line cards at the egress

  and

  - from Cisco NCS 5500 series line cards at ingress to Cisco NCS 5700 series line cards at egress.

- For QOS EXP-Egress marking applied on a Layer 3 interface on Cisco NCS550x and NCS55Ax routers, there is a limit of two unique policy maps per NPU. This limit is three unique policy maps per NPU for routers that have the Cisco NC57 line cards installed.

  You can apply these policies to as many interfaces as your system resources allow. However, if you apply more than the permitted limit of unique policies, you may encounter unexpected failure.

- For QOS egress marking (CoS, DEI) applied on a Layer 2 interface, there is a limit of 13 unique policy-maps per NPU. If you exceed this number, you may encounter unexpected failure.

### Supported Packet Marking Operations

This table shows the supported packet marking operations.

| Supported Mark Types | Range | Support for Unconditional Marking | Support for Conditional Marking |
|---|---|---|---|
| set dscp | 0-63 | ingress | No |
| set QoS-group | 0-7 | ingress | No |
| set traffic-class | 0-7 | ingress | No |

### Class-based Unconditional Packet Marking

***Table 4: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Class-based Unconditional Packet Marking | Release 7.4.1 | This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes. |

The packet marking feature allows you to partition your network into multiple priority levels or classes of service, as follows:

- Use QoS unconditional packet marking to set the IP precedence or IP DSCP values for packets entering the network. Routers within your network can then use the newly marked IP precedence values to determine how the traffic should be treated.

  On ingress direction, after matching the traffic based on either the IP Precedence or DSCP value, you can set it to a particular discard-class. Weighted random early detection (WRED), a congestion avoidance technique, thereby uses discard-class values to determine the probability that a packet is dropped.

  If however, you set a discard-class of 3, the packet is dropped at ingress itself.

- Use QoS unconditional packet marking to assign MPLS packets to a QoS group. The router uses the QoS group to determine how to prioritize packets for transmission. To set the traffic class identifier on MPLS packets, use the **set traffic-class** command in policy map class configuration mode.

**Note** Setting the traffic class identifier does not automatically prioritize the packets for transmission. You must first configure an egress policy that uses the traffic class.

**Note**

- Unless otherwise indicated, the class-based unconditional packet marking for Layer 3 physical interfaces applies to bundle interfaces.

- From IOS XR Release 7.2.1 onwards with NC57 line cards, propagation of PREC->COS marking happens by default on egress Layer 3 subinterfaces. This applies to single and double-tag L3 subinterfaces, and to NC57 line cards in Native mode.

## Handling QoS for Locally Originated Packets

Packets that are generated and transmitted by a router are called Locally Originated Packets (LOPs). These are different from packets that pass through the router. Each device uses a default precedence value as determined by the device. The default value, used by Locally Originated Control Protocols (LOCPs) such as BGP, OSPF, CCM(CSM), and RSVP, is a precedence of 6 or Differentiated Services Codepoint (DSCP) of 48. Locally Originated Management Protocols (LOMPs) such as Telnet and SSH use a precedence value of 2 or DSCP of 16. SNMP uses a precedence value of 0. Some protocols such as BGP, RSVP, CFM, and LDP and the management protocols are capable of setting a specific precedence or DSCP value.

**Note** Bidirectional Forwarding Detection (BFD) uses a DSCP value of 48 (CS6) for single- and multi-hops sessions on NCS55xx.

The following applies to Traffic Class (TC) alignment:

- Before Release 7.6.1:
  - Locally generated control plane packets, such as IS-IS and BGP, are generated using traffic-class 6.
  - Locally generated BFD over Bundle (IETF) packets, which are generated on the Network Processing Unit (NPU), are generated using traffic-class 6.
- From Release 7.6.1 onwards:
  - Locally generated control plane packets, such as IS-IS and BGP, are generated using traffic-class 7.
  - Locally generated BFD over Bundle (IETF) packets, which are generated on the Network Processing Unit (NPU), are generated using traffic-class 7.
- When the BFD packets are offloaded to the hardware and generated on the NPU, the egress QoS policies are applied. These packets are classified along with the regular data plane traffic.

On the router, datapath packets and injected packets aren't differentiated if both their traffic classes share the same Virtual Output Queues (VOQs). Therefore, in the case of a congested VOQ, the LOCP packets are dropped. To avoid the LOCP packets drop, Cisco recommends that you have a different traffic class for data path traffic. Alternatively, you can also specify a higher bandwidth for traffic-class 7 (if ingress traffic rate is predictable).

Classifying traffic helps the router to recognize traffic as a certain type and mark that traffic. By marking traffic early on its travel, you can prevent excessive reclassification later. You can mark traffic at the protocol level as shown in the following examples:

### Ethernet

The following configuration shows that the outbound Control Hub packets are marked with a precedence value of 2 and EXP of 2, instead of a precedence and EXP value of 6. The SSH packets have a precedence value of 3 instead of 2.

```
ethernet cfm
  mep domain FOO service FOOBAR mep-id 1
   cos 2
ssh server dscp 24
```

### BGP

```
neighbor x.x.x.x dscp
```

### MPLS LDP

```
mpls ldp signalling dscp
```

### Telnet

```
telnet ipv4 dscp
```

### SNMP

```
snmp-server ipv4 precedence/dscp
```

### Syslog

```
logging ipv4 precedence/dscp
netflow
flow exporter-map TEST dscp
```

### NTP

```
ntp ipv4 precedence/dscp
ssh client dscp 56
ssh server dscp 56
```

**Note** By default, the router marks the Precision Time Protocol (PTP) traffic as high priority. Therefore, the need to prioritize PTP traffic in the QoS configuration is not required.

All LOCPs originating on the RP or LC CPU have the discard priority set in the appended Buffer Header (BHDR). The discard priority ensures that the LOCPs are not dropped internally (under normal circumstances). Such LOCPs include non-IP (IS-IS and ARP) based control packets. The discard priority is not set for LOMPs. Therefore, such packets are treated as normal traffic, both in terms of classification and re-marking, and may be dropped under congestion conditions. Therefore, you must ensure that you do not inadvertently re-mark and drop such traffic.

**Note** By default, all LOCPs are assigned to traffic-class 7. Considering that LOCPs and LOMPs are generated by the RP, an Ingress QoS policy cannot be applied. Therefore, you must ensure that the egress QoS policy includes a class-map which matches traffic-class 7. By definition, the egress QoS policy matches all implicitly marked packets.

LOCPs are not subject to traffic policing, Weighted Random Early Detection (WRED), or Tail-drop queue-limit operation. The LOCP packets are not subject to WRED, even if the **max_th** value is being met. The tail-drop queue-limit must be hit before the LOCP packets are dropped.

All LOCPs with the discard priority set are by default put into an implicitly allocated high priority queue of each physical egress interface.

By default, all LOCPs that have the discard priority set are put into an implicitly allocated high priority queue of each physical egress interface.

When configuring QoS policies, you may attach a policy to the physical interface, which then references the sub-interfaces. Or, alternatively, you may attach QoS policies to the sub-interfaces directly. If you attach QoS policies to the sub-interfaces directly, the operator is prevented from attaching a QoS policy to the physical interface. LOCPs, including those being transmitted on a sub-interface, are always sent out on the default high-priority queue of the physical interface. The operator is therefore prevented from assigning any bandwidth to the physical interface, which could be reserved for use by LOCPs. During over-subscription, it may lead to a LOCPs drop and as a result, sessions may be terminated.

To prevent session termination, a minimum bandwidth of MIN (1% of interface BW, 10 mbps) is reserved for the default high-priority queue associated with the physical interface that has no QoS policy applied. If a QoS policy is applied to the physical interface, the minimum bandwidth for the default HP queue is controlled by the configured policy.

- Any QoS classification does not affect the queue-selection for LOCP.
- Irrespective of the QoS policy configured, non-IP LOP control packets are always sent to the high-priority queue. For example, ISIS and ARP

LOCPs can be mapped to a corresponding QoS group. The following example illustrates how this can be achieved:

```
control-plane
 ! local control-packets
  copy precedence qos-group
```

The precedence value of the control packet is mapped to the respective QoS group number.

## Protecting Locally Originated BFD Packets

For releases before Release 7.6.1, BFD packets are injected into traffic-class 6, with drop priority 0 (equivalent of discard-class 0). If transit traffic is also classified into traffic-class 7 and the associated egress queue is congested, BFD packets may be dropped.

From Release 7.6.1, BFD packets are injected into traffic-class 7, with drop priority 0 (equivalent of discard-class 0). If transit traffic is also classified into traffic-class 7 and the associated egress queue is congested, BFD packets may be dropped.

The recommendation is to configure transit traffic-class 7 in the ingress QoS policy with discard-class 1 or 2. You must then configure WRED in the egress QoS policy that drops these packets before dropping discard-class 0.

**Note** The default queue length is 16 ms.

Example

```
class-map match-any NC
 match traffic-class 6 (for releases before Release 7.6.1)
 match traffic-class 7 (from Release 7.6.1)
!
policy-map CORE-OUT
 class NC
  random-detect discard-class 1 7 ms 8 ms
```

Hardware Programming

```
RP/0/RP0/CPU0:R26-Spine2-5508#sh qos interface hu0/0/0/35 output
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/35 ifh 0x1408  -- output policy
NPU Id:                        5
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Policy Name:                 7  wred-out
SPI Id:                        0x0
VOQ Base:                      1592
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   NC
Egressq Queue ID                          =   1598 (LP queue)
Queue Max. BW.                            =   20480000 kbps (20 %)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   20000000 kbps
Peak burst                                =   33600 bytes (default)
TailDrop Threshold                        =   40108032 bytes / 16 ms (default)

WRED profile for Discard_Class 1
WRED Min. Threshold                       =   17563648 bytes (7 ms)
WRED Max. Threshold                       =   20054016 bytes (8 ms)

Default RED profile
WRED Min. Threshold                       =   0 bytes (0 ms)
WRED Max. Threshold                       =   0 bytes (0 ms)

WRED ECN                                  =   Disabled
```

## Prioritization of IS-IS and ARP Packets to Manage Transit Traffic

***Table 5: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Prioritization of IS-IS and ARP Packets to Manage Transit Traffic | Release 7.5.1 | This feature gives you the option to assign the highest priority to IS-IS and Address Resolution Protocol (ARP) packets in transit. This feature is disabled by default.<br><br>The feature provides more flexibility in transit traffic management on a per-hop basis and also fine-tunes the traffic profile management for transit traffic.<br><br>This feature introduces the **hw-module profile qos arp-isis-priority-enable** command. |

## Overview of IS-IS and ARP Traffic Prioritization

Transit traffic refers to all traffic that enters an ingress interface, is compared against the forwarding table entries, and forwarded out an egress interface toward its destination. While the exact path of the transit path may not be of interest to the sender or receiver, you may still want some of the Integrated Intermediate System-to-Intermediate System (IS-IS) and Address Resolution Protocol (ARP) transit traffic to be managed and routed efficiently between specific source and destination addresses. You can now achieve higher levels of flexibility and fine-tune the traffic profile management for transit traffic by enabling the ability to assign the highest priority level to IS-IS and ARP traffic on Layer 2 networks.

This feature is useful if you manage environments such as data centers where you have complete end-to-end control over your network, and you want to avoid any drops in IS-IS and ARP traffic during congestion.

Cisco IOS XR Release 7.5.1 introduces the **hw-module profile qos arp-isis-priority-enable** command to enable prioritization of IS-IS and ARP traffic in transit on Layer 2 networks. Configuring this command assigns a priority level of TC 7 to transit traffic.

**Note** Assigning highest priority levels to IS-IS and ARP traffic lead to higher volumes of ARP traffic and flood high-priority queues, which may cause other traffic to drop. It's best to assign priority levels based on your network configuration and traffic volume.

## Guidelines

- This feature is disabled by default. Configure the **hw-module profile qos arp-isis-priority-enable** command to enable prioritization of IS-IS and ARP traffic in transit on Layer 2 networks.
- This feature is supported on routers that have the Cisco NC57 line cards installed and operate in the native and compatibility mode.
- Configuring **hw-module profile qos arp-isis-priority-enable** assigns a priority level of TC 7 for ISIS and ARP traffic. When this feature is in its default state of disabled, the default priority level is TC 0.
- This feature doesn't allow you to assign TC values.
- Reload the line card for the **hw-module** command to be functional.

## Enabling IS-IS and ARP Traffic Prioritization

To enable IS-IS and ARP traffic prioritization, configure the **hw-module profile qos arp-isis-priority-enable** command.

```
Router#config
Router(config)#hw-module profile qos arp-isis-priority-enable
Router(config)#commit
Router(config)#exit
Router# reload location <lc location>
```

Prioritization is based on IS-IS destination MAC address (01:80:c2:00:00:14 and 01:80:c2:00:00:15) and ARP ether type 0x080. When you configure the **hw-module profile qos arp-isis-priority-enable** command, priority level for IS-IS and ARP and traffic is set as TC 7.

### Verification

The following example shows the verification command for NC 57 line cards. The assigned priority level is **TC 07.**

```
RP/0/RP0/CPU0:ios#show controllers fia diag 0 "pp vis pkttm" location 0/0/CPU0

Node ID: 0/0/CPU0

R/S/I: 0/0/0
==================================================
|     Packet TM Information Core=0 Results      |
==================================================
| Name                        | Value   | Size   |
|                             |         | (bits) |
==================================================
| Fwd_Action_Destination      | 0c353e  |  21    |
|   type                      | PORT_ID |  6     |
|   value                     | 353e    |  15    |
| Tc                          | 07      |  3     |
| Dp                          | 00      |  2     |
| Int_Stat_Meter_Dp_Cmd       | 00      |  4     |
| Lag_Lb_Key                  | 1772    |  16    |
| St_Vsq_Ptr                  | 00      |  8     |
| Visibility                  | 01      |  1     |
| System_Headers_Size         | 28      |  7     |
| NWK_Header_Truncate_Size    | 00      |  8     |
| NWK_Header_Append_Size_Ptr  | 02      |  8     |
| Counter_ID_0                | N/A     |  0     |
| Counter_ID_1                | N/A     |  0     |
| Counter_ID_2                | N/A     |  0     |
| Counter_ID_3                | N/A     |  0     |
| Counter_ID_4                | N/A     |  0     |
| Counter_ID_5                | N/A     |  0     |
| Counter_ID_6                | N/A     |  0     |
| Counter_ID_7                | 00a000  |  20    |
| Counter_ID_8                | N/A     |  0     |
| Counter_ID_9                | N/A     |  0     |
| Meter_ID_0                  | N/A     |  0     |
| Meter_ID_1                  | N/A     |  0     |
| Meter_ID_2                  | N/A     |  0     |
| Ethernet_Meter_ID           | 02      |  3     |
| snif0_cmd                   | 00      |  5     |
| snif0_code                  | 0009    |  9     |
| snif0_qualifier             | 00      |  8     |
| snif1_cmd                   | 00      |  5     |
| snif1_code                  | 01e0    |  9     |
| snif1_qualifier             | 00      |  8     |
| snif2_cmd                   | 00      |  5     |
| snif2_code                  | 0000    |  9     |
| snif2_qualifier             | 00      |  8     |
==================================================
```

# QoS Re-marking of IP Packets in Egress Direction

The router support the marking of IP DSCP bits of all IP packets to zero, in the egress direction. This feature helps to re-mark the priority of IP packets, which is mostly used in scenarios like IP over Ethernet over MPLS over GRE. This functionality is achieved using the ingress policy-map with **set dscp 0** option configured in class-default.

### Configuration Example

```
Router# configure
Router(config)# policy-map ingress-set-dscp-zero-policy
Router(config-pmap)# class class-default
```

```
Router(config-pmap-c)# set dscp 0
Router(config-pmap-c)# end-policy-map
Router(config-pmap)# commit
```

#### Running Configuration

```
policy-map ingress-set-dscp-zero-policy
class class-default
  set dscp 0
!
end-policy-map
!
```

# QoS Re-marking of Ethernet Packets in Egress Direction

The router supports Layer 2 marking of Ethernet packets in the egress direction.

## QoS L2 Re-marking of Ethernet Packets in Egress Direction

The router supports Layer 2 marking of Ethernet packets in the egress direction.

To enable this feature, you must:

- Configure the policy maps for queuing and marking at the egress interface.
- Set traffic-class in the ingress and use **match traffic-class** in the egress for queuing.
- Ensure that the **set qos-group** command is configured in ingress policy and the corresponding **match qos-group** command is configured in the egress marking policy. If there is no corresponding QoS group, you will experience traffic failure.

  The ingress ‘push VLAN’ is translated to ‘pop VLAN’ for the egress traffic. In this case, (CoS, DEI) re-marking is not supported for the VLAN tag. For example:

  1. rewrite ingress tag push dot1q/dot1ad <> symmetric
  2. rewrite ingress tag push dot1q/dot1ad <> second-dot1q <> symmetric
  3. rewrite ingress tag translate 1-to-2 dot1q/dot1ad <> second-dot1q <> symmetric

#### Running Configuration

```
policy-map egress-marking
class qos1
set cos 1
!
class qos2
set cos 2
set dei 1
!
class qos3
set cos 3
!
class class-default
set cos 7
!
```

```
end-policy-map
!
```

## QoS L2 Re-Marking of Ethernet Packets on L3 Flows in Egress Direction

The router supports Layer 2 marking of Ethernet packets on Layer 3 flows in the egress direction.

To enable this feature, you must:

- Configure the policy maps for marking at the egress interface.
- Ensure that the **set qos-group** command is configured in ingress policy and the corresponding **match qos-group** command is configured in the egress marking policy. If there is no corresponding QoS group, you will experience traffic failure.

### Restrictions

The following restrictions apply while configuring the Layer 2 marking of Ethernet packets on Layer 3 flows in the egress direction.

- **set discard-class** is not supported in ingress policy with peering mode.
- Egress marking statistics are not available.
- Layer 2 (802.1p) Egress marking is supported on Layer 3 flows for these types of traffic: IP-to-IP, IP-to-MPLS, and MPLS-to-IP traffic.
- Layer 2 marking of Ethernet packets on Layer 3 flows in the egress direction is supported only in the peering mode.

### Running Configuration

Ingress Policy:

You must first set up the qos-group at ingress.

```
class-map match-any Class0
 match mpls experimental topmost 0
 match precedence routine
 match dscp 0-7
 end-class-map
class-map match-any Class1
 match mpls experimental topmost 1
 match precedence priority
 match dscp 8-15
 end-class-map
class-map match-any Class2
 match mpls experimental topmost 2
 match precedence immediate
 match dscp 16-23
 end-class-map
class-map match-any Class3
 match mpls experimental topmost 3
 match precedence flash
 match dscp 24-31
 end-class-map
class-map match-any Class4
 match mpls experimental topmost 4
 match precedence flash-override
 match dscp 32-39
```

```
 end-class-map
class-map match-any Class5
 match mpls experimental topmost 5
 match precedence critical
 match dscp 40-47
 end-class-map
class-map match-any Class6
 match mpls experimental topmost 6
 match precedence internet
 match dscp 48-55
 end-class-map
class-map match-any Class7
 match mpls experimental topmost 7
 match precedence network
 match dscp 56-63
 end-class-map
!

policy-map ncs_input
 class Class7
  set traffic-class 7
  set qos-group 7
!
 class Class6
  set traffic-class 6
  set qos-group 6
!
 class Class5
  set traffic-class 5
  set qos-group 5
!
 class Class4
  set traffic-class 4
  set qos-group 4
 !
 class Class3
  set traffic-class 4
  set qos-group 3
 !
 class Class2
  set traffic-class 2
  set qos-group 2
 !
 class Class1
  set traffic-class 2
  set qos-group 1
 !
 class Class0
  set traffic-class 0
  set qos-group 0
 !
 end-policy-map
!
```

Egress Policy:

At the egress, run these commands to mark the packets.

```
class-map match-any qos7
match qos-group 7
 end-class-map
!
class-map match-any qos6
match qos-group 6
 end-class-map
```

```
!
class-map match-any qos5
match qos-group 5
 end-class-map
!
class-map match-any qos4
match qos-group 4
 end-class-map
!
class-map match-any qos3
match qos-group 3
 end-class-map
!
class-map match-any qos2
 match qos-group 2
 end-class-map
!
class-map match-any qos1
match qos-group 1
 end-class-map
!

policy-map ncs_output
 class qos7
  set cos 7
  set dei 1
!
 class qos6
  set cos 6
  set dei 1
!
 class qos5
  set cos 5
  set dei 1
!
 class qos4
  set cos 4
  set dei 1
!
 class qos3
  set cos 3
  set dei 1
!
 class qos2
  set cos 2
  set dei 1

!
 class qos1
  set cos 1
  set dei 1
!
 end-policy-map
!
```

# QoS L2 Re-Marking of Ethernet Packets on L3 Flows in Egress Direction on L3 sub-interfaces

*Table 6: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Cisco NC57 Native Mode: 802.1p marking | Release 7.4.1 | This feature is now supported on routers that have the Cisco NC57 line cards installed and operate in the native mode. |

The router supports Layer 2 marking of Ethernet packets on Layer 3 flows in the egress direction on L3 subinterfaces.

To enable this feature, you must:

- Configure the policy maps for marking at the egress interface.
- Ensure that the **set qos-group** command is configured in ingress policy and the corresponding **match qos-group** command is configured in the egress marking policy. If there is no corresponding QoS group, you experience traffic failure.

### Restrictions

The following restrictions apply while configuring the Layer 2 marking of Ethernet packets on Layer 3 flows in the egress direction.

- **set discard-class** is not supported in ingress policy with peering mode.
- Egress marking statistics are not available.
- Layer 2 (CoS, DEI) Egress marking is supported on Layer 3 flows on L3 subinterfaces for these types of traffic: IP-to-IP, IP-to-MPLS, and MPLS-to-IP traffic.

### Running Configuration

Ingress Policy:

You must first set up the qos-group at ingress. This is applicable only when you want to mark packets at the egress.

```
class-map match-all COS0_DEI0
 match cos 0
 match dei 0
 end-class-map
class-map match-all COS0_DEI1
 match cos 0
 match dei 1
 end-class-map
class-map match-all COS1_DEI0
 match cos 1
 match dei 0
 end-class-map
class-map match-all COS1_DEI1
 match cos 1
 match dei 1
 end-class-map
class-map match-all COS2_DEI0
```

```
 match cos 2
 match dei 0
 end-class-map
class-map match-all COS2_DEI1
 match cos 2
 match dei 1
 end-class-map
class-map match-all COS3_DEI0
 match cos 3
 match dei 0
 end-class-map
class-map match-all COS3_DEI1
 match cos 3
 match dei 1
 end-class-map
class-map match-all COS4_DEI0
 match cos 4
 match dei 0
 end-class-map
class-map match-all COS4_DEI1
 match cos 4
 match dei 1
 end-class-map
class-map match-all COS5_DEI0
 match cos 5
 match dei 0
 end-class-map
class-map match-all COS5_DEI1
 match cos 5
 match dei 1
 end-class-map
class-map match-all COS6_DEI0
 match cos 6
 match dei 0
 end-class-map
class-map match-all COS6_DEI1
 match cos 6
 match dei 1
 end-class-map
class-map match-all COS7_DEI0
 match cos 7
 match dei 0
 end-class-map
class-map match-all COS7_DEI1
 match cos 7
 match dei 1
 end-class-map

policy-map ncs_input
 class COS7_DEI0
  set qos-group 7
  set discard-class 0
!
 class COS7_DEI1
  set qos-group 7
  set discard-class 1
!
 class COS6_DEI0
  set qos-group 6
  set discard-class 0
!
 class COS6_DEI1
  set qos-group 6
  set discard-class 1
```

```
!
 class COS5_DEI0
  set qos-group 5
  set discard-class 0
!
 class COS5_DEI1
  set qos-group 5
  set discard-class 1
!
 class COS4_DEI0
  set qos-group 4
  set discard-class 0
!
 class COS4_DEI1
  set qos-group 4
  set discard-class 1
!
 class COS3_DEI0
  set qos-group 3
  set discard-class 0
!
 class COS3_DEI1
  set qos-group 3
  set discard-class 1
!
 class COS2_DEI0
  set qos-group 2
  set discard-class 0
!
 class COS2_DEI1
  set qos-group 2
  set discard-class 1
!
 class COS1_DEI0
  set qos-group 1
  set discard-class 0
!
 class COS1_DEI1
  set qos-group 1
  set discard-class 1
!
 class COS0_DEI0
  set qos-group 0
  set discard-class 0
!
 class COS0_DEI1
  set qos-group 0
  set discard-class 1
!
```

Egress Policy:

At the egress, run these commands to mark the packets.

```
class-map match-all qos7_dc0
match qos-group 7
match discard-class 0
 end-class-map
!
class-map match-all qos7_dc1
match qos-group 7
match discard-class 1
 end-class-map
!
```

```
class-map match-all qos6_dc0
match qos-group 6
match discard-class 0
 end-class-map
!
class-map match-all qos6_dc1
match qos-group 6
match discard-class 1
 end-class-map
!
class-map match-all qos5_dc0
match qos-group 5
match discard-class 0
 end-class-map
!
class-map match-all qos5_dc1
match qos-group 5
match discard-class 1
 end-class-map
!
class-map match-all qos4_dc0
match qos-group 4
match discard-class 0
 end-class-map
!
class-map match-all qos4_dc1
match qos-group 4
match discard-class 1
 end-class-map
!
class-map match-all qos3_dc0
match qos-group 3
match discard-class 0
 end-class-map
!
class-map match-all qos3_dc1
match qos-group 3
match discard-class 1
 end-class-map
!
class-map match-all qos2_dc0
match qos-group 2
match discard-class 0
 end-class-map
!
class-map match-all qos2_dc1
match qos-group 2
match discard-class 1
 end-class-map
!
class-map match-all qos1_dc0
match qos-group 1
match discard-class 0
 end-class-map
!
class-map match-all qos1_dc1
match qos-group 1
match discard-class 1
 end-class-map
!
class-map match-all qos0_dc0
match qos-group 0
match discard-class 0
 end-class-map
```

```
!
class-map match-all qos0_dc1
match qos-group 0
match discard-class 1
 end-class-map
!


policy-map ncs_output
 class qos7_dc0
  set cos 7
  set dei 0
  set mpls experimental imposition 7
!
 class qos7_dc1
  set cos 7
  set dei 1
  set mpls experimental imposition 7
!
 class qos6_dc0
  set cos 6
  set dei 0
  set mpls experimental imposition 6
!
 class qos6_dc1
  set cos 6
  set dei 1
  set mpls experimental imposition 6
!
 class qos5_dc0
  set cos 5
  set dei 0
  set mpls experimental imposition 5
!
 class qos5_dc1
  set cos 5
  set dei 1
  set mpls experimental imposition 5
!
 class qos4_dc0
  set cos 4
  set dei 0
  set mpls experimental imposition 4
!
 class qos4_dc1
  set cos 4
  set dei 1
  set mpls experimental imposition 4
!
 class qos3_dc0
  set cos 3
  set dei 0
  set mpls experimental imposition 3
!
 class qos3_dc1
  set cos 3
  set dei 1
  set mpls experimental imposition 3
!
 class qos2_dc0
  set cos 2
  set dei 0
  set mpls experimental imposition 2
!
```

```
 class qos2_dc1
  set cos 2
  set dei 1
  set mpls experimental imposition 2
!
 class qos1_dc0
  set cos 1
  set dei 0
  set mpls experimental imposition 1
!
 class qos1_dc1
  set cos 1
  set dei 1
  set mpls experimental imposition 1
!
 class qos0_dc0
  set cos 0
  set dei 0
  set mpls experimental imposition 0
!
 class qos0_dc1
  set cos 0
  set dei 1
  set mpls experimental imposition 0
!
 end-policy-map
!
```

# Layer 2 Ingress QoS Matching for IPv4 and IPv6 Destination Addresses

*Table 7: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Layer 2 Ingress QoS Matching for IPv4 and IPv6 Destination Addresses | Release 7.5.1 | Using this feature, you can match class maps to IPv4 and IPv6 destination addresses on Layer 2 networks. The Layer 2 interface service policy has the relevant class maps, actioning them for ingress QoS operations.<br><br>This feature provides you with an additional level of classification for aggregated customer traffic at your ingress, thus giving you granular control on traffic flows.<br><br>This feature introduces the following commands:<br><br>• **match destination-address**<br><br>• **hw-module profile qos l2-match-dest-addr-v4v6** |

## Overview

As a service provider, you provide Layer 2 connectivity for different classes of customer traffic across your network. With aggregated customer traffic arriving at your ingress, you need to provide differential treatment depending on specific destination addresses for the traffic. Such ability gives you granular control over traffic, allowing you to classify specific traffic flows depending on the type of services for which your customers have signed up.

You can match class maps to IPv4 and IPv6 destination addresses on Layer 2 networks to ensure such granular control. The interface service policy has the relevant class maps, actioning them for ingress QoS marking.

## Guidelines and Limitations

- You can match up to 4 IPv4 and IPv6 addresses each in a class.
- For match on IPv6, only up to 64-bit prefix match is supported.
- The L2VPN traffic can be Virtual Private Wire Service (VPWS) or Virtual Private LAN Service (VPLS).
- Redundant and non-redundant pseudowires are supported.
- This feature isn’t supported with egress ACL enabled.
- This feature isn't supported on NC57 line cards.
- Traffic classification for VLAN tags is supported as shown in the following table.

| VLAN Tag Condition | IPv4 Addresses | IPv6 Addresses | Combination of IPv4 and IPv6 Addresses |
|---|---|---|---|
| With no VLAN tags | ☑ | ☑ | ☑ |
| With a single VLAN tag | ☑ | ☑ | ☑ |
| With a double VLAN tag | ☑ | ☑ | ☑ |

## Configure Layer 2 Ingress QoS Matching for IPv4 and IPv6 Destination Addresses

Perform the following steps to configure Layer 2 ingress QoS matching for IPv4 and IPv6 destination addresses. This example covers:

- **match-all** criteria for an IPv4 address and a Layer 2 classification (**match dscp**) in the same class map.

**Note** You can use the **match-all** criteria only when you want to match one specific destination address with other Layer 2 classification options such (CoS, DEI) or DSCP.

- **match-any** criteria for IPv4 and IPv6 addresses in the same class map.

1. Enable the ability to match class maps to IPv4 and IPv6 destination addresses on Layer 2 networks. Reload the router for the **hw-module** command to be functional.
2. Create a class map and specify **match-all** criteria for an IPv4 address and DSCP.

3. Create a class map and specify **match-any** criteria for IPv4 and IPv6 addresses.
4. Create a policy map and associate the class maps you created with the traffic policy and specify **class-action**.
5. Attach the policy map to the interface.

### Configuration

```
/*Enable the ability to match class maps to IPv4 and IPv6 destination addresses on Layer 2
 networks*/
Router(config)#hw-module profile qos l2-match-dest-addr-v4v6
Router(config)#commit
Router#reload

/*Create a class map and specify match-all criteria for an IPv4 address and DSCP*/
Router(config)#class-map match-all ipv4_dst_cs1
Router(config-cmap)#match destination-address ipv4 192.168.1.4 255.255.255.255
Router(config-cmap)#match dscp cs1

/*Create a class map and specify match-any criteria for IPv4 and IPv6 addresses*/
Router(config-cmap)#class-map match-any V4_V6_MATCH
Router(config-cmap)#match destination-address ipv4 10.0.0.0 255.0.0.0
Router(config-cmap)#match destination-address ipv4 20.1.0.0 255.255.0.0
Router(config-cmap)#match destination-address ipv4 20.1.1.1 255.255.255.255
Router(config-cmap)#match destination-address ipv4 30.1.0.1 255.255.255.0
Router(config-cmap)#match destination-address ipv6 101:1:12::1/64
Router(config-cmap)#match destination-address ipv6 201:1:1::1/32
Router(config-cmap)#match destination-address ipv6 201:1:3::2/64
Router(config-cmap)#match destination-address ipv6 301:1:3::2/64
Router(config-cmap)#commit

/*Create a policy map, associate the class maps with the traffic policy; specify class-action:
 police rate, in this example*/
Router(config-cmap)#policy-map PMAP_L2_V4_V6_MATCH
Router(config-pmap)#class ipv4_dst_cs1
Router(config-pmap-c)#police rate 10 mbps
Router(config-pmap-c-police)#class V4_V6_MATCH
Router(config-pmap-c)#police rate 10 mbps
Router(config-pmap-c-police)#commit

/*Attach the policy map with class-actions that you set in the class maps*/
Router(config-pmap-c-police)#int Bundle-Ether100.2
Router(config-if)#service-policy input PMAP_L2_V4_V6_MATCH
Router(config-if)#commit
```

You have successfully configured Layer 2 ingress QoS matching for IPv4 and IPv6 destination addresses.

### Running Configuration

```
qos l2-match-dest-addr-v4v6
!

class-map match-all ipv4_dst_cs1
 match destination-address ipv4 192.168.1.4 255.255.255.255
 match dscp cs1

!
 class-map match-any V4_V6_MATCH
  match destination-address ipv4 10.0.0.0 255.0.0.0
  match destination-address ipv4 20.1.0.0 255.255.0.0
  match destination-address ipv4 20.1.1.1 255.255.255.255
```

```
  match destination-address ipv4 30.1.0.1 255.255.255.0
  match destination-address ipv6 101:1:12::1/64
  match destination-address ipv6 201:1:1::1/32
  match destination-address ipv6 201:1:3::2/64
  match destination-address ipv6 301:1:3::2/64

!
!

  policy-map PMAP_L2_V4_V6_MATCH
   class ipv4_dst_cs1
    police rate 10 mbps
     class V4_V6_MATCH
    police rate 10 mbps
!
!
```

### Verification

To verify that the configuration was successful, run the **sh policy-map pmap-name** command for the policy map you created with all class maps associated. The output displays all the **match-any** and **match-all** configurations for IPv4 and IPv6 addresses.

```
Router#sh policy-map pmap-name PMAP_L2_V4_V6_MATCH detail
class-map match-all ipv4_dst_cs1
 match destination-address ipv4 192.168.1.4 255.255.255.255
 match dscp cs1
 end-class-map

!
class-map match-any V4_V6_MATCH
 match destination-address ipv4 10.0.0.0 255.0.0.0
 match destination-address ipv4 20.1.0.0 255.255.0.0
 match destination-address ipv4 20.1.1.1 255.255.255.255
 match destination-address ipv4 30.1.0.1 255.255.255.0
 match destination-address ipv6 101:1:12::1/64
 match destination-address ipv6 201:1:1::1/32
 match destination-address ipv6 201:1:3::2/64
 match destination-address ipv6 301:1:3::2/64
 end-class-map
!
policy-map PMAP_L2_V4_V6_MATCH
 class ipv4_dst_cs1
  police rate 10 mbps
  !

 !
 class V4_V6_MATCH
  police rate 10 mbps
  !
 !
 class class-default
 !
 end-policy-map
!
!

Router#sh run interface bundle-ether 100
interface Bundle-Ether100
service-policy input ipv4_dst_cs1
 ipv4 address 192.168.1.4 255.255.255.255
```

```
 dscp cs1
 service-policy input PMAP_L2_V4_V6_MATCH
 ipv4 address 10.1.0.1 255.255.255.0
 ipv6 address 10:1::1/96
!
```

# Bundle Traffic Policies

A policy can be bound to bundles. When a policy is bound to a bundle, the same policy is programmed on every bundle member (port). For example, if there is a policer or shaper rate, the same rate is configured on every port. Traffic is scheduled to bundle members based on the load balancing algorithm.

Both ingress and egress traffic is supported. Percentage-based policies , absolute rate-based policies, and time-based policies are supported.

**Note** Egress marking is not supported on BVI interfaces.

For details, see Configure QoS on Link Bundles, on page 146.

# Shared Policy Instance

***Table 8: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Shared Policy Instance | Release 7.3.1 | This feature allows you to share a single instance of QoS policy across multiple subinterfaces, allowing for aggregate shaping of the subinterfaces to one rate. The ability to facilitate queue consumption in this manner offers the advantage of saving on QoS and hardware resources, while ensuring that the specified rate is not exceeded. |
| Shared Policy Instance | Release 7.4.1 | This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes. |

Traditionally, when services required by your end-customers mapped one-on-one to an interface, attaching the QoS policy-map directly to the interface was the way to meet customer SLAs. However, with increasing demand for triple play configurations—requiring the management of voice and video queues in addition to data queues —you may have several forwarding constructs. This scenario calls for the need to apply an aggregate QoS policy across interfaces to provide the necessary traffic.

After you create the traffic class and traffic policy, you can optionally use a shared policy instance to allocate a single set of QoS resources and share them across a group of subinterfaces.

With shared policy instance, you can share a single instance of a QoS policy across multiple subinterfaces, allowing for aggregate shaping, policing, and marking of the subinterfaces to one rate. All the subinterfaces that share the instance of a QoS policy must belong to the same main interface. The number of subinterfaces that share the QoS policy instance can range from 2 to the maximum number of subinterfaces on the main interface.

When a shared policy instance of a policy map is shared by several subinterfaces, QoS operations such as aggregate shaping, policing, and marking are applied for traffic on all the interfaces that use the same shared policy instance.

Traditionally, policies were bound to interfaces. However, different types of interfaces, such as Layer 2 and Layer 3, can use a single shared-policy-instance, which allows flexibility in the "attachment point" that binds the policy map.

As an example, consider the following policy configuration:

```
policy-map hqos_gold
 class class-default
  service-policy child_hqos_gold
  shape average 20 mbps
 !
 end-policy-map
!
policy-map child_hqos_gold
 class voice
  priority level 1
  shape average 64 kbps
 !
 class video
  priority level 1
  shape average 4 mbps
 !
 class data
  bandwidth 5 mbps
 !
 class class-default
 !
 end-policy-map
!

interface TenGigE 0/1/0/10.300 l2transport
 service-policy output hqos_gold shared-policy-instance hqos_gold_customer1
!
interface TenGigE 0/1/0/10.400 l2transport
 service-policy output hqos_gold shared-policy-instance hqos_gold_customer1
!
```

The keyword **shared-policy-instance** and the instance name **hqos_gold_customer1** identify the subinterfaces that share an aggregate SLA. These are shared on a physical main interface or a bundle member. In other words, in a mix of Layer 2 and Layer 3 subinterfaces in the same shared policy instance, both layers support classification criteria and action.

In the case of bundles, sharing is applicable within a bundle member and not the entire bundle. Depending on the traffic hashing, shared policy instance may or may not take effect across the subinterface under the bundle main interface.

All subinterfaces that share the same shared policy instance share resources as well. Hence, the **show policy-map** statistics values and **show qos** values for all the subinterfaces are the same.

## Restrictions and Guidelines

The following restrictions and guidelines apply while configuring shared policy instance for a policy map.

- Subinterfaces that are part of the same shared policy must belong to the same main interface. In other words, subinterfaces of different main interfaces cannot be part of the same shared policy.
- There is no restriction on the number of unique shared policies across a system. However, the limit of maximum number of subinterfaces with QoS policies applies.
- There is no restriction on the number of unique shared policies per main interface, port, core, NPU, or line card.
- You cannot use the same shared policy name on the ingress and egress of the same subinterface.
- Shared policy instance is not supported with multi-policies. For example, on the egress, you cannot apply a marking policy and a queueing policy under a shared policy.
- A shared policy can include a combination of Layer 2 and Layer 3 subinterfaces.

## Attaching a Shared Policy Instance to Multiple Subinterfaces

To attach a shared policy instance to multiple subinterfaces:

1. Enter interface configuration mode and configure a subinterface.
2. Attach a policy map to an input or output subinterface for it to be the service policy for that subinterface.

```
RP/0/RP0/CPU0:router(config)#interface HundredGigE0/3/0/0.1
RP/0/RP0/CPU0:router(config-subif)#service-policy output pm-out shared-policy-instance spi1
```

### Running Configuration

```
interface HundredGigE0/3/0/0.1
service-policy output pm-out shared-policy-instance spi1
ipv4 address 20.0.0.1 255.255.255.0
encapsulation dot1q 1
!
```

### Verification

The show policy-map shared-policy-instance command includes an option to display counters for the shared policy instance.

**Note**

- For bundle subinterfaces, use RP as the location keyword.
- For physical subinterfaces, use LC as the location keyword.

For example, for a physical interface:

```
RP/0/RP0/CPU0:ios#show policy-map shared-policy-instance spi1 output location 0/3/CPU0

Shared Policy Instance spi1 output: pm-out

Class cm-tc-1
```

```
  Classification statistics          (packets/bytes)               (rate - kbps)
    Matched             :           772637560/1143503679080        9622860
    Transmitted         :           731260312/1082265352040        5052880
    Total Dropped       :            41377248/61238327040          4569980
  Queueing statistics
    Queue ID                              : 1433
    Taildropped(packets/bytes)            : 41377248/61238327040
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                              : 1432
    Taildropped(packets/bytes)            : 0/0
Policy Bag Stats time: 1604675533816  [Local Time: 11/06/20 15:12:13.816]
```

Use the **clear qos counters shared-policy-instance** command to clear counters for the shared policy instance.

**Note**

- For bundle subinterfaces, use RP as the location keyword.
- For physical subinterfaces, use LC as the location keyword.

For example, for a physical interface:

```
RP/0/RP0/CPU0:ios#clear qos counters shared-policy-instance spi1 output location 0/3/CPU0
```

The **show qos shared-policy-instance** command allows you to display the QoS hardware programming values.

**Note**

- For bundle subinterfaces, use RP as the location keyword.
- For physical subinterfaces, use LC as the location keyword.

For example, for a physical interface:

```
RP/0/RP0/CPU0:ios#show qos shared-policy-instance spi1 output location 0/3/CPU0
Fri Nov  6 15:21:44.200 UTC
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/3/0/0.1 ifh 0x60040c8  -- output policy
NPU Id:                        0
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Policy Name:                   pm-out
SPI Id:                        0x3000001
VOQ Base:                      1432
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   cm-tc-1
Egressq Queue ID                          =   1433 (LP queue)
Queue Max. BW.                            =   5118857 kbps (5 %)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   5000000 kbps
Peak burst                                =   33600 bytes (default)
TailDrop Threshold                        =   6258688 bytes / 10 ms (default)
WRED not configured for this class
```

```
Level1 Class                              =   class-default
Egressq Queue ID                          =   1432 (Default LP queue)
Queue Max. BW.                            =   no max (default)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   50000000 kbps
Peak burst                                =   33600 bytes (default)
TailDrop Threshold                        =   62652416 bytes / 10 ms (default)
WRED not configured for this class
```

# Ingress Short-Pipe

When QoS traffic leaves an MPLS network, the MPLS label stack is removed on the penultimate ingress Label Switch Router (LSR), leaving an IPv4 or IPv6 packet to be forwarded. MPLS experimental bits (or EXP or pipe mode) carries out this disposition process and the packet is marked with a Differentiated Services Code Point (DSCP) or precedence value (also called DSCP or Precedence-based classification).

Usually, QoS traffic supports DSCP and precedence-based classifications only when there is no MPLS label in the packet. Using the ingress short-pipe feature, however, you can classify a packet that contains one MPLS label using the type-of-service (ToS) field of the IPv4 or IPv6 header. This classification method is called ingress short-pipe. To classify an IP packet this way, you must:

1. Create a child class map.
2. Specify a ToS value in the child class map.
3. Attach the child class map to a parent class map.
4. Create a policy map containing the parent class map.
5. Set any ingress action such as traffic class or QoS group. From Release 7.1.1 onwards, you can also set ingress action DSCP (or precedence value).

With the ingress short-pipe feature, you get an increased visibility into traffic packets. Plus, the feature also removes the limitation of classifying MPLS packets that come into IPv4 or IPv6 networks.

## Restrictions and Other Important Points

Ensure that you read these points before you configure the ingress short-pipe feature.

- This feature isn’t supported on:
  - NC57-24DD
  - NC57-18DD-SE
  - NC57-36H-SE
  - NC57-36H6D-S
  - NC57-MOD-S
  - NCS-57B1-6D24-SYS
  - NCS-57B1-5DSE-SYS

  - NCS-57C3-MOD-SYS
  - NCS-57D2-18DD-SYS

- This feature works only when there is one MPLS header in the traffic packet. If there are two or more MPLS headers, the ingress-short pipe feature fails. For example, in case of Explicit Null where there are two labels at the disposition, this feature will not work.
- You can carry out ingress classification using either the MPLS experimental bits (or EXP or pipe mode) classification OR the DSCP/precedence (or short-pipe) classification. Ensure that you do not mix the classification methods, else it may result in an unknown behavior, and the classification may not work at all.
- This feature is supported only on L3VPN, and not supported on L2VPN.
- This feature works for regular IPv4/IPv6 traffic, but will not work for IPv6 VPN Provider Edge over MPLS (6VPE).
- You can add only one child class map to a parent class map.
- This feature supports the invocation of short-pipe and legacy DSCP classification for the same parent class map.
- The child class map can contain only match precedence and match dscp commands.
- This feature is not supported in peering mode.

# Configure Ingress Short-Pipe

This section details a sample configuration for the ingress short-pipe feature and another sample to configure classification for labeled and non-labeled packets under the same parent class.

**Sample configuration to classify a packet that contains one MPLS label using the type-of-service (ToS) field of the IPv4 or IPv6 header (or the ingress short-pipe method):**

```
class-map match-any in_pipe
 match mpls disposition class-map child_pipe
end-class-map
!
class-map match-any child_pipe
 match precedence 1
match dscp ipv4 af11
 end-class-map
!
class-map match-any ingress-business-high
match dscp af21 af22
end-class-map

class-map match-any ingress-business-low
match dscp af11 af12
end-class-map

policy-map ingress-classifier
class in_pipe
set traffic-class 5
set dscp af31
class ingress-business-high
set traffic-class 4
class ingress-business-low
```

```
set traffic-class 2
class class-default
set traffic-class 0
!
```

**Note** The **set dscp** option is available from Release 7.1.1 onwards.

You can configure classification for both labeled and non-labeled packets under the same parent class as in the following sample configuration. In this example, for MPLS labeled packets, DSCP configured under the child class is classified, while for non-labeled packets, DSCP/ToS configured in the **match dscp <value>** statement is classified.

DSCP value range is from 0 through 63. The range option is not supported. Up to 8 items per class are supported. Up to 64 **match dscp** values in total.

```
class-map match-any in_pipe
match mpls disposition class-map child_pipe  (labeled case)
match dscp af11 (non-labeled case)
end-class-map
!
class-map match-any child_pipe
match precedence 1
match dscp ipv4 af11
end-class-map
!
class-map match-any ingress-business-high
match dscp af21 af22
end-class-map

class-map match-any ingress-business-low
match dscp af11 af12
end-class-map

policy-map ingress-classifier
class in_pipe
set traffic-class 5

class ingress-business-high
set traffic-class 4
class ingress-business-low
set traffic-class 2
class class-default
set traffic-class 0
!
```

### Associated Commands

- match mpls disposition class-map

# Selective Egress Policy-Based Queue Mapping

With selective egress policy-based queue mapping, you can combine traffic class (TC) maps in various permutations at the egress.

**Note** Modular chassis do not support this feature.

The primary aim of introducing the egress TC (traffic class) mapping is to classify the traffic in the ingress using a single policy and place the classified traffic into queues, by assigning the traffic classes. At the egress, you can support different groupings of TCs.

Based on different Service Level Agreements (SLAs) that each customer has signed up for, you can group some TCs into priority queues for real time (RT) traffic, other TCs into guaranteed bandwidth (BW) traffic, and the rest into best effort (BE) traffic delivery.

Let us consider an example where three customers have purchased these services, based on their requirements:

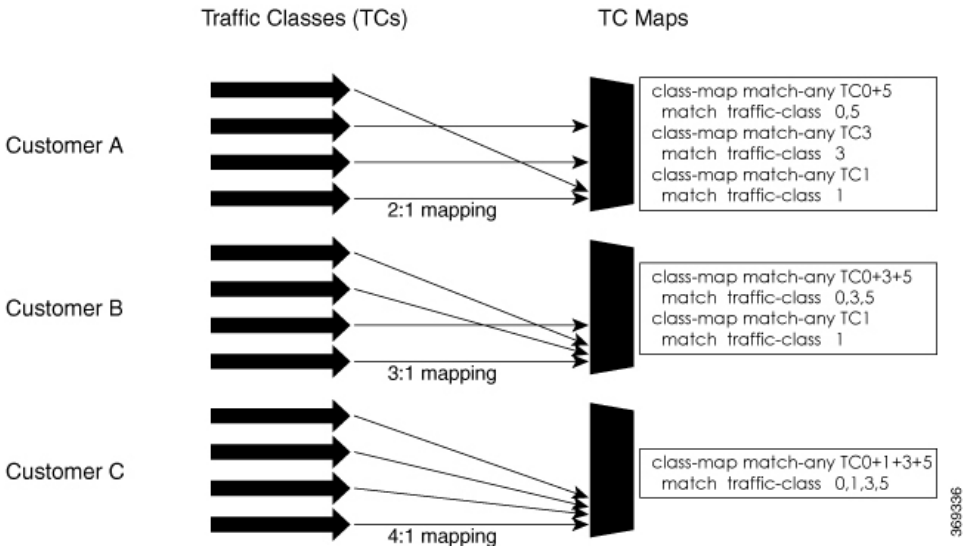
- Customer A - Requires RT traffic, reserved BW traffic and BE traffic delivery.
- Customer B – Requires reserved BW traffic and BE traffic delivery.
- Customer C – Needs only BE traffic delivery.

Using the selective egress policy-based queue mapping, you can create three profiles this way:

- Customer A – Priority queue RT traffic (TC1), Guaranteed BW traffic (TC3), Best effort traffic (TC0, TC5)
- Customer B – Guaranteed BW traffic (TC1), Best effort traffic (TC0, TC3, TC5)
- Customer C - Best effort traffic (TC0, TC1, TC3, TC5)

Using the egress TC-mapping, you can create three different profiles that you can use for each customer based on their SLAs with the provider.

***Figure 1: Selective Egress Policy-Based Queue Mapping Helps Create Customer Profiles Based on Their SLAs***



# Restrictions and Other Important Points

- Ensure that you read these points before you configure the selective egress policy-based queue-mapping feature.
  - There can be only one TC (Traffic Class) mapped class to a PM (Policy Map).
  - You cannot use a TC that you used in a mapped class, in a non-mapped class under the same PM.
  - You can have a maximum of three unique TC mapped PMs or profiles per platform.
  - Every TC mapped class must include **traffic-class 0** in the range values.
  - The TC-mapping range is from 0 through 5.
  - When a TC-mapped class is present in a PM, the class default becomes a dummy class. This means that the class default statistics and QoS values are not applicable.
  - All the class default limitations apply to the TC-mapped class; for example, you cannot configure **priority** command under the TC mapped class.

**Note** A TC-mapped PM or profile is a PM that contains a TC-mapped class.

Example of a TC-mapped class:

**match traffic-class 0 1 2 3**

Example of a TC non-mapped class:

**match traffic-class 1**

# Configure Selective Egress Policy-Based Queue Mapping

This section details a sample configuration for the selective egress policy-based queue-mapping feature and a use case to show how this feature works.

**Sample configuration**

```
class-map match-any <name>
 match traffic-class <value>
commit

policy-map tc_pmap
 class tc035
  shape average percent 1
 !
 class class-default
!
 end-policy-map
!
 class-map match-any tc035
match traffic-class 0 3 5
 end-class-map
!
```

### Verification

Run the **show qos interface** and **show policy-map interface** commands.

When TC mapping class is present in a policy map, the class default does not have any values calculated.

**show qos interface** bundle-Ether 44 output sample

```
NOTE:- Configured values are displayed within parentheses
NPU Id:                        0
Total number of classes:       3
Interface Bandwidth:           100000000 kbps
Policy Name:                   tc_pmap
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   tc1

Level1 Class                             =   tc035

Level1 Class                             =   class-default

Interface HundredGigE0/0/0/30 Ifh 0xf000208 (Member) -- output policy
NPU Id:                        0
Total number of classes:       3
Interface Bandwidth:           100000000 kbps
Policy Name:                   tc_pmap
VOQ Base:                      1264
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   tc1
Egressq Queue ID                         =   1265 (LP queue)
Queue Max. BW.                           =   10063882 kbps (10 %)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Guaranteed service rate                  =   10000000 kbps
TailDrop Threshold                       =   12517376 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                             =   tc035
Egressq Queue ID                         =   1264 (LP queue)
Queue Max. BW.                           =   1011732 kbps (1 %)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Guaranteed service rate                  =   1000000 kbps
TailDrop Threshold                       =   1253376 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                             =   class-default
Queue Max. BW.                           =   no max (default)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   0 / (BWR not configured)
```

**show policy-map interface** bundle-Ether 44 output sample

```
Bundle-Ether44 output: tc_pmap

Class tc1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :              429444/53823648             0
    Transmitted         :              429444/53823648             0
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                            : None (Bundle)
    Taildropped(packets/bytes)          : 0/0
Class tc035
```

```
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :             1288331/161470820            0
    Transmitted         :             1288331/161470820            0
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                              : None (Bundle)
    Taildropped(packets/bytes)            : 0/0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                              : None (Bundle)
    Taildropped(packets/bytes)            : 0/0
Policy Bag Stats time: 1557216940000  [Local Time: 05/07/19 08:15:40.000]
RP/0/RP0/CPU0:BB1#
```

### Use Case

With the ingress traffic matching the same match criteria, you can group the egress traffic up to three unique TC mapped profiles. Using this feature, you can provide differentiated services to customers based on the SLAs they have signed up for.

In the example that follows, the ingress policy-map sets the ingress match criteria for the traffic class from 0 through 5. Based on the SLAs, you can group the TC values at the egress PM to deliver differentiated services.

After you group the TC values, you can apply specific egress actions under that class.

**Ingress match:**

```
class EXP1
  set traffic-class 1
!
class EXP2
  set traffic-class 2
!
class EXP3
  set traffic-class 3
!
class EXP4
  set traffic-class 4
!
class EXP5
  set traffic-class 5
!
class class-default
!
end-policy-map
!
```

**Egress match:**

**Sample TC mapped class for policy-map PM1**

```
class-map match-any TC2:1
match traffic-class 0 1
end-class-map
```

Sample TC mapped class for policy-map PM2

```
class-map match-any TC3:1
match traffic-class 0 1 2
end-class-map
```

Sample TC mapped class for policy-map PM3

```
class-map match-any TC6:1
match traffic-class 0 1 2 3 4 5
end-class-map
```

# Configuring QoS Groups with an ACL

You can create QoS groups and configure ACLs to classify traffic into the groups based on a specified match condition. In this example, we match by the QoS group value (0-511).

### Supported ACL Types

Your router supports the following ACL types.

**Note** If you configure QoS group with an unsupported ACL type, the system doesn't display any error message.

| ACL Type | Example |
|---|---|
| IPv4 DSCP | permit ipv4 any any dscp af43 |
| UDP DSCP | permit udp any any dscp af43 |
| UDP Fragments IPv4 | udp any any fragments - IPv4 |
| UDP Fragments IPv6 | udp any any fragments - IPv6 |
| TCP Fragments IPv4 | tcp any any fragments - IPv4 |
| TCP Fragments IPv6 | tcp any any fragments - IPv6 |
| IPV4 DSCP Fragments | permit ipv4 any any dscp af43 fragments |
| UDP DSCP Fragments | permit udp any any dscp af43 fragments |
| UDP Host Fragments | permit udp host <sip> host <dip> fragments |
| TCP Host Fragments | permit tcp host <sip> host <dip> dscp af43 fragments |
| TCP DSCP | permit tcp <source network > <destination network> dscp af43 |
| TCP Port based | permit tcp any any eq <port> |
| UDP Port based | permit udp any any eq <port> |
| TCP Flags | permit tcp host <sip> host <dip> established fin psh syn urg |

Restrictions

- ACLs with fragment match are supported on systems with only NC57 line cards, also referred to as native mode.
- IPv6 ACLs with fragment match are supported only in **short** and **short-l2-qos** mode.

### Prerequisites

Before you can configure QoS groups with an ACL, the QoS peering profile must be enabled on the router or the line card. After enabling QoS peering, the router or line card must be reloaded, as shown in the following configuration.

#### Enabling QoS Peering Profile on the Router

Enter the global configuration mode and enable the QoS peering profile for the router as shown:

```
RP/0/RP0/CPU0:router(config)# hw-module profile qos ingress-model peering
RP/0/RP0/CPU0:router(config)# exit
RP/0/RP0/CPU0:router# reload
```

#### Enabling QoS Peering Profile on the Line Card

Enter the global configuration mode and enable the QoS peering profile for the line card as shown:

```
RP/0/RP0/CPU0:router(config)# hw-module profile qos ingress-model peering location 0/0/CPU0
RP/0/RP0/CPU0:router(config)# exit
RP/0/RP0/CPU0:router# reload location 0/0/CPU0
```

### Configuration

Use the following set of configuration statements to configure an ACL with QoS groups.

```
/*
 Enter the global configuration mode, and configure an ACL with the required QoS groups.
*/
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# ipv4 access-list qos-acl
RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 host 5.0.0.1 any set qos-group 1
RP/0/RP0/CPU0:router(config-ipv4-acl)# 11 permit ipv4 host 6.0.0.1 any set qos-group 2
RP/0/RP0/CPU0:router(config-ipv4-acl)# 12 permit ipv4 host 7.0.0.1 any set qos-group 3
RP/0/RP0/CPU0:router(config-ipv4-acl)# 13 deny ipv4 any any


/* Create a policy map with the required classes.
In this example, we also create a default class for traffic that does not belong to any of
 the specified
classes. */
RP/0/RP0/CPU0:router(config)# policy-map qos-acl-map
RP/0/RP0/CPU0:router(config-pmap)# class qos1
RP/0/RP0/CPU0:router(config-pmap-c)# set dscp af43
RP/0/RP0/CPU0:router(config-pmap-c)# set traffic-class 2
RP/0/RP0/CPU0:router(config-pmap-c)# exit

RP/0/RP0/CPU0:router(config-pmap)# class qos2
RP/0/RP0/CPU0:router(config-pmap-c)# set precedence critical
RP/0/RP0/CPU0:router(config-pmap-c)# set traffic-class 7
RP/0/RP0/CPU0:router(config-pmap-c)# exit

RP/0/RP0/CPU0:router(config-pmap)# class qos3
RP/0/RP0/CPU0:router(config-pmap-c)# set precedence 2
```

```
RP/0/RP0/CPU0:router(config-pmap-c)# set traffic-class 2
RP/0/RP0/CPU0:router(config-pmap-c)# exit

RP/0/RP0/CPU0:router(config-pmap)# class qos4
RP/0/RP0/CPU0:router(config-pmap-c)# set traffic-class 4
RP/0/RP0/CPU0:router(config-pmap-c)# set dscp cs4
RP/0/RP0/CPU0:router(config-pmap-c)# exit

RP/0/RP0/CPU0:router(config-pmap)# class class-default
RP/0/RP0/CPU0:router(config-pmap-c)# police rate percent 20
RP/0/RP0/CPU0:router(config-pmap-c-police)# exit


/* Create the class maps for specifying the match conditions. */
RP/0/RP0/CPU0:router(config)# class-map match-any qos1
RP/0/RP0/CPU0:router(config-cmap)# match qos-group 1
RP/0/RP0/CPU0:router(config-cmap)# end-class-map

RP/0/RP0/CPU0:router(config)# class-map match-any qos2
RP/0/RP0/CPU0:router(config-cmap)#  match qos-group 2
RP/0/RP0/CPU0:router(config-cmap)# end-class-map

RP/0/RP0/CPU0:router(config)# class-map match-any qos3
RP/0/RP0/CPU0:router(config-cmap)# match qos-group 3
RP/0/RP0/CPU0:router(config-cmap)# end-class-map

RP/0/RP0/CPU0:router(config)# class-map match-any qos4
RP/0/RP0/CPU0:router(config-cmap)# match qos-group 4
RP/0/RP0/CPU0:router(config-cmap)# end-class-map


/* Apply the access list and the QoS map to the Gigabit interface, and commit your
configuration. */
RP/0/RP0/CPU0:router(config)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-if)# ipv4 address 12.0.0.1/24
RP/0/RP0/CPU0:router(config-if)# no shut
RP/0/RP0/CPU0:router(config-if)# service-policy input qos-acl-map
RP/0/RP0/CPU0:router(config-if)# ipv4 access-group qos-acl ingress compress level 3

RP/0/RP0/CPU0:router(config-if)# commit
Tue Mar 28 10:23:34.106 IST

RP/0/0/CPU0:Mar 28 10:37:48.570 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : Interface
TenGigE0/0/0/1, changed state to Down
RP/0/0/CPU0:Mar 28 10:37:48.608 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : Interface
TenGigE0/0/0/1, changed state to Up

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

### Running Configuration

Confirm your configuration.

```
RP/0/RP0/CPU0:router(config)# show run
Tue Mar 28 10:37:55.737 IST

Building configuration...
!! IOS XR Configuration 0.0.0

ipv4 access-list qos-acl
10 permit ipv4 host 5.0.1.1 any set qos-group 1
11 permit ipv4 host 6.0.1.1 any set qos-group 2
```

```
12 permit ipv4 host 7.0.1.1 any set qos-group 3
13 deny ipv4 any any

class-map match-any qos1
match qos-group 1
end-class-map
!
class-map match-any qos2
match qos-group 2
end-class-map
!
class-map match-any qos3
match qos-group 3
end-class-map
!
class-map match-any qos4
match qos-group 4
end-class-map
!

policy-map qos-acl-map
class qos1
  set dscp af43
  set traffic-class 2
!
class qos2
  set precedence critical
  set traffic-class 7
!
class qos3
  set precedence 2
  set traffic-class 2
!
class qos4
  set traffic-class 4
  set dscp cs4
!
class class-default
  police rate percent 20
  !
!
end-policy-map
!

interface TenGigE0/0/0/1
service-policy input qos-acl-map
ipv4 address 12.0.0.1 255.255.255.0
ipv4 access-group qos-acl ingress compress level 3

!
```

You have successfully configured an ACL with QoS groups.

# Configuring an ACL with Fragment Match

Usually, IP ACLs process non-fragmented packets and the first fragments of a packet using permit and deny actions. These packets may have Layer 3 and 4 information that the ACLs match for a permit or deny action. By default, however, ACLs permit noninitial fragments. This could lead to potential security issues with users with malicious intent using the noninitial fragments to launch denial of service (DoS) attacks.

With this feature, you can now to set QoS policies for noninitial fragment packets, thus having more granular control over noninitial IP fragments of a packet. Noninitial IP fragments have the fragment offset value non-zero. To know more about fragments, see the *IP Addresses and Services Configuration Guide for Cisco NCS 5500 Series Routers*.

# Restrictions and Guidelines

The following restrictions and guidelines apply while configuring an ACL with fragment match.

- To enable IPv6 fragment classification support, configure **hw-module profile qos ipv6 short-l2qos-enable** or **hw-module profile qos ipv6 short**.
- IPv6 fragmentation is supported with only one Extension Header (EH).

# Configuring an ACL with Fragment Match

### For IPv4 and IPv6 ACLs

To configure an ACL with fragment match, you must:

1. Create IPv4 and IPv6 ACLs with fragment match.

**Note** To enable IPv6 fragment classification support, configure **hw-module profile qos ipv6 short-l2qos-enable** or **hw-module profile qos ipv6 short**.

2. Create two class maps, one for IPv4 and IPv6, and attach the respective ACLs to the class maps.
3. Create a policy map with these two class maps and set action.

```
Router(config)#ipv4 access-list v4_ace
Router(config-ipv4-acl)#permit ipv4 any any fragments
Router(config-ipv4-acl)#exit
Router(config)#
Router(config)#ipv6 access-list v6_ace
Router(config-ipv6-acl)#permit ipv6 any any fragments
Router(config-ipv6-acl)#exit
Router(config)#
Router(config)#class-map match-any v4_class
Router(config-cmap)#match access-group ipv4 v4_ace
Router(config-cmap)#exit
Router(config)#
Router(config)#class-map match-any v6_class
Router(config-cmap)#match access-group ipv6 v6_ace
Router(config-cmap)#exit
Router(config)#policy-map frag_policy
Router(config-pmap)#class v4_class
Router(config-pmap-c)#set traffic-class 3
Router(config-pmap-c)#police rate 100 mbps
Router(config-pmap-c-police)#exit
Router(config-pmap-c)#
Router(config-pmap-c)#class v6_class
Router(config-pmap-c)#police rate 150 mbps peak-rate 200 mbps
Router(config-pmap-c-police)#exit
```

```
Router(config-pmap-c)#exit
Router(config-pmap)#exit
```

### Running Configuration

```
ipv4 access-list v4_ace
 permit ipv4 any any fragments
 exit
!
ipv6 access-list v6_ace
 permit ipv6 any any fragments
 exit
!
class-map match-any v4_class
 match access-group ipv4 v4_ace
 exit
!
class-map match-any v6_class
 match access-group ipv6 v6_ace
 exit

policy-map frag_policy
 class v4_class
  set traffic-class 3
  police rate 100 mbps
   exit
!
 class v6_class
  police rate 150 mbps peak-rate 200 mbps
```

### Verification

Run the **show policy-map pmap-name frag_policy detail** command to confirm the ACL fragment matches and the **show qos int hundredGigE 0/5/0/2 input** command to confirm the policer details.

```
Router#show policy-map pmap-name frag_policy detail
ipv4 access-list v4_ace
10 permit ipv4 any any fragments

ipv6 access-list v6_ace
10 permit ipv6 any any fragments

class-map match-any v4_class
match access-group ipv4 v4_ace
 end-class-map
!
class-map match-any v6_class
match access-group ipv6 v6_ace
 end-class-map
!
policy-map frag_policy
class v4_class
  set traffic-class 3
  police rate 100 mbps
  !
 !
 class v6_class
  police rate 150 mbps peak-rate 200 mbps
  !
 !
 class class-default
!
 end-policy-map
```

```
!

Router#show qos int hundredGigE 0/5/0/2 input
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/5/0/2 ifh 0xa000088  -- input policy
NPU Id:                        0
Total number of classes:       3
Interface Bandwidth:           100000000 kbps
Policy Name:                   frag_policy
SPI Id:                        0x0
Accounting Type:               Layer2 (Include Layer 2 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   v4_class
New traffic class                        =   3

Policer Bucket ID                        =   0x12
Policer Stats Handle                     =   0x0
Policer committed rate                   =   99609 kbps (100 mbits/sec)
Policer conform burst                    =   124672 bytes (default)

Level1 Class                             =   v6_class

Policer Bucket ID                        =   0x11
Policer Stats Handle                     =   0x0
Policer committed rate                   =   150390 kbps (150 mbits/sec)
Policer peak rate                        =   200195 kbps (200 mbits/sec)
Policer conform burst                    =   186624 bytes (default)
Policer exceed burst                     =   436096 bytes (default)

Level1 Class                             =   class-default

Default Policer Bucket ID                =   0x10
Default Policer Stats Handle             =   0x0
Policer not configured for this class
```

# Restrictions

Refer to the following table for Ingress QoS Scale limitation.

***Table 9: Ingress QoS Scale Limitation***

| QoS Mode | Class-Map Size | Maximum number of Interfaces with Ingress QoS Applied | |
|---|---|---|---|
| | | Per Core | Per NPU |
| Normal | 4 | 1023 | 2046 |
| Normal | 8 | 511 | 1022 |
| Normal | 16 | 255 | 510 |
| Normal | 32 | 127 | 254 |
| Enhanced | 4 | 871 | 1742 |
| Enhanced | 8 | 435 | 870 |
| Enhanced | 16 | 217 | 434 |

| QoS Mode | Class-Map Size | Maximum number of Interfaces with Ingress QoS Applied | |
|---|---|---|---|
| | | Per Core | Per NPU |
| Enhanced | 32 | 108 | 216 |

**Note** If you apply an ingress policy map to a bundle that has bundle members only from a single core of an NPU, the QoS resources are consumed on both cores of that NPU.

**Example:** For Default Configuration, which is Normal (2 counter mode) QoS Mode & 32 Class Map-Size, you can configure 191 interfaces with Ingress Policy per core.

Other restrictions to follow:

- If you have a **set traffic class** statement explicitly configured in the ingress service policy, it's mandatory to have a corresponding **match traffic class** on egress for the traffic to be correctly matched and the stats to be accounted in **show policy-map interface <> output** command. To match the ingress traffic to the egress class-default, traffic class should be set to 0 on ingress.
- If you have a **set traffic class that is configured** in ingress service policy, and no corresponding **match traffic class** on egress, the traffic won't go to class default and the stats for this traffic flow won't be seen in **show policy-map interface <> output** command.
- If you don't have any **set traffic class** statement in ingress, then traffic will hit the default-class on egress.
- If you have a **set discard-class** statement configured in the ingress service policy, it's mandatory to have a corresponding **match discard-class** on egress for the traffic to be correctly matched and the stats to be accounted in **show policy-map interface <> output** command.
- If you have a **set discard-class** statement configured in the ingress service policy and don't have a corresponding **match discard-class** on egress, the traffic won't hit the class-default and the stats for this flow won't be accounted in **show policy-map interface <> output** command.
- The system doesn't support class-map size on peering mode.
- Even if you have an egress policy that has the drop action configured, the transmitted counter stats still shows an increment.
- Depending on the packet size, the traffic shaped value for low shaper rates, such as 10mbps, have greater deviation than 5% of tolerance from the shaper value. For higher shaper rates, the deviation is within the limit of 5% of tolerance from the shaper value for all packet sizes.
- If the shaper rate is less than 7 Mbps, the calculation of queue-limit is based on 10 ms of guaranteed service rate, and which leads to different queue-limit for each shaper value. This consumes the rate-profile for each queue-limit, and can also lead to a queue-limit of less than 1 MTU causing larger packets to drop.

### Restrictions for Peering QoS Profile

- After enabling the QoS peering feature using the **hw-module profile qos ingress-model peering** command, you can set the Layer 2 class of service (CoS) or drop eligible indicator (DEI) values at the egress using the **set cos** or **set dei** commands, respectively. However, at the egress, ensure that you don't

set the MPLS experimental imposition (EXP) values (using the **set mpls experimental imposition** command). Otherwise, when committing the policy map with these configurations at the egress, you will encounter an error. This error occurs because the internal fields required for egress EXP marking are not available with peering enabled.

- **explicit set discard-class** statement isn't supported.
- This feature is supported only on L3 interfaces and is limited to 1000 L3 interfaces per system.
- **set mpls exp topmost** statement isn't supported within QoS in peering mode.
- **access group** statement isn't supported.
- (Only in Release 6.2.x and Release 6.3.x) **set mpls exp imposition** statement isn't supported on ingress interface.
- 2-Level ingress policers isn't supported.
- (From Release 6.5.x) Egress H-QOS with peering profile support is enabled, but ingress H-QOS with peering profile isn't supported.
- Depending on the packet size, the traffic shaped value for low shaper rates, such as 10mbps, have greater deviation than 5% of tolerance from the shaper value. For higher shaper rates, the deviation is within the limit of 5% of tolerance from the shaper value for all packet sizes.
- (From Release 7.2.1) On NC57 line cards, QoS ingress peering profile isn't supported. This restriction is applicable for systems operating in compatibility mode and native mode.

### Restrictions for QoS on BVI

- The system doesn't support the egress policy on Bridge-Group Virtual Interface (BVI), but BVI (CoS, DEI) marking is supported by applying the policy to its corresponding Layer 2 interface, which is part of the same bridge domain.
- If you apply L3 ingress QoS policy on L2 interface, which is a part of the same bridge-domain as BVI, the classification might not work if packets are destined to the BVI MAC address.
- If a QoS policy is attached to BVI, the policy is inherited by the L2 interfaces, which are part of the same bridge-domain. Hence, any other policy can't be applied on the L2 interfaces. Similarly, if a QoS policy is attached to any of the L2 interfaces, any QoS policy can't be applied on the BVI, which is part of the same bridge-domain.
- In the two-pass forwarding model for packets from Layer 2 to Layer 3 over BVI, where Layer 2 and Layer 3 forwarding are split across two paths and packet processing happens in two cycles, you can apply separate QoS policies for Layer 2 and BVI interfaces.

### Restrictions for Egress Drop Action

- A maximum of 8 interfaces can have the drop action configured and a maximum of 8 classes in any single policy can have the drop action.
- A drop action in any particular class can't be combined with other actions.
- Drop action in a policy applied on the main interface isn't inherited onto sub-interfaces.

- Match condition for drop action PM can only be based on qos-group, discard class based match isn't supported.

# In-Place Policy Modification

The In-Place policy modification feature allows you to modify a QoS policy even when the QoS policy is attached to one or more interfaces. A modified policy is subjected to the same checks that a new policy is subject to when it is bound to an interface. If the policy-modification is successful, the modified policy takes effect on all the interfaces to which the policy is attached. However, if the policy modification fails on any one of the interfaces, an automatic rollback is initiated to ensure that the pre-modification policy is in effect on all the interfaces.

You can also modify any class map used in the policy map. The changes made to the class map take effect on all the interfaces to which the policy is attached.

**Note**

- The QoS statistics for the policy that is attached to an interface are lost (reset to 0) when the policy is modified.
- When a QoS policy attached to an interface is modified, there might not be any policy in effect on the interfaces in which the modified policy is used for a short period of time.
- The system does not support the show policy-map statistics for marking policies.
- An in-place modification of an ACL does not reset the policy-map statistics counter.

**Note**

- For QOS EXP-Egress marking applied on a Layer 3 interface on Cisco NCS550x and NCS55Ax routers, there is a limit of two unique policy-maps per NPU. This limit is three unique policy maps per NPU for routers that have the Cisco NC57 line cards installed. When the maximum limit for policy-maps is reached and you try to modify a policy-map which is shared between different interfaces, you may get an error.
- For QOS egress marking (CoS, DEI) applied on a Layer 2 interface, there is a limit of 13 unique policy-maps per NPU. When the maximum limit for policy-maps is reached and you try to modify a policy-map which is shared between different interfaces, you may get an error.

### Verification

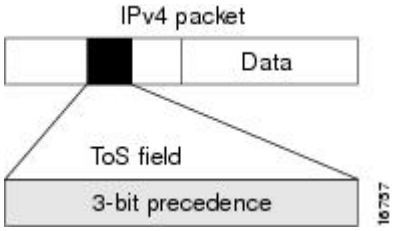
If unrecoverable errors occur during in-place policy modification, the policy is put into an inconsistent state on target interfaces. No new configuration is possible until the configuration session is unblocked. It is recommended to remove the policy from the interface, check the modified policy and then re-apply accordingly.

# References for Modular QoS Service Packet Classification

## Specification of the CoS for a Packet with IP Precedence

Use of IP precedence allows you to specify the CoS for a packet. You can create differentiated service by setting precedence levels on incoming traffic and using them in combination with the QoS queuing features. So that, each subsequent network element can provide service based on the determined policy. IP precedence is usually deployed as close to the edge of the network or administrative domain as possible. This allows the rest of the core or backbone to implement QoS based on precedence.

***Figure 2: IPv4 Packet Type of Service Field***




You can use the three precedence bits in the type-of-service (ToS) field of the IPv4 header for this purpose. Using the ToS bits, you can define up to eight classes of service. Other features configured throughout the network can then use these bits to determine how to treat the packet in regard to the ToS to grant it. These other QoS features can assign appropriate traffic-handling policies, including congestion management strategy and bandwidth allocation. For example, queuing features such as LLQ can use the IP precedence setting of the packet to prioritize traffic.

### IP Precedence Bits Used to Classify Packets

Use the three IP precedence bits in the ToS field of the IP header to specify the CoS assignment for each packet. You can partition traffic into a maximum of eight classes and then use policy maps to define network policies in terms of congestion handling and bandwidth allocation for each class.

Each precedence corresponds to a name. IP precedence bit settings 6 and 7 are reserved for network control information, such as routing updates. These names are defined in RFC 791.

### IP Precedence Value Settings

By default, the routers leave the IP precedence value untouched. This preserves the precedence value set in the header and allows all internal network devices to provide service based on the IP precedence setting. This policy follows the standard approach stipulating that network traffic should be sorted into various types of service at the edge of the network and that those types of service should be implemented in the core of the network. Routers in the core of the network can then use the precedence bits to determine the order of transmission, the likelihood of packet drop, and so on.

Because traffic coming into your network can have the precedence set by outside devices, we recommend that you reset the precedence for all traffic entering your network. By controlling IP precedence settings, you prohibit users that have already set the IP precedence from acquiring better service for their traffic simply by setting a high precedence for all of their packets.

The class-based unconditional packet marking and LLQ features can use the IP precedence bits.

## IP Precedence Compared to IP DSCP Marking

If you need to mark packets in your network and all your devices support IP DSCP marking, use the IP DSCP marking to mark your packets because the IP DSCP markings provide more unconditional packet marking options. If marking by IP DSCP is undesirable, however, or if you are unsure if the devices in your network support IP DSCP values, use the IP precedence value to mark your packets. The IP precedence value is likely to be supported by all devices in the network.

You can set up to 8 different IP precedence markings and 64 different IP DSCP markings.

# Conditional Marking of MPLS Experimental bits for L3VPN Traffic

The conditional marking of MPLS experimental bits is achieved for Layer 3 Virtual Private Network (L3VPN) 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.

This feature is supported on both IPv4 and IPv6 traffic in the L3VPN network. Conditional marking can be used to mark the MPLS experimental bits 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 are the packets which have exceeded the limit and have the color yellow and discard-class set to 1.

Conditional marking of MPLS experimental bits for L3VPN traffic is supported on both physical and bundle main interfaces as well as sub-interfaces.

### Restrictions for Conditional Marking of MPLS Experimental bits on L3VPN

1. In the case of two PE routers connected back-to-back and the only label that the traffic between the routers have is the BGP label, then the explicit null label should be configured.
2. A maximum of three policy-maps which perform conditional marking of MPLS experimental bits can be configured per Network Processor Unit (NPU) of the Cisco NCS 5500 Series Routers.
3. In the ingress policy-map if qos-group is being set for the incoming traffic packets, then setting of dscp and mpls experimental bits will not work.
4. Both the ingress and egress policy-maps must be applied in order to attain the expected behaviour. If either one of them is not applied then it may lead to undefined behaviour.
5. If the egress policy-map does not match on qos-group or discard-class and set the mpls experiment bits to the required value, then the mpls experimental bits will be set to a value of zero, by default.

## QoS DSCP Preservation

*Table 10: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| QoS DSCP Preservation | Release 7.2.1 | This feature is now also supported on routers that have the Cisco NC57 line cards installed and operate in the native mode. |

The DSCP value of the packet is preserved only for L3VPN networks when the packets are destined to the directly connected routes on the PE routers. To preserve the DSCP value for packets to destinations beyond the egress PE routers for L3VPN, you should use the **label mode per-vrf** command under the VRF at the PE routers. Similarly for MPLS networks, the default behaviour is uniform mode where the penultimate hop copies the MPLS EXP bit to IP DSCP and IP DSCP value is not preserved beyond the egress PE router. To preserve the IP DSCP value, you should use the **mpls ip-ttl-propagate disable** command at the penultimate hop.

DSCP value preservation is not supported on NC57-24DD and NC57-18DD-SE line cards for Cisco IOS XR Release 7.0.2.

## Policy-map for conditional marking of incoming IPv4 and IPv6 traffic

The incoming packets are classified based on the ingress policy-map and the following actions are done.

- Set qos-group
- Discard class or drop precedence is set implicitly or as a result of a policing action.
- Packets that violate the configured policer are dropped in the ingress processing itself.

### Running Configuration:

```
policy-map ingress
 class af11
  police rate percent 10 peak-rate percent 20
  !
  set qos-group 1
 !
 class af22
  police rate percent 30 peak-rate percent 50
  !
  set qos-group 2
 !
 class af32
  set qos-group 3
  police rate percent 30 peak-rate percent 60
  !
 !
 class class-default
 !
 end-policy-map
!
```

## Policy-map for conditional marking of outgoing MPLS traffic

The IPv4 or IPv6 ingress packet undergoes MPLS encapsulation during the egress processing in the PE router which performs the label imposition. The MPLS experimental bits are marked on the basis of egress policy-map which performs the following actions:

- Match on qos-group or discard class or both
- Set the MPLS experimental bits based on the match criteria

### Running Configuration:

```
policy-map egress
 class qos1_disc0 # This class matches on qos-group 1 and discard-class 0
```

```
  set mpls experimental imposition 1
 !
 class qos1_disc1 # This class matches on qos-group 1 and discard-class 1
  set mpls experimental imposition 5
 !
 class qos2_disc0 # This class matches on qos-group 2 and discard-class 0
  set mpls experimental imposition 2
 !
 class qos2_disc1 # This class matches on qos-group 2 and discard-class 1
  set mpls experimental imposition 6
 !
 class qos3_disc0 # This class matches on qos-group 3 and discard-class 0
  set mpls experimental imposition 3
 !
 class qos3_disc1 # This class matches on qos-group 3 and discard-class 1
  set mpls experimental imposition 7
 !
 class class-default
 !
 end-policy-map
!
```

# Conditional Marking of MPLS Experimental bits for L2VPN Traffic

**Note** This feature is not available on NC57-24DD and NC57-18DD-SE line cards for Cisco IOS XR Release 7.0.2.

Conditional marking of MPLS EXP bits is supported on Virtual Private Wire Service (VPWS), Virtual Private LAN Service (VPLS) and Ethernet Virtual Private Network (EVPN) traffic in the L2VPN network.

The conditional marking of MPLS experimental bits is achieved for Layer 2 Virtual Private Network (L2VPN) 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 on a combination of qos-group and discard-class and sets the mpls experiment bits to the corresponding value.

Conditional marking can be used to mark the MPLS experimental bits 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 are the packets which have exceeded the limit and have the color yellow and discard-class set to 1.

Conditional marking of MPLS experimental bits for L2VPN traffic is supported on both physical and bundle main interfaces as well as sub-interfaces.

### Restrictions for Conditional Marking of MPLS Experimental bits on L2VPN

1. Conditional marking of MPLS EXP bits is not supported for broadcast, unknown unicast and multicast (BUM) traffic.
2. In the case of two PE routers connected back-to-back and the only label that the traffic between the routers have is the BGP label, then the explicit null label should be configured.
3. A maximum of two policy-maps which perform conditional marking of MPLS experimental bits can be configured per Network Processor Unit (NPU) of the Cisco NCS 5500 Series Routers. However, the same policy can be applied on multiple interfaces on the same NPU.

4. In the ingress policy-map if qos-group is being set for the incoming traffic packets, then setting of dscp and mpls experimental bits will not work.
5. Both the ingress and egress policy-maps must be applied in order to attain the expected behaviour. If either one of them is not applied then it may lead to undefined behaviour.
6. If the egress policy-map does not match on qos-group or discard-class and set the mpls experiment bits to the required value, then the mpls experimental bits will be set to a value of zero, by default.

## Policy-map for conditional marking of incoming traffic

The incoming packets on the Power Edge router are classified based on the ingress policy-map and these actions are taken.

- Set qos-group
- Discard class or drop precedence is set implicitly or as a result of a policing action.
- Set traffic class
- Packets that violate the configured policer are dropped in the ingress processing itself.

**Running Configuration:**

```
class-map af11
   match cos 1
!

policy-map ingress
 class af11
  police rate percent 10 peak-rate percent 20
  !
  set qos-group 1
  set Traffic-class 3
 !
 class class-default
 !
 end-policy-map
!
```

## Policy-map for conditional marking of outgoing MPLS traffic

The ingress packet undergoes MPLS encapsulation during the egress processing in the PE router which performs the label imposition. The MPLS experimental bits are marked on the basis of egress policy-map which performs the following actions:

- Match on qos-group or discard class or both
- Set the MPLS experimental bits based on the match criteria

**Running Configuration:**

```
class-map match-all qos-group2_0
   match qos-group 2
   match discard-class 0

policy-map egress-marking
 class qos-group2_0 # This class matches on qos-group 2 and discard-class 0
```

```
  set mpls experimental imposition 1
 !
 class class-default
 !
 end-policy-map
!
policy-map Egress-Queuing
 class Traffic-class3
  shape average 500 mbps
!
 class class-default
!
end-policy-map
!
```

# Conditional Marking of MPLS Experimental Bits for EVPN-VPWS Single-Homing Services

*Table 11: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Conditional Marking of MPLS Experimental Bits for EVPN-VPWS Single-Homing Services | Release 7.3.1 | This feature enables you to differentiate traffic in the MPLS forwarding domain and manage traffic from ingress PE to egress PE based on the MPLS EXP bit of the MPLS header.<br>This feature is supported only for EVPN-VPWS single-homing services, and not supported for EVPN-VPWS multi-homing services. |

The conditional marking of MPLS experimental bits is achieved for EVPN-VPWS single-homing services 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 on a combination of qos-group and discard-class and sets the MPLS experiment bits to the corresponding value.

Conditional marking can be used to mark the MPLS experimental bits 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 are the packets that have exceeded the limit and have the color yellow and discard-class set to 1.

Conditional marking of MPLS experimental bits for EVPN-VPWS single-homing services are supported on both physical and bundle main interfaces as well as sub-interfaces.

### MPLS EXP Marking for EVPN Multi-Homed Services

*Table 12: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|

| | | |
|---|---|---|
| MPLS EXP Marking for EVPN Multi-Homed Services | Release 7.6.1 | You can now differentiate traffic in the MPLS forwarding domain and manage traffic from ingress PE to egress PE based on the MPLS EXP bit of the MPLS header for EVPN multi-home scenarios. |

**Note** This feature is not supported on routers that have the Cisco NC57 line cards installed and operate in the native mode or compatibility mode.

### Configuration

- The ingress policing is applied on the UNI interface. It is with set qos-group and set traffic class.
- The marking policy is applied at the core facing NNI interface.
- MPLS EXP imposition is marked while packets egress from NNI Interface.

### Running Configuration

```
interface TenGigE0/0/0/2.203 l2transport  => This is UNI
encapsulation dot1q 203
service-policy input pol50-100

interface TenGigE0/0/0/10  ===============> This is the core NNI
description *** CORE IF  ***
cdp
service-policy input in_mpls
service-policy output eg_mark
ipv4 address 192.18.44.18 255.255.255.0
ipv6 address 2005:18:44::18/48
lldp
  enable
!
monitor-session test ethernet direction tx-only port-level
!
load-interval 30
!
l2vpn
xconnect group 203
  p2p 203
   interface TenGigE0/0/0/2.203
   neighbor evpn evi 1 service 203

policy-map pol50-100
class class-default
  set traffic-class 2
  set qos-group 4
  police rate 50 mbps peak-rate 100 mbps
  !
!
end-policy-map
!

policy-map eg_mark
class qg4dc0
  set mpls experimental imposition 2
!
```

```
class qg4dc1
 set mpls experimental imposition 3
!
class class-default
!
end-policy-map
!
class-map match-all qg4dc0
match qos-group 4
match discard-class 0
end-class-map
!

class-map match-all qg4dc1
match qos-group 4
match discard-class 1
end-class-map
```

### Verification

Verify that you have configured conditional marking of MPLS experimental bits for EVPN-VPWS single-homing services successfully.

```
Router#show qos int tenGigE 0/0/0/2.101 input
NOTE:- Configured values are displayed within parentheses
Interface TenGigE0/0/0/2.101 ifh 0x41da  -- input policy
NPU Id:                        0
Total number of classes:       1
Interface Bandwidth:           10000000 kbps
Policy Name:                   pol50-100
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   class-default
New traffic class                         =   2
New qos group                             =   4

Policer Bucket ID                         =   0x18
Policer Stats Handle                      =   0x0
Policer committed rate                    =   49219 kbps (50 mbits/sec)
Policer peak rate                         =   98438 kbps (100 mbits/sec)
Policer conform burst                     =   62336 bytes (default)
Policer exceed burst                      =   187008 bytes (default)
--------------------------------------------------------------------------------

Router#show qos int tenGigE 0/0/0/10 output
Tue Sep  1 04:18:27.508 UTC
NOTE:- Configured values are displayed within parentheses
Interface TenGigE0/0/0/10 ifh 0xe0  -- output policy
NPU Id:                        0
Total number of classes:       3
Interface Bandwidth:           10000000 kbps
Policy Name:                   eg_mark
VOQ Base:                      0
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   qg4dc0
New imposition exp                        =   2
Queue Max. BW.                            =   no max (default)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   0 / (BWR not configured)

Level1 Class                              =   qg4dc1
New imposition exp                        =   3
```

```
Queue Max. BW.                          =   no max (default)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   0 / (BWR not configured)

Level1 Class                            =   class-default
Queue Max. BW.                          =   no max (default)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   0 / (BWR not configured)
--------------------------------------------------------------------------------
```

# Classifying Packets Based On MPLS Experimental Bits in MPLS Over GRE

*Table 13: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Classifying Packets Based On MPLS Experimental Bits in MPLS Over GRE | Release 7.5.2 | For MPLS over GRE scenarios that tunnel MPLS traffic over non-MPLS networks, you can now perform QoS classification for specific traffic or applications based on MPLS EXP bit field values in the MPLS header.<br><br>In earlier releases, you could perform QoS classification only in the outer GRE IP header using DiffServ Code Point (DSCP) or IP precedence bits that helped you achieve the required line rate minus the granularity.<br><br>This feature introduces the **hw-module profile qos gre-exp-classification-enable** command. |

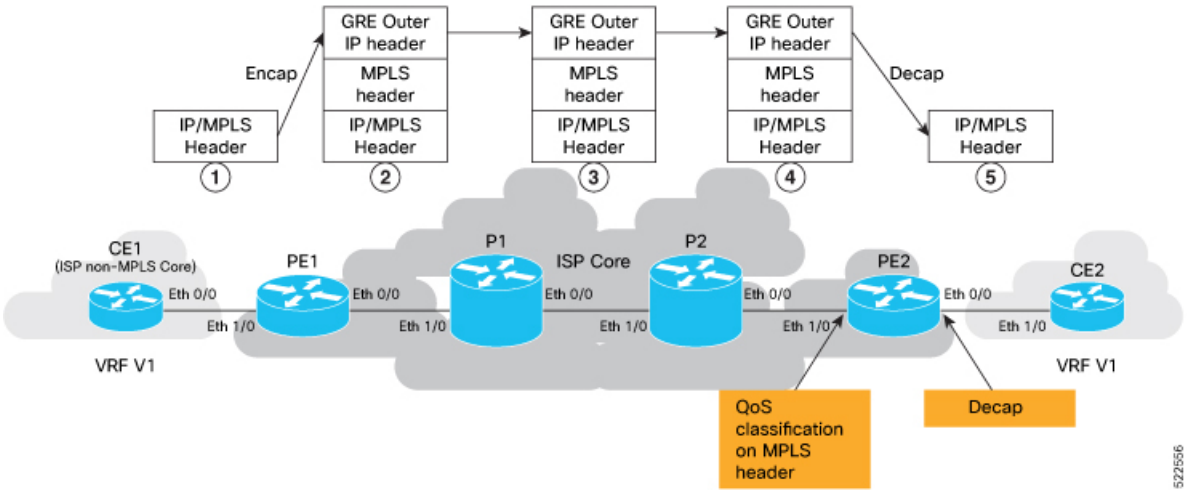
You can now perform QoS classification based on the MPLS header that the GRE IP header encapsulates in a single-pass GRE scenario. This classification takes place on the MPLS labels (specifically the experimental bits or EXP) in the MPLS header after the GRE is decapsulated using Policy-Based Routing (PBR). Because the MPLS labels identify specific customer traffic and applications, the classification is more granular. As the traffic moves along the single-pass GRE, the POP tag removes the MPLS header, and the outgoing traffic with QoS classification has the inner IP header. (Also, see Terms Used for Classifying Packets Based On MPLS Experimental Bits in MPLS Over GRE .)

To enable QoS classification on the inner MPLS header, run the command **hw-module profile qos gre-exp-classification-enable**.

**Note** Ensure that you reload the line card for this command to take effect. If you don't reload the line card, QoS classification will continue on the outer GRE IP header with DSCP or IP precedence marking.

To understand how this functionality works, see the figure **QoS Classification on Inner MPLS Header for MPLS over GRE**.

***Figure 3: QoS Classification on Inner MPLS Header for MPLS over GRE***




1. The MPLS packet approaches the beginning of the GRE tunnel. The GRE tunnel is configured across a non-MPLS network on the provider edge (PE) devices, PE1 and PE2, which are at either end of the tunnel.
2. GRE encapsulation is added at tunnel entry point.
3. The MPLS packet traverses the non-MPLS network, where only the outer IP header is parsed by routers.
4. When the packet reaches the ingress interface (Eth 1/0) on router PE2, QoS classification is performed based on MPLS EXP bits. Before the packet reaches the egress interface (Eth 0/0) of PE2, the outer GRE header is removed.
5. The packet with the MPLS header now travels toward its ultimate destination to CE2.

## Behavioral Specifications

- This feature is supported only in scenarios where you enable GRE single-pass. (You configure the **tunnel mode gre ipv4 encap** command in GRE single-pass. See Set Up and Configuration for QoS Classification on MPLS Header, on page 66.)
- To preserve the inner DSCP markings and IP Time-to-Live (TTL) values, you must disable the propagation of IP Time-to-Live (TTL) to and from the MPLS header for forwarded and local packets. See **mpls ip-ttl-propagate**.

## Restrictions

This feature isn't supported on routers that have Cisco NC57 line cards installed and operate in the native and compatibility mode.

## Set Up and Configuration for QoS Classification on MPLS Header

This section describes the configuration details for QoS classification on the MPLS header in a single pass GRE tunnel per the Figure **QoS Classification on Inner MPLS Header for MPLS over GRE**.

Configuration for PE1 (encap node) where you:

- enable GRE single-pass.
- assign the GRE tunnel source and destination addresses.
- assign IPv4 addresses for PE1 and PE2.

```
Router(config)#interface tunnel-ip_intf1
Router(config-if)#ipv4 address 203.0.113.1 255.255.255.252
Router(config-if)#tunnel mode gre ipv4 encap
Router(config-if)#tunnel source 13.1.1.1
Router(config-if)#tunnel destination 11.1.1.1
Router(config-if)#root
Router(config)#interface Eth 1/0
Router(config-if)#ipv4 address 209.165.201.1 255.255.255.224
Router(config-if)#exit
Router(config-if)#interface Eth 0/0
Router(config-if)#ipv4 address 198.51.100.1 255.255.255.0
Router(config-if)#commit
```

Configuration for intermediate hop router (P1) where you configure the forwarding addresses for packets.

```
Router(config)#interface Eth1/0
Router(config-if)#ipv4 address 198.51.100.2 255.255.255.0
Router(config-if)#root
Router(config)#
Router(config)#interface Eth0/0
Router(config-if)#ipv4 address 203.0.113.1 255.255.255.0
Router(config-if)#root
Router(config)#commit
```

Configuration for intermediate hop router (P2) where you configure the forwarding addresses for packets.

```
Router(config)#interface Eth1/0
Router(config-if)#ipv4 address 203.0.113.2 255.255.255.0
Router(config-if)#root
Router(config)#
Router(config)#interface Eth0/0
Router(config-if)#ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)#root
Router(config)#commit
```

Configuration for PE2 (decap node) where you:

- disable TTL to preserve the IP header DSCP and TTL.
- configure **hw-module profile qos gre-exp-classification-enable**.

**Note** Ensure that you reload the line card for this command to take effect. If you don't reload the line card, QoS classification will continue on the outer GRE IP header with DSCP or IP precedence marking. We recommend that you use the **hw-module location all reload** command in Sysadmin VM.

- configure MPLS EXP bits, class-map, and the policy-map before decapsulation.

```
Router(config)#mpls ip-ttl-propagate disable
Router(config)# hw-module profile qos gre-exp-classification-enable
NOTE:To activate this profile, you must manually reload the chassis/all line cards
Router(config)#
```

```
#reload the device to get effect of qos gre-exp-classification
/*We recommend that you use the hw-module location all reload command in Sysadmin
VM*/
Router#admin
root connected from 192.0.108.4 using ssh on sysadmin-vm:0_RP0
sysadmin-vm:Router#hw-module location all reload
!

/*After reloading, configure MPLS exp bits, class-map, and the policy-map*/
Router(config)#class-map match-any exp1
Router(config-cmap)#match mpls experimental topmost 1
Router(config-cmap)#end-class-map
Router(config)#class-map match-any exp2
Router(config-cmap)#match mpls experimental topmost 2
Router(config-cmap)#end-class-map
Router(config)#class-map match-any cs1
Router(config-cmap)#match dscp cs1
Router(config-cmap)#end-class-map
Router(config)#policy-map gre-ip_in
Router(config-pmap)#class exp1
Router(config-pmap-c)#set traffic-class 1
Router(config-pmap-c)#exit
Router(config-pmap)#class exp2
Router(config-pmap-c)#set traffic-class 2
Router(config-pmap-c)#exit
Router(config-pmap)#class class-default
Router(config-pmap-c)#exit
Router(config-pmap)#end-policy-map
Router(config)#
Router(config)#interface Eth0/0
Router(config-if)#service-policy input gre-ip_in
Router(config-if)#ipv4 address 192.0.2.2 255.255.255.0
Router(config-if)#load-interval 30
Router(config)#
Router(config)#interface Eth1/0
Router(config-if)#ipv4 address 209.165.202.129 255.255.255.224
Router(config-if)#load-interval 30
Router(config-if)#root
Router(config-if)#commit
```

### Running Configuration

```
interface tunnel-ip_intf1
 ipv4 address 203.0.113.1 255.255.255.252
 tunnel mode gre ipv4 encap
 tunnel source 13.1.1.1
 tunnel destination 11.1.1.1
!
interface Eth 1/0
 ipv4 address 209.165.201.1 255.255.255.224

!

interface Eth 0/0
 ipv4 address 198.51.100.1 255.255.255.0

!
!

/*Configuration for intermediate hop router (P1)*/
interface Eth1/0
 ipv4 address 198.51.100.2 255.255.255.0
```

```
!

interface Eth0/0
 ipv4 address 203.0.113.1 255.255.255.0

!
!
/*Configuration for intermediate hop router (P2)*/
interface Eth1/0
 ipv4 address 203.0.113.2 255.255.255.0
!
interface Eth0/0
 ipv4 address 192.0.2.1 255.255.255.0

!
!
mpls ip-ttl-propagate disable
hw-module profile qos gre-exp-classification-enable
reload loc all
class-map match-any exp1
 match mpls experimental topmost 1
 end-class-map

!
class-map match-any exp2
 match mpls experimental topmost 2
 end-class-map
!
class-map match-any cs1
 match dscp cs1
 end-class-map
!
policy-map gre-ip_in
 class exp1
  set traffic-class 1

!
 class exp2
  set traffic-class 2

!
 class class-default

!
end-policy-map
!
interface Eth0/0
 service-policy input gre-ip_in
 ipv4 address 192.0.2.2 255.255.255.0
 load-interval 30

!
interface Eth1/0
 ipv4 address 209.165.202.129 255.255.255.224
 load-interval 30
!
```

### Verification

Run the **show policy-map** command on PE2 (decapsulation node) to view the classification statistics that confirm QoS classification is on the MPLS header.

```
Router#show policy-map int Eth1/0
Eth1/0 input: gre-ip_in
```

```
Class exp1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            55436316/8426320032           504105
    Transmitted         :            55436316/8426320032           504105
    Total Dropped       :                   0/0                    0
Class exp2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            55436286/8426315472           504105
    Transmitted         :            55436286/8426315472           504105
    Total Dropped       :                   0/0                    0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
Policy Bag Stats time: 1648011883484
```

When you don't enable QoS classification on the MPLS header, the **show policy-map** command on PE2 (decapsulation node) confirms that the classification is IP precedence (under **class-default**).

```
Router#show policy-map int Eth1/0
Eth1/0 input: gre-ip_in

Class exp1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
Class exp2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           108028694/16420361488          1008187
    Transmitted         :           108028694/16420361488          1008187
    Total Dropped       :                   0/0                    0
Policy Bag Stats time: 1648013053478
```

# Terms Used for Classifying Packets Based On MPLS Experimental Bits in MPLS Over GRE

### Generic Routing Encapsulation (GRE)

A protocol used to encapsulate packets, GRE is useful when you need to transport packets across an unsupported network protocol, say IPv6 packets across an IPv4 network. In this case, IPv6 packets are encapsulated or wrapped around by packets that support the protocol (IPv4) and transported across the network. This is akin to cars being transported across oceans (unsupported protocol for cars) on cargo ships (the ocean being the supported protocol for the ships).

### GRE Tunneling

The encapsulation of packets within other packets is called GRE tunneling. Here, if packets are to be transported from one router to another without any additional processing by intermediate routers and devices, you can configure a GRE tunnel between the two routers. As the packets flow across the network, the intermediate routers read the forwarding information on the packet headers and send them to their next hop. This is similar to the cars now being transported on a cargo truck on land, passing through a tunnel to reach the other end of the road faster, with no other action taking place on the cars as they are carried on the truck inside the tunnel.

Without the tunnel, transport across this route would be impossible, akin to an unsupported network between the two ends of the GRE tunnel.

### GRE Headers

Each packet consists of the payload—the actual packet content—and an IP header that contains information critical for the packet transportation: the source address, the destination address, packet number, packet length, and so on.

As with other network protocols, GRE adds its own headers to the packets: the GRE header that shows the protocol used by the encapsulated packet (the cars, in our preceding example) and an outer IP header that wraps the encapsulated packet's payload and IP header. A GRE header thus has two IP headers.

### MPLS over GRE

A mechanism for tunneling MPLS packets over non-MPLS networks by creating a GRE tunnel across a non-MPLS network. The MPLS packets are encapsulated within the GRE tunnel packets, and the encapsulated packets traverse the non-MPLS network through the GRE tunnel. When GRE tunnel packets reach the other side of the non-MPLS network, the GRE tunnel packet header is removed and the inner MPLS packet is forwarded to its final destination.

# QPPB

QoS Policy Propagation via BGP (QPPB) is a mechanism that allows propagation of quality of service (QoS) policy and classification by the sending party that is based on the following:

- Access lists
- Community lists
- Autonomous system paths in the Border Gateway Protocol (BGP)

Thus, helps in classification that is based on the destination address instead of the source address.

QoS policies that differentiate between different types of traffic are defined for a single enterprise network. For instance, one enterprise may want to treat important web traffic, not-important web traffic, and all other data traffic as three different classes. And thereafter, use the different classes for the voice and video traffic.

Hence, QPPB is introduced to overcome the following problems:

- The administrative challenges of classifying that is based on ACLs.
- The administrative problems of just listing the networks that need premium services.

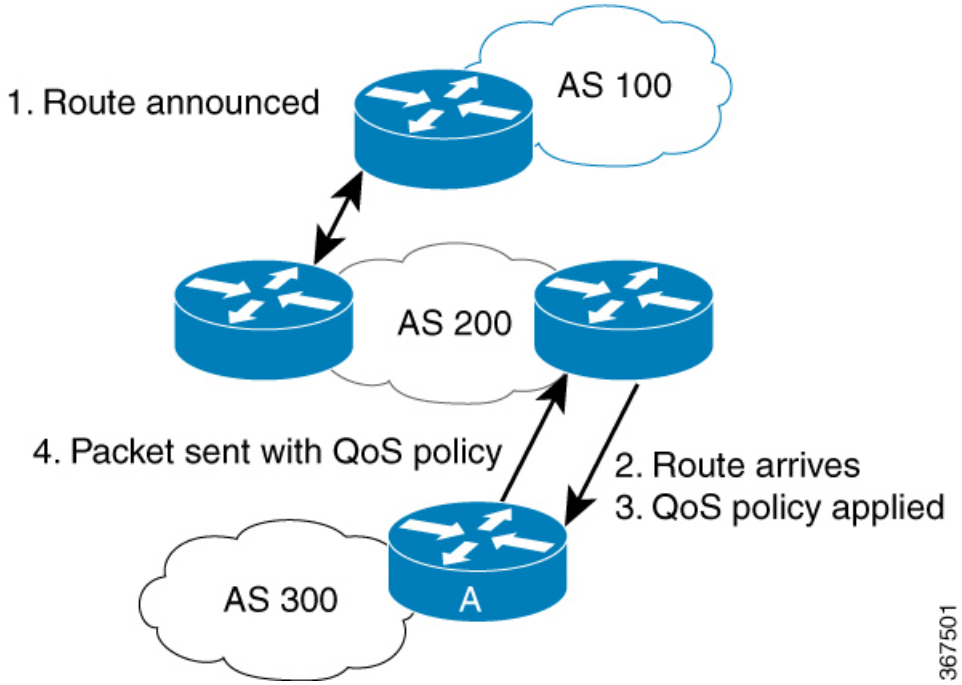
QPPB allows marking of packets that are based on QoS group value associated with a Border Gateway Protocol (BGP) route.

**Benefits of QPPB**

- QPPB provides an IP prefix-based QoS capability.
- Traffic to IP addresses that have specific IP prefixes can be prioritized above other IP addresses.
- IP prefixes of interest are tagged through the control plane that uses common BGP route-map techniques, including the community attribute.

- Traffic to the tagged BGP prefixes is then classified and prioritized via the data forwarding plane by using the IOS-XR MQC (Modular QoS CLI) mechanisms, such as re-marking.
- QPPB provides the glue between the BGP control plane and the IP data forwarding plane in support of IP prefix-based QoS.
- BGP configuration within QPPB uses a table map to match specific prefixes learned through BGP neighbors, and then sets the router's local QoS Group variable maintained within the Forwarding Information Base (FIB) for those specific prefixes.
- The router supports a subset of full QPPB options - only IP destination prefix mode on input policy is supported.

***Figure 4: Sample Scenario***




Router A learns routes from AS 200 and AS 100. QoS policy is applied to any ingress interface of Router A to match the defined route maps with destination prefixes of incoming packets. Matching packets on Router A to AS 200 or AS 100 are sent with the appropriate QoS policy from Router A.

BGP maintains a scalable database of destination prefixes, QPPB, by using BGP table maps. BGP adds the ability to map a qos-group value to desired IP destinations. These qos-group values are used in QOS policies applied locally on ingress interfaces. Whenever a packet bound for such destinations is encountered, the qos-group value matching that destination route looks up with work inside the policy classmap, and marks that packet for any configured policy actions.

# QPPB: Guidelines and Limitations

- QPPB is supported only on the -SE variants of NCS 5700-line card-based routers and the following -SE variants of NCS 5500-line card-based routers:
  - NCS-55A1-36H-SE-S

- NCS-55A2-MOD-SE-H-S
- NCS-55A2-MOD-SE-S
- NC55-36x100G-A-SE
- NC55-MOD-A-SE-S

- QPPB doesn't work for /32 IPv4 prefixes on NCS 5500 line card-based routers when you enable the **hw-module fib mpls label lsr-optimized** profile.
- You must configure the **hw-module profile qos ipv6 short** command for QPPB to work with IPv6 address families and packets.
- When you enable the peering mode (using the **hw-module profile qos ingress-model peering** command), the QPPB feature doesn't work.

# Configuration Workflow

Use the following configuration workflow for QPPB:

- Define route policy.
- Put Route policy at table-policy attach point under BGP.
- Define classmaps and ingress policy to use the qos-groups that are used in table-policy.
- Enable ipv4/ipv6 QPPB configuration under the desired interfaces.
- Configure the QPPB hardware profile, *hw-module profile qos ipv6 short*.
- If you use ipv6 QPPB, you must reload that linecard. If you use only ipv4 QPPB, linecard reload is not mandatory.

### Define route policy

A routing policy instructs the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another.

The routing policy language (RPL) provides a language to express routing policy. You must set up destination prefixes either to match inline values or one of a set of values in a prefix set.

Example:

```
prefix-set prefix-list-v4
    70.1.1.1,
    70.2.1.0/24,
    70.2.2.0/24 ge 28,
    70.2.3.0/24 le 28
end-set
prefix-set prefix-list-v6
    2001:300::2,
    2003:200::3
end-set

route-policy qppb1
    if destination in (60.60.0.2) then
        set qos-group 5
    elseif destination in prefix-list-v4 then
```

```
        set qos-group 4
    else
        set qos-group 1
     pass
endif
end-policy
```

For NC57 line cards, the set qos-group value under the route policy for QPPB support is 0-15. Even though the set qos-group value option displays supported values <0-31> as show below, the configuration fails for any value above 15 for NC57 line cards.

```
RP/0/RP0/CPU0:Fretta(config-rpl-if)#set qos-group ?
  <0-31>     decimal number
```

### Put Route policy at table-policy attach point under BGP

The table-policy attach point permits the route policy to perform actions on each route as they are installed into the RIB routing table. QPPB uses this attachment point to intercept all routes as they are received from peers. Ultimately the RIB will update the FIB in the hardware forwarding plane to store destination prefix routing entries, and in cases where table policy matches a destination prefix, the qos-group value is also stored with the destination prefix entry for use in the forwarding plane.

Example:

```
router bgp 900
    [vrf <name>]
    bgp router-id 22.22.22.22
    address-family ipv4 unicast
        table-policy qppb1
    address-family ipv6 unicast
        table-policy qppb2
    neighbor 30.2.2.1
        remote-as 500
        address-family ipv4 unicast
            route-policy pass in
            route-policy pass out
        address-family ipv6 unicast
            route-policy pass in
            route-policy pass out
```

### Ingress interface QOS and ipv4/ipv6 bgp configuration

QPPB would be enabled per interface and individually for V4 and V6. An ingress policy would match on the qos groups marked by QPPB and take desired action.

If a packet is destined for a destination prefix on which BGP route policy has stored a qos-group, but it ingresses on an interface on which qppb is not enabled, it would not be remarked with qos-group.

Earlier, router supported matching on qos-group only in peering profile ‘hw-module profile qos ingress-model peering location <>’ . QPPB now permits classmaps to match qos-group in the default “non peering mode qos” as well. Also QPPB and hierarchical QOS policy profiles can work together if Hqos is used.

Example:

```
class-map match-any qos-group5
    match qos-group 5
    end-class-map

class-map match-any qos-group4
    match qos-group 4
    end-class-map
```

```
policy-map ingress-marker-po1
    class qos-group5
        set precedence 0
        set discard-class 0
        set traffic-class 1

    class qos-group4
        set precedence 1
        set discard-class 1
        set traffic-class 2
    class class-default

end-policy-map
```

# Configuring QPPB on an Interface

1. RP/0/RP0/CPU0:router # configure

   Enters interface configuration mode and associates one or more interfaces to the VRF.

2. 
   ```
   RP/0/RP0/CPU0:router(config)# interface
   type interface-path-id
   ```

   Enters interface configuration mode and associates one or more interfaces to the VRF.

3. **ipv4 | ipv6 bgp policy propagation inputqos-groupdestination**

   Example:

   ```
   RP/0/RP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination
   ```

   Enables QPPB on an interface

4. commit

# Egress Interface Configuration

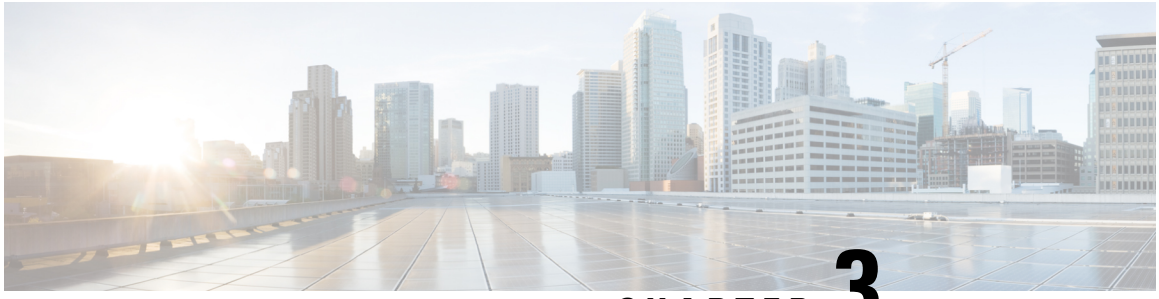
The traffic-class set on ingress has no existence outside the device. Also, traffic-class is not a part of any packet header but is associated internal context data on relevant packets. It can be used as a match criteria in an egress policy to set up various fields on the outgoing packet or shape flows.

**Restrictions**:

- No IP precedence marking.
- No policing on egress policy.

```
class-map match-any level1
     match traffic-class 1
end-class-map

class-map match-any level2
    match traffic-class 2
end-class-map

policy-map output-po1
    class level1
```

```
        bandwidth percent 50
    class level2
        bandwidth percent 20
        queue-limit 50 ms
end-policy-map

interface hun 0/5/0/0
     ipv4 address 30.1.1.1/24
     ipv6 address 2001:da8:b0a:12f0::1/64
     service-policy output output-po1
```



CHAPTER 3

# Configuring Modular QoS Congestion Avoidance

This chapter covers the following topics:


- Modular QoS Congestion Avoidance , on page 77
- Tail Drop and the FIFO Queue, on page 78
- Random Early Detection and TCP, on page 80
- Weighted Random Early Detection, on page 83
- Explicit Congestion Notification (ECN), on page 89


## Modular QoS Congestion Avoidance

Congestion avoidance techniques monitor traffic flow to anticipate and avoid congestion at common network bottlenecks. Avoidance techniques are implemented before congestion occurs as compared with congestion management techniques that control congestion after it has occurred.

**Note** From Cisco IOS XR Release 7.3.1 onwards, systems with Cisco NC57 line cards running in compatibility mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

Starting with Cisco IOS XR Release 7.4.1 systems with Cisco NC57 line cards running in native mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

**Note** For traffic requiring header decapsulation, the size of the header that is being removed is still included for the egress queuing actions. To offset this header size (required to achieve line rate for small frame sizes), configure an egress user policy with user overhead accounting on the egress interface. This policy can be a dummy policy configuration as well (allowing full traffic rate), if a policy isn't already in use or required on the egress interface.

You can enable user overhead accounting using the optional configuration of **accounting user-defined** <***overhead size in bytes***> while attaching the service policy on the egress interface.

Congestion avoidance is achieved through packet dropping. The router supports these QoS congestion avoidance techniques:

- Tail Drop and the FIFO Queue, on page 78
- Random Early Detection and TCP, on page 80
- Weighted Random Early Detection, on page 83

# Tail Drop and the FIFO Queue

Tail drop is a congestion avoidance technique that drops packets when an output queue is full until congestion is eliminated. Tail drop treats all traffic flow equally and does not differentiate between classes of service. It manages the packets that are unclassified, placed into a first-in, first-out (FIFO) queue, and forwarded at a rate determined by the available underlying link bandwidth.

## Tail Drop and the FIFO Queue: Tip to Optimize Hardware Resources

Tail drop is a congestion avoidance mechanism that works by discarding packets at the tail end of a queue when the queue length exceeds a certain threshold. These thresholds are critical in managing network congestion and are determined based on the Guaranteed Service Rate (GSR) of a traffic class, typically set at a value representing 10 milliseconds of the GSR.

### Tail Drop Thresholds-Rate Class Profile Association and ASIC Limitations

When configuring tail drop thresholds, each threshold value is associated with a rate class profile, which defines the performance characteristics for that threshold. (A rate class profile typically includes bandwidth limits, priority levels, queueing policies, and drop policies like tail drop thresholds. These profiles manage how different types of traffic are treated as they traverse the network, ensuring that critical services get the bandwidth and latency treatment they require.)

However, there is a limitation: each ASIC can support only up to 64 rate class profiles. This hardware constraint necessitates efficient utilization of these profiles.

### Resource Optimization Tip: Share Rate Class Profiles

Since multiple traffic classes can have the same tail drop threshold value, they can share a rate class profile. This sharing conserves hardware resources, allowing for a more scalable network design. Conversely, if each traffic class requires a unique tail drop threshold, more rate class profiles must be created, which can quickly exhaust the available hardware resources.

Therefore, to optimize network performance and resource utilization, we recommend that you strategically fine-tune tail drop threshold values so that more traffic classes share rate class profiles.

# Configure Tail Drop

Packets satisfying the match criteria for a class accumulate in the queue reserved for the class until they are serviced. The **queue-limit** command is used to define the maximum threshold for a class. When the maximum threshold is reached, the enqueued packets to the class queue result in tail drop (packet drop).

### Restrictions

- When configuring the **queue-limit** command, you must configure one of the following commands: **priority**, **shape average**, **bandwidth** or **bandwidth remaining**, except for the default class.

### Configuration Example

You have to accomplish the following to complete the tail drop configuration:

1. Creating (or modifying) a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Specifying the maximum limit the queue can hold for a class policy configured in a policy map.
4. Specifying priority to a class of traffic belonging to a policy map.
5. (Optional) Specifying the bandwidth allocated for a class belonging to a policy map or specifying how to allocate leftover bandwidth to various classes.
6. Attaching a policy map to an output interface to be used as the service policy for that interface.

```
Router# configure
Router(config)# class-map qos-1
Router(config-cmap)# match traffic-class 1
Router(config-cmap)# commit
Router(config-pmap)# exit

Router(config)# policy-map test-qlimit-1
Router(config-pmap)# class qos-1
Router(config-pmap-c)# queue-limit 100 us
Router(config-pmap-c)# priority level 7
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy output test-qlimit-1
Router(config-if)# commit
```

### Running Configuration

```
class-map qos-1
 match traffic-class 1
commit
```

```
policy-map test-qlimit-1
 class qos-1
  queue-limit 100 us
  priority level 7
 !
 class class-default
 !
 end-policy-map
!
```

### Verification

```
Router# show qos int hundredGigE 0/6/0/18 output

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220  -- output policy
NPU Id:                        3
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
VOQ Base:                      11176
VOQ Stats Handle:              0x88550ea0
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class (HP7)                      =   qos-1
Egressq Queue ID                        =   11177 (HP7 queue)
TailDrop Threshold                      =   1253376 bytes / 100 us (100 us)
WRED not configured for this class

Level1 Class                            =   class-default
Egressq Queue ID                        =   11176 (Default LP queue)
Queue Max. BW.                          =   101803495 kbps (default)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   1 (BWR not configured)
TailDrop Threshold                      =   1253376 bytes / 10 ms (default)
WRED not configured for this class
```

### Related Topics

- Tail Drop and the FIFO Queue, on page 78

### Associated Commands

- queue-limit

# Random Early Detection and TCP

The Random Early Detection (RED) congestion avoidance technique takes advantage of the congestion control mechanism of TCP. By randomly dropping packets prior to periods of high congestion, RED tells the packet source to decrease its transmission rate. Assuming the packet source is using TCP, it decreases its transmission rate until all packets reach their destination, indicating that the congestion is cleared. You can use RED as a way to cause TCP to slow transmission of packets. TCP not only pauses, but it also restarts quickly and adapts its transmission rate to the rate that the network can support.

RED distributes losses in time and maintains normally low queue depth while absorbing traffic bursts. When enabled on an interface, RED begins dropping packets when congestion occurs at a rate you select during configuration.

# Configure Random Early Detection

The **random-detect** command with the **default** keyword must be used to enable random early detection (RED).

### Guidelines

If you configure the **random-detect default** command on any class including class-default, you must configure one of the following commands: **shape average**, **bandwidth**, and **bandwidth remaining**.

### Configuration Example

You have to accomplish the following to complete the random early detection configuration:

1. Creating (or modifying) a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Enabling RED with default minimum and maximum thresholds.
4. (Optional) Specifying the bandwidth allocated for a class belonging to a policy map or specifying how to allocate leftover bandwidth to various classes.
5. (Optional) Shaping traffic to the specified bit rate or a percentage of the available bandwidth.
6. Attaching a policy map to an output interface to be used as the service policy for that interface.

```
Router# configure
Router(config)# class-map qos-1
Router(config-cmap)# match traffic-class 1
Router(config-cmap)# commit
Router(config-pmap)# exit

Router# configure
Router(config)# policy-map test-wred-2
Router(config-pmap)# class qos-1
Router(config-pmap-c)# random-detect default
Router(config-pmap-c)# shape average percent 10
Router(config-pmap-c)# end-policy-map
Router(config)# commit
Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy output test-wred-2
Router(config-if)# commit
```

### Running Configuration

```
class-map qos-1
 match traffic-class 1
commit
```

```
policy-map test-wred-2
 class qos-1
  random-detect default
  shape average percent 10
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE 0/6/0/18
 service-policy output test-wred-2
!
```

## Verification

```
Router# show qos int hundredGigE 0/6/0/18 output

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220  -- output policy
NPU Id:                        3
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
VOQ Base:                      11176
VOQ Stats Handle:              0x88550ea0
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   qos-1
Egressq Queue ID                         =   11177 (LP queue)
Queue Max. BW.                           =   10082461 kbps (10 %)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 (BWR not configured)
Guaranteed service rate                  =   10000000 kbps
TailDrop Threshold                       =   12517376 bytes / 10 ms (default)

Default RED profile
WRED Min. Threshold                      =   12517376 bytes (10 ms)
WRED Max. Threshold                      =   12517376 bytes (10 ms)

Level1 Class                             =   class-default
Egressq Queue ID                         =   11176 (Default LP queue)
Queue Max. BW.                           =   101803495 kbps (default)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 (BWR not configured)
Guaranteed service rate                  =   50000000 kbps
TailDrop Threshold                       =   62652416 bytes / 10 ms (default)
WRED not configured for this class
```

## Related Topics

- Random Early Detection and TCP, on page 80

## Associated Commands

- random-detect

# Weighted Random Early Detection

The Weighted Random Early Detection (WRED) drops packets selectively based on any specified criteria, like discard-class. WRED uses this matching criteria to determine how to treat different types of traffic.

You can configure WRED using the **random-detect** command and different discard-class values. The value can be range or a list of values that are valid for that field. You can also use minimum and maximum queue thresholds to determine the dropping point. Ensure that the WRED maximum threshold value is close to the queue limit. When the maximum threshold value is reached, packets start to get dropped.

You can also configure WRED threshold values per discard class. Such an approach helps differentiate when to drop packets among different discard classes, helping prioritize packets among discard classes.

When a packet arrives, the following actions occur:

- The average queue size is calculated.
- If the average queue size is less than the minimum queue threshold, the arriving packet is queued.
- If the average queue size is between the minimum queue threshold for that type of traffic and the maximum threshold for the interface, the packet is either dropped or queued, depending on the packet drop probability for that type of traffic.
- If the average queue size is greater than the maximum threshold, the packet is dropped.

## Average Queue Size for WRED

The router automatically determines the parameters to use in the WRED calculations. The average queue size is based on the previous average and current size of the queue. The formula is:

average = (old_average * (1-2 -x)) + (current_queue_size * 2 -x)

where *x* is the exponential weight factor.

For high values of *x*, the previous average becomes more important. A large factor smooths out the peaks and lows in queue length. The average queue size is unlikely to change very quickly, avoiding a drastic change in size. The WRED process is slow to start dropping packets, but it may continue dropping packets for a time after the actual queue size has fallen below the minimum threshold. The slow-moving average accommodates temporary bursts in traffic.

**Note**

- The exponential weight factor, *x*, is fixed and is not user configurable.
- If the value of *x* gets too high, WRED does not react to congestion. Packets are sent or dropped as if WRED were not in effect.
- If the value of *x* gets too low, WRED overreacts to temporary traffic bursts and drops traffic unnecessarily.

For low values of *x*, the average queue size closely tracks the current queue size. The resulting average may fluctuate with changes in the traffic levels. In this case, the WRED process responds quickly to long queues. Once the queue falls below the minimum threshold, the process stops dropping packets.

# Configure Weighted Random Early Detection

This configuration task is similar to that used for RED except that the **random-detect** command is not configured in RED.

### Restrictions

- You cannot use the **random-detect** command in a class configured with the **priority** command, because WRED cannot be configured in a class that has been set for priority queueing (PQ).
- When configuring the **random-detect** command, you must configure one of the following commands: **shape average**, **bandwidth**, and **bandwidth remaining**.

### Configuration Example

You have to accomplish the following to complete the random early detection configuration:

1. Creating (or modifying) a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Enabling WRED by specifying the match criteria (discard-class).
4. (Optional) Specifying the bandwidth allocated for a class belonging to a policy map or specifying how to allocate leftover bandwidth to various classes.
5. (Optional) Shaping traffic to the specified bit rate or a percentage of the available bandwidth.
6. (Optional) Changing queue limit to fine-tune the amount of buffers available for each queue.
7. Attaching a policy map to an output interface to be used as the service policy for that interface.

```
Router# configure
Router(config)# class-map qos-1
Router(config-cmap)# match traffic-class 1
Router(config-cmap)# commit
Router(config-pmap)# exit

Router# configure
Router(config)# policy-map test-wred-1
Router(config-pmap)# class qos-1
Router(config-pmap-c)# random-detect default
Router(config-pmap-c)# random-detect discard-class 0 10 ms 500 ms
Router(config-pmap-c)# shape average percent 10
Router(config-pmap-c)# commit

Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy output policy1
Router(config-if)# commit
```

### Running Configuration

```
class-map qos-1
 match traffic-class 1
commit
```

```
policy-map test-wred-1
 class qos-1
  random-detect default
  random-detect discard-class 0 10 ms 500 ms
  shape average percent 10
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE 0/6/0/18
 service-policy output test-wred-1
!
```

## Verification

```
Router# show qos int hundredGigE 0/0/0/20 output

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/20 ifh 0x38  -- output policy
NPU Id:                        0
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Policy Name:                   test-wred-1
VOQ Base:                      1184
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   qos-1
Egressq Queue ID                         =   1185 (LP queue)
Queue Max. BW.                           =   10000152 kbps (10 %)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Guaranteed service rate                  =   10000000 kbps
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   1250000896 bytes / 1000 ms (default)

WRED profile for Discard_Class 0
WRED Min. Threshold                      =   12499968 bytes (10 ms)
WRED Max. Threshold                      =   624999936 bytes (500 ms)

Default RED profile
WRED Min. Threshold                      =   7499776 bytes (6 ms)
WRED Max. Threshold                      =   12499968 bytes (10 ms)

WRED ECN                                 =   Disabled

Level1 Class                             =   class-default
Egressq Queue ID                         =   1184 (Default LP queue)
Queue Max. BW.                           =   no max (default)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Guaranteed service rate                  =   50000000 kbps
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   62499840 bytes / 10 ms (default)
WRED not configured for this class
```

## Related Topics

- Weighted Random Early Detection, on page 83

- Configure Random Early Detection, on page 81

### Associated Commands

- random-detect

## Configure WRED Counters by Class

*Table 14: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Configure WRED Counters by Class | Release 7.4.1 | This feature enables the display of WRED statistics per class, thus providing a more accurate and granular statistics profile for packet drops. Such insight allows you to monitor, anticipate, and avoid congestion at common bottlenecks on your network.<br>This functionality introduces the hw-module profile qos wred-stats-enable command and modifies the output of the **show policy-map interface** command. |

- **Statistics profile**—Up until now, running the **show policy-map interface** command displayed the drop statistics per class under tail drop. Which meant that no matter what drop mechanism you deployed (Tail Drop and the FIFO Queue, on page 78, Random Early Detection and TCP, on page 80, or Weighted Random Early Detection, on page 83), the statistics profile that displayed was a number aggregated under the tail drop entry. Given that WRED provides a far more granular control over packet drops than the other mechanisms, the generic statistics profile didn't provide you with the validation that you required for your WRED actions.
- **WRED counters by class**—This functionality introduces a new command, **hw-module profile qos wred-stats-enable**. Configuring this command enables the display of WRED and RED statistics per class, thus providing a more accurate and granular statistic for packet drops.
- **Why WRED counters by class**—Prior to this functionality, you did not have visibility into the WRED drop counters and had no way to validate WRED actions. With this improvement, the **show policy-map interface** command provides the much-needed visibility by clearly depicting the drop counters per class for WRED. This visibility gives you the confidence that WRED is indeed in action and is providing the requisite congestion avoidance.
- **Guidelines and limitations**
  - WRED max-drop threshold statistics are unavailable.
  - WRED statistics are collected separately for each drop precedence value.
- **WRED counters support cheat-sheet**—Here's a quick look at some key support areas and their details for WRED counters by class.

| Support | Details |
|---|---|
| Line card | Supported on Cisco NC57 line cards with external Ternary Content-Addressable Memory (TCAM) operating in native mode. |
| Statistics | WRED drop count is available for all the supported discard-values (from 0 through 2). |

- **Configure WRED counters by class**—To configure WRED counters by class you must:
  - Configure the **hw-module profile** for WRED statistics. Manually reload the chassis to activate the profile.
  - Configure a class map.
  - Create a service policy map.
  - Attach the service policy to an interface.

```
/*Configure hw-module profile for wred stats*/
Router#conf
Router(config)#hw-module profile qos wred-stats-enable
Router(config)#commit
Router#reload

/*Configure a class map*/
Router(config)#class-map TC3
Router(config-cmap)#match traffic-class 3
Router(config-cmap)#commit
Router(config-cmap)#end-class-map


/*Create a service policy map*/
Router(config)#policy-map EGRESS_WRED
Router(config-pmap)#class TC3
Router(config-pmap-c)#shape average 1 gbps
Router(config-pmap-c)#random-detect discard-class 0 1 ms 2 ms
Router(config-pmap-c)#random-detect discard-class 1 3 ms 5 ms
Router(config-pmap-c)#random-detect discard-class 2 10 ms 20 ms
Router(config-pmap-c)#commit
Router(config-pmap)end-policy-map


/*Attach the service policy to an interface*/
Router(config)#int hundredGigE 0/7/0/2
Router(config-if)#service-policy output EGRESS_WRED
Router(config-if)#commit
```

- **Running Configuration**

```
class-map match-any TC3
  match traffic-class 3
  end-class-map
!
policy-map EGRESS_WRED
  class TC3
    shape average 1 gbps
    random-detect discard-class 0 1 ms 2 ms
```

```
    random-detect discard-class 1 3 ms 5 ms
    random-detect discard-class 2 10 ms 20 ms
!
    class class-default
!
    end-policy-map
!
interface HundredGigE0/7/0/2
 service-policy output EGRESS_WRED
 ipv4 address 192.168.0.1 255.255.255.0
!
```

- **Verification**

```
Router#show qos int hundredGigE 0/7/0/2 output
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/7/0/2 ifh 0xe000088  -- output policy
NPU Id:                        0
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Policy Name:                   EGRESS_WRED
SPI Id:                        0x0
VOQ Base:                      1160
PFC enabled:                   0
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   TC3
Egressq Queue ID                          =   1163 (LP queue)
Queue Max. BW.                            =   1000000 kbps (1 gbits/sec)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   1000000 kbps
Peak burst                                =   36864 bytes (default)
TailDrop Threshold                        =   4999168 bytes / 40 ms (default)

WRED profile for Discard_Class 0
WRED Min. Threshold                       =   124928 bytes (1 ms)
WRED Max. Threshold                       =   249856 bytes (2 ms)

WRED profile for Discard_Class 1
WRED Min. Threshold                       =   374784 bytes (3 ms)
WRED Max. Threshold                       =   624896 bytes (5 ms)

WRED profile for Discard_Class 2
WRED Min. Threshold                       =   1249792 bytes (10 ms)
WRED Max. Threshold                       =   2499840 bytes (20 ms)

Default RED profile
WRED Min. Threshold                       =   256 bytes (0 ms)
WRED Max. Threshold                       =   256 bytes (0 ms)

WRED ECN                                  =   Disabled

Level1 Class                              =   class-default
Egressq Queue ID                          =   1160 (Default LP queue)
Queue Max. BW.                            =   no max (default)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   50000000 kbps
Peak burst                                =   36864 bytes (default)
TailDrop Threshold                        =   62499840 bytes / 10 ms (default)
WRED not configured for this class
```

- **show policy-map interface output for statistics verification**

```
Router#show policy-map int hundredGigE 0/7/0/2 output

HundredGigE0/7/0/2 output: EGRESS_WRED

Class TC3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            1151236/1151236000           0
    Transmitted         :             578807/578807000            0
    Total Dropped       :             572429/572429000            0
  Queueing statistics
    Queue ID                            : 1163
    Taildropped(packets/bytes)          : 572429/572429000

    WRED profile for Discard Class 0
    RED Transmitted (packets/bytes)            : 0/0
    RED random drops(packets/bytes)            : 0/0
    RED maxthreshold drops(packets/bytes)      : N/A
    RED ecn marked & transmitted(packets/bytes): N/A
    WRED profile for Discard Class 1
    RED Transmitted (packets/bytes)            : 0/0
    RED random drops(packets/bytes)            : 0/0
    RED maxthreshold drops(packets/bytes)      : N/A
    RED ecn marked & transmitted(packets/bytes): N/A
    WRED profile for Discard Class 2
    RED Transmitted (packets/bytes)            : 578807/578807000
    RED random drops(packets/bytes)            : 572429/572429000
    RED maxthreshold drops(packets/bytes)      : N/A
    RED ecn marked & transmitted(packets/bytes): N/A
    RED profile (default)
    RED Transmitted (packets/bytes)            : 0/0
    RED random drops(packets/bytes)            : 0/0
    RED maxthreshold drops(packets/bytes)      : N/A
    RED ecn marked & transmitted(packets/bytes): N/A
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                  0/0                   0
    Transmitted         :                  0/0                   0
    Total Dropped       :                  0/0                   0
  Queueing statistics
    Queue ID                            : 1160
    Taildropped(packets/bytes)          : 0/0
```

- **Clear WRED statistics on an interface**

  Run the following command to clear WRED statistics on an interface.

```
Router#clear qos counters int hundredGigE 0/7/0/2 output
```

### Associated Commands

hw-module profile qos wred-stats-enable

# Explicit Congestion Notification (ECN)

Weighted Random Early Detection (WRED) is implemented at the core routers of a network. Edge routers assign IP precedences to packets, as the packets enter the network. With WRED, core routers then use these precedences to determine how to treat different types of traffic. WRED provides separate thresholds and weights for different IP precedences, enabling the network to provide different qualities of service, in regard

to packet dropping, for different types of traffic. Standard traffic may be dropped more frequently than premium traffic during periods of congestion.

ECN is an extension to WRED. ECN marks packets instead of dropping them when the average queue length exceeds a specific threshold value. When configured, ECN helps routers and end hosts to understand that the network is congested and slow down sending packets. However, if the number of packets in the queue is above the maximum threshold, packets are dropped based on the drop probability.

WRED starts dropping packets probabilistically before a queue becomes full, which means that it works during enqueuing. ECN marking with WRED (ECN-WRED) takes place as packets are about to be transmitted from the queue (during the dequeue process), when the router decides whether to forward, drop, or mark packets based on the congestion state of the queue. Also, WRED and ECN don't work together.

RFC 3168, *The Addition of Explicit Congestion Notification (ECN) to IP*, states that with the addition of active queue management (for example, WRED) to the Internet infrastructure, routers are no longer limited to packet loss as an indication of congestion.

**Note** You cannot use this feature when you have set qos-group or mpls experimental along with a traffic class in the ingress policy.

### Implementing ECN

Implementing ECN requires an ECN-specific field that has 2 bits—the ECN-capable Transport (ECT) bit and the CE (Congestion Experienced) bit—in the IP header. The ECT bit and the CE bit can be used to make four ECN field combinations of 00 to 11. The first number is the ECT bit and the second number is the CE bit.

***Table 15: ECN Bit Setting***

| ECT Bit | CE Bit | Combination Indicates |
|---|---|---|
| 0 | 0 | Not-ECN-capable. |
| 0 | 1 | Endpoints of the transport protocol are ECN-capable. |
| 1 | 0 | Endpoints of the transport protocol are ECN-capable. |
| 1 | 1 | Congestion experienced. |

The ECN field combination 00 indicates that a packet is not using ECN. The ECN field combinations 01 and 10—Called ECT(1) and ECT(0), respectively—are set by the data sender to indicate that the endpoints of the transport protocol are ECN-capable. Routers treat these two field combinations identically. Data senders can use either one or both of these two combinations. The ECN field combination 11 indicates congestion to the endpoints. Packets arriving a full queue of a router will be dropped.

### Packet Handling When ECN Is Enabled

When the number of packets in the queue is below the minimum threshold, packets are transmitted.

If the number of packets in the queue is above the maximum threshold:

- For traffic flows that are not ECN-enabled in only WRED-configured queues, packets are tail-dropped after the queue size exceeds the WRED maximum threshold.
- For traffic flows that are ECN-enabled in only WRED-configured queues, packets are tail-dropped when the queue size exceeds the tail-drop threshold.
- When you configure ECN remarking on your router, incoming packets with ECT bit settings 0 or 1 are marked as CE.

**Note** When the number of packets reaches the queue limit, all packets are dropped. This is the identical treatment that a packet receives when you enable WRED without ECN configured on the router.

Three different scenarios arise if the number of packets in the queue is between the minimum threshold and the maximum threshold:

- If the ECN field on the packet indicates that the endpoints are ECN-capable (that is, the ECT bit is set to 1 and the CE bit is set to 0, or the ECT bit is set to 0 and the CE bit is set to 1)—and the WRED algorithm determines that the packet should have been dropped based on the drop probability—the ECT and CE bits for the packet are changed to 1, and the packet is transmitted. This happens because ECN is enabled and the packet gets marked instead of dropped.
- If the ECN field on the packet indicates that neither endpoint is ECN-capable (that is, the ECT bit is set to 0 and the CE bit is set to 0), packet is dropped once the queue limit is reached.
- If the ECN field on the packet indicates that the network is experiencing congestion (that is, both the ECT bit and the CE bit are set to 1), the packet is transmitted. No further marking is required.

**Note** Applicable until Cisco IOS XR Release 7.11.1: When the incoming IP traffic with ECN bits set to 10 passes through the ingress qos-policy-map that has the class-map definition of `set DSCP/PREC <value>`, then the ECN bits in the IP header gets modified to 01. This is applicable to NC57 routers operating in the Native mode.

Applicable from Cisco IOS XR Release 7.11.2 : NC57 line cards operating in compatibility mode do not support ECN-bit preservation for terminated VXLAN packets when you configure ACL compression for ingress traffic.

### Limitations

- ECN configuration is not supported per discard class but for all packets enqueued to a VOQ.
- **The following limitation is not applicable to NC57 routers:**

  **SRv6 Disposition Node** — For the incoming traffic that has the outer SRv6 IP header having the ECN bits as non-zero value and the inner IP header ECN bits as zero value, then the ECN bits of the inner IP header will be overwritten to 01.
- **The following limitation is specific to NC57 routers operating in Native mode.**

**Note** ECN remarking functionality is not supported with SRv6 tunnels. The inner header in the following scenario is IP and the outer header is SRv6.

**SRv6 Imposition Node** — The ECN bits in outer header traffic class is set to 0 when SRv6 encapsulation is set to traffic-class propagate along with Ingress MQC set to any DSCP/PREC value. This scenario does not impact the inner header ECN bits.

**SRv6 Disposition Node** — The ECN bits in the inner packet header is reset to 0 when the Ingress MQC is set to any DSCP/PREC value.

### Configuration Example

```
Router# configure
Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth percent 50
Router(config-pmap-c)# random-detect 1000 packets 2000 packets
Router(config-pmap-c)# random-detect ecn
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# commit
```

### Verification

Use the **show policy-map interface** to verify the configuration.

```
Router# show policy-map interface tenGigE 0/0/0/6 output
TenGigE0/0/0/6 output: pm-out-queue

Class cm-tc-1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            85528554/87581239296          4830672
    Transmitted         :            16240891/16630672384          966585
    Total Dropped       :            69287663/70950566912          3864087
  Queueing statistics
    Queue ID                              : 1113
    Taildropped(packets/bytes)            : 69287663/70950566912

    WRED profile for
    RED Transmitted (packets/bytes)             : N/A
    RED random drops(packets/bytes)             : N/A
    RED maxthreshold drops(packets/bytes)       : N/A
    RED ecn marked & transmitted(packets/bytes): N/A
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                              : 1112
    Taildropped(packets/bytes)            : 0/0
```

**Note** No ECN-specific statistics are displayed in the show output for this command. ECN is enabled if all rows indicate **N/A**, as highlighted in the example.

# Configure ECN Maximum Marking Probability

*Table 16: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Configure ECN Maximum Marking Probability | Release 7.3.3 | This feature allows you to configure percentages for ECN maximum marking probability after considering your network congestion and end-to-end application needs. With this flexibility, you can fine-tune the network's reactivity to congestion according to the tolerance levels of your end-to-end applications.<br><br>By controlling the probability of packets being ECN marked at the ECN maximum threshold, you can optimize the congestion notification from the traffic destination to the traffic source. This optimization avoids aggressive throttling at the source (from too high ECN maximum marking probability) or traffic drops in transit due to significantly less throttling at the source (from too low ECN maximum marking probability).<br><br>This functionality modifies the existing **random-detect** command. |

- **First, something about packet drop probability and Explicit Congestion Notification (ECN)**—The probability that a packet will be ECN-marked is based on the minimum threshold, maximum threshold, and maximum mark probability percent.

  When the average queue size is above the minimum threshold, ECN starts marking packets to indicate congestion. When an ECN-marked packet reaches the end host, the host transmits confirmation of receipt of this packet to the transmitting host. The confirmation indicates to the transmitting host that congestion has begun and that the transmission or packet rate must be reduced.

  Maximum mark probability is the percentage of packets marked when the average queue size is at the maximum threshold. The ECN mark probability increases linearly from 0 when queue length is less than minimum threshold to 10% when the queue length equals or exceeds the maximum threshold. For example, when the average queue size is halfway between minimum-threshold and maximum-threshold the mark probability is 5%, and half the packets transmitted will be ECN marked.

- **Configurable ECN Maximum Marking Probability**—So far, the ECN maximum marking probability was not user-configurable and was fixed at 10%. The arrangement of preset marking probabilities meant that the router couldn't adapt the congestion notification to network characteristics and application tolerance. With the flexibility to configure maximum ECN marking probability, you can configure the

maximum marking probability to a higher or lower percent, depending on the requirements of congestion handling in the network for a given traffic flow and application type. This also allows you to choose a consistent ECN maximum marking probability across all network elements instead of operating them at the factory default values. The following table maps the linear increase in mark probability percentage as the queue length conditions change from 0 to the tail drop threshold.

| Queue length conditions | Mark probability increases linearly... |
|---|---|
| Queue length is less than the minimum threshold value | from 0 |
| Average queue length is between minimum-threshold and maximum-threshold | corresponding to the average queue length:<br>(maximum mark probability) * (average queue length –minimum threshold value)/(maximum threshold value – minimum threshold value) |
| Queue length equals the maximum threshold | to the configured maximum probability percentage |
| maximum threshold value < queue length < tail drop threshold | to 100% |

- **Some FAQs**
  - **What if I set the ECN marking probability percentage to 10%?**

    In such a case, there's no change in traffic behavior from what it was for previously ECN-marked packets. Tail drop comes into play when the queue length exceeds the tail drop threshold.

  - **How does traffic behave beyond my configured value of maximum ECN marking probability?**

    Let's say you've set the maximum ECN marking probability to 5%. Any more increase in the average queue length beyond maximum ECN threshold shifts the marking probability to 100%, and once the queue length exceeds the tail drop threshold, Tail Drop and the FIFO Queue, on page 78 comes into effect.

  - **How do I enable this functionality?**

    Create a policy with **random-detect ecn** and **random-detect** <*min-threshold*> <*max-threshold*> **probability percent** <*value*> configured on an interface.

  - **How do I disable this functionality?**

    To disable this functionality, do not configure the **probability percent** value.

  - **Are the user-configured values programmed precisely in the hardware?**

    The user-configured values are rounded off to the nearest granularity during programming in the hardware. The round-off is minimal when the 'Pmax/(ECNmax-ECNmin)' ratio is a power of 2, where Pmax is the user-configured maximum marking probability in percent and ECNmin and ECNmax are the user-configured minimum and maximum ECN thresholds.

    The **show qos interface** command output displays the actual value programmed in hardware and the user-configured value for reference.

- **Important Guidelines**

- **Line Cards that support this functionality**:
  - NC55-36X100G
  - NC55-18H18F
  - NC55-24X100G-SE
  - NC55-36X100G-S
  - NC55-24H12F-SE
  - NC55-36X100G-A-SE
  - NCS-55A1-36H-SE-S
  - NCS-55A1-36H-S
  - NCS-55A1-24H
  - NCS-55A1-48Q6H
- **The following interface types** support this functionality:
  - Physical interfaces
  - Bundle interfaces
  - Subinterfaces
  - Bundle subinterfaces
- This functionality is **supported for all interface speeds**.
- If you have a **policy map with one or multiple classes with maximum ECN marking probability enabled**, you can:
  - Apply the map to any of the supported interface types.
  - Remove the map from any of the supported interface types.
  - Modify the map while you're attaching it to multiple interfaces.
- The probability percentage option is supported only with **random-detect ecn** configured in the same class. Else, the policy is rejected when applied on an interface

- **Configuring ECN Maximum Marking Probability**—After you have configured Explicit Congestion Notification (ECN), on page 89, the **random-detect** command displays an option to add the probability in percent.

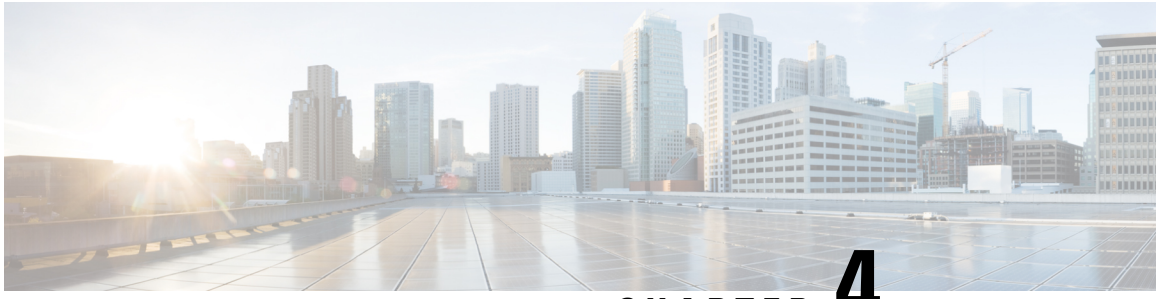
### Configuration Example

```
Router#configure
Router(config)#policy-map policy1
Router(config-pmap)#class class1
Router(config-pmap-c)#bandwidth percent 50
Router(config-pmap-c)#random-detect 1 mbytes 2 mbytes probability percent 5
Router(config-pmap-c)#random-detect ecn
Router(config-pmap-c)#commit
```

### Verification

Run the **show qos interface** command to view the configured ECN Maximum Marking Probability value (displayed as **ECN Mark Probability**).

```
Router#sh qos interface FourHundredGigE 0/1/0/4 output


NOTE:- Configured values are displayed within parentheses
Interface FourHundredGigE0/1/0/4 ifh 0x800150  -- output policy
------------------------------------------------------------------------------
.
.
 Level2 Class                              =   LOSSLESSTCP_Egress
   Egressq Queue ID                          =   5475 (LP queue)
   Queue Max. BW.                            =   no max (default)
   Inverse Weight / Weight                   =   3 / (20)
   Guaranteed service rate                   =   78329670 kbps
   TailDrop Threshold                        =   390070272 bytes / 40 ms (default)

   Default RED profile
   WRED Min. Threshold                       =   9787392 bytes (1 ms)
   WRED Max. Threshold                       =   19580928 bytes (2 ms)
   ECN Mark Probability                     =   4.99 (5)

   WRED ECN                                  =   Enabled
```



CHAPTER 4

# Priority Flow Control Overview

*Table 17: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Priority Flow Control | Release 7.3.2 | Previously available in Release 6.6.3, this feature is a link-level flow control mechanism that enables you to selectively pause traffic based on its class of service (CoS). |

Priority flow control (PFC; IEEE 802.1Qbb), which is also referred to as Class-based Flow Control (CBFC) or Per Priority Pause (PPP), is a mechanism that prevents frame loss that is due to transient congestion. PFC is similar to 802.3x Flow Control (pause frames) or link-level flow control (LLFC). However, instead of pausing all traffic on a link, PFC functions on a per class-of-service (CoS) basis.

During congestion, PFC sends a pause frame that indicates which CoS value must be paused. When the congestion is mitigated, the router stops sending the PFC frames to the upstream node.

For details about the Priority Flow Control feature, see the *Interface and Hardware Component Configuration Guide for Cisco NCS 5500 Series Routers*.


- Restrictions, on page 97
- Configuring Priority Flow Control Thresholds, on page 99
- Priority Flow Control Watchdog Overview, on page 105
- Configure a Priority Flow Control Watchdog Interval, on page 105
- Monitoring and Logging Packet Drops on Lossless PFC-Enabled Queues, on page 108


## Restrictions

The following restrictions apply while configuring PFC thresholds, PFC watchdog and QoS policies in PFC profile.

- The PFC feature is only supported in the non-HQoS profile.
- The PFC feature is only supported on the following line card or fixed chassis PIDs of the NCS5500 Series:
  - NC55-36X100G

- NC55-18H18F
- NC55-24X100G-SE
- NC55-36X100G-S
- NC55-24H12F-SE
- NC55-36X100G-A-SE
- NCS-55A1-36H-SE-S
- NCS-55A1-36H-S
- NCS-55A1-24H
- NCS-55A1-48Q6H

- When PFC is enabled, only two parameter scheduling is supported. So, the egress actions could have either priority or weighted fair queue (WFQ) scheduling apart from a shaper action. For WFQ scheduling, either bandwidth remaining ratio (BRR) or bandwidth is supported and a mix of the two in the same policy-map is rejected. With PFC, BRR has a weight range of 1 to 256 as against 1 to 4096 for the non PFC, non-HQoS case.
- The hw-module profile for enabling PFC per queue and defining PFC Tx thresholds are global configurations per line card location. So, only one set of pause, resume thresholds and headroom sizes per traffic-class can be configured for all ports on a given line card location. With the introduction of this new configuration model, the pause action under an egress policy class-map is deprecated.
- While a line card reload is required for any addition or removal of traffic classes in the hw-module profile, you can update the pause, resume thresholds and buffer / headroom sizes on already configured traffic-classes without requiring a line card reload.
- PFC watchdog (PFCWD) interval has a minimum granularity of 100ms. When you configure higher values, they are rounded up to the nearest multiple of 100ms.
- There is no XML schema or yang model for the new hw-module profile. Only CLI configuration and show commands are available.
- In the PFC profile, only monotonically increasing priority levels with traffic-class are supported. For example, TC7 should be p1, TC6 should be p2, and so on, until TC1 is p7 with class-default being lowest priority. Rate classes can be mixed anywhere.
- When an egress queue is disabled as part of a watchdog queue shutdown action and once the pause frames stop, the queue drains the enqueued packets on the wire while it waits to be restored back.
- If PFCWD is disabled and a traffic-class experiences sustained PFC storm beyond 7s, then the traffic on that queue continues to be dropped even when the PFC storm stops. Traffic needs to be stopped and restarted or all the incoming ports involved in sending traffic to this egress port needs to be flapped (shut and no shut) to manually recover the egress port.
- The class-map actions of queue-limit and WRED thresholds without ECN enabled have no impact on PFC-enabled queues. Queue-limit is redundant because packet is buffered based on pause or x-off and headroom configured per source port.
- Even when PFCWD shuts down an egress queue experiencing storm, if the ingress traffic was already causing a congestion due to the application of sub-rate shapers or BRR policy, then the ingress buffers build up and PFC Tx is still generated towards the sender. In other words, PFCWD only removes the

source of a storm. It does not suppress PFC Tx generated from sustained congestion due to user configuration.

- In the PFC profile, there is a 25% reduction in available buffer descriptors. Further, 20% of the remaining buffer space is reserved for headroom. This leads to only 80% of the 75% buffers being available in the shared pool for all ingress VoQs on that NP core.

# Configuring Priority Flow Control Thresholds

You can set values for pause threshold (x-off), resume threshold (x-on) and headroom for a traffic class on all PFC enabled ports on a given line card location using the hw-module profile priority-flow-control command.

The existing queue-limit for that traffic class on an egress queuing policy on that line card will have no impact and the effective queue limit is pause threshold + headroom.

**Note** For optimal functionality in hardware, for a given traffic class, the resume threshold should not be more than 10% of the pause threshold and the headroom should be at least 100KB.

For details on how to configure PFC on an interface, see the *Interface and Hardware Component Configuration Guide for Cisco NCS 5500 Series Routers*.

### Configuration Example

Apart from the line card level PFC parameters configured through the hw-module profile, PFC Receive functionality requires an egress policy to be applied and egress traffic to be mapped to unique queues per CoS value on all PFC enabled interfaces.

You can accomplish this with the following sequence of configuration.

1. Configure ingress classification policy to map incoming traffic to the right CoS / priority queue.
2. Configure egress queuing policy with the relevant shaping, priority and weighted fair queue scheduling actions
3. Attach the ingress policy on all interfaces where traffic can come in.
4. Attach the egress policy on all PFC enabled interfaces in the system.
5. Configure hw-module priority-flow-control profile and configure pause, resume and headroom for all PFC traffic-classes on all line card locations requiring PFC feature.

Reload the line card for all traffic-class addition and deletions. Parameters within an already configured traffic-class can be edited 'in place' without requiring a line card reload.

```
Hw-module configuration:
===================
RP/0/RP1/CPU0:NCS5504(config)#hw-module profile priority-flow-control location 0/0/CPU0
tRP/0/RP1/CPU0:NCS5504(config-pfc-loc)# traffic-class 3 pause-threshold 403200 bytes
resume-threshold 40320 bytes headroom 441600 bytes
RP/0/RP1/CPU0:NCS5504(config-pfc-loc)# traffic-class 4 pause-threshold 403200 bytes
resume-threshold 40320 bytes headroom 441600 bytes
RP/0/RP1/CPU0:NCS5504(config-pfc-loc)#

Class-map configuration:
```

```
===================
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any MAIN_IN_CMAP_1
match prRP/0/RP1/CPU0:NCS5504(config-cmap)# match precedence 1
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any MAIN_IN_CMAP_2
RP/0/RP1/CPU0:NCS5504(config-cmap)# match precedence 2
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any MAIN_IN_CMAP_3
RP/0/RP1/CPU0:NCS5504(config-cmap)# match precedence 3
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any MAIN_IN_CMAP_4
RP/0/RP1/CPU0:NCS5504(config-cmap)# match precedence 4
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any MAIN_IN_CMAP_5
RP/0/RP1/CPU0:NCS5504(config-cmap)# match precedence 5
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any MAIN_IN_CMAP_6
RP/0/RP1/CPU0:NCS5504(config-cmap)# match precedence 6
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any MAIN_IN_CMAP_7
RP/0/RP1/CPU0:NCS5504(config-cmap)# match precedence 7
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any PFC_OUT_CMAP_1
RP/0/RP1/CPU0:NCS5504(config-cmap)# match traffic-class 1
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any PFC_OUT_CMAP_2
RP/0/RP1/CPU0:NCS5504(config-cmap)# match traffic-class 2
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any PFC_OUT_CMAP_3
RP/0/RP1/CPU0:NCS5504(config-cmap)# match traffic-class 3
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any PFC_OUT_CMAP_4
RP/0/RP1/CPU0:NCS5504(config-cmap)# match traffic-class 4
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any PFC_OUT_CMAP_5
RP/0/RP1/CPU0:NCS5504(config-cmap)# match traffic-class 5
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any PFC_OUT_CMAP_6
RP/0/RP1/CPU0:NCS5504(config-cmap)# match traffic-class 6
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#class-map match-any PFC_OUT_CMAP_7
RP/0/RP1/CPU0:NCS5504(config-cmap)# match traffic-class 7
RP/0/RP1/CPU0:NCS5504(config-cmap)# end-class-map
RP/0/RP1/CPU0:NCS5504(config)#!
RP/0/RP1/CPU0:NCS5504(config)#

Ingress Policy-map configuration:
=========================
RP/0/RP1/CPU0:NCS5504(config)#policy-map MAIN_OUT_TC_SUPPORTING_UUT
RP/0/RP1/CPU0:NCS5504(config-pmap)# class MAIN_IN_CMAP_1
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# set traffic-class 1
```

```
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class MAIN_IN_CMAP_2
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# set traffic-class 2
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class MAIN_IN_CMAP_3
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# set traffic-class 3
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class MAIN_IN_CMAP_4
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# set traffic-class 4
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class MAIN_IN_CMAP_5
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# set traffic-class 5
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class MAIN_IN_CMAP_6
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# set traffic-class 6
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class MAIN_IN_CMAP_7
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# set traffic-class 7
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)#
RP/0/RP1/CPU0:NCS5504(config-pmap-c)#


Egress policy-map configuration:
============================
RP/0/RP1/CPU0:NCS5504(config)#policy-map PFC_UUT_bwrr_microsoft
asRP/0/RP1/CPU0:NCS5504(config-pmap)# class PFC_OUT_CMAP_1
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# bandwidth remaining ratio 5
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# queue-limit 192 us
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class PFC_OUT_CMAP_2
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# bandwidth remaining ratio 5
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# queue-limit 192 us
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class PFC_OUT_CMAP_3
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# bandwidth remaining ratio 20
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# random-detect ecn
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# random-detect 224 kbytes 275 kbytes
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class PFC_OUT_CMAP_4
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# bandwidth remaining ratio 20
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# random-detect ecn
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# random-detect 224 kbytes 275 kbytes
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class PFC_OUT_CMAP_5
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# bandwidth remaining ratio 20
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# queue-limit 192 us
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class PFC_OUT_CMAP_6
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# bandwidth remaining ratio 1
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# queue-limit 192 us
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class PFC_OUT_CMAP_7
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# shape average percent 10
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# priority level 1
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# queue-limit 192 us
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# class class-default
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# bandwidth remaining ratio 20
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# queue-limit 192 us
RP/0/RP1/CPU0:NCS5504(config-pmap-c)# !

Applying policy to ingress and egress interface:
========================================
```

```
RP/0/RP1/CPU0:NCS5504#show running-config interface hundredGigE 0/0/0/0
interface HundredGigE0/0/0/0
 service-policy input MAIN_OUT_TC_SUPPORTING_UUT
 ipv4 address 100.1.9.1 255.255.255.0
 ipv6 address 100:1:9::1/96
 priority-flow-control mode on

interface HundredGigE0/1/0/19
 service-policy output PFC_ECN_UUT_bwrr_microsoft
 ipv4 address 10.1.9.1 255.255.255.0
 ipv6 address 10:1:9::1/96
 priority-flow-control mode on
!
```

### Running Configuration

```
RP/0/RP1/CPU0:NCS5504# show running-config hw-module profile priority-flow-control location
 0/0/CPU0
Tue Oct 13 10:31:58.035 UTC
hw-module profile priority-flow-control location 0/0/CPU0
traffic-class 3 pause-threshold 403200 bytes resume-threshold 40320 bytes headroom 441600
bytes
traffic-class 4 pause-threshold 403200 bytes resume-threshold 40320 bytes headroom 441600
bytes
!

RP/0/RP1/CPU0:NCS5504#show running-config class-map
Tue Oct 13 10:32:02.400 UTC
class-map match-any MAIN_IN_CMAP_1
match precedence 1
end-class-map
!
class-map match-any MAIN_IN_CMAP_2
match precedence 2
end-class-map
!
class-map match-any MAIN_IN_CMAP_3
match precedence 3
end-class-map
!
class-map match-any MAIN_IN_CMAP_4
match precedence 4
end-class-map
!
class-map match-any MAIN_IN_CMAP_5
match precedence 5
end-class-map
!
class-map match-any MAIN_IN_CMAP_6
match precedence 6
end-class-map
!
class-map match-any MAIN_IN_CMAP_7
match precedence 7
end-class-map
!
class-map match-any PFC_OUT_CMAP_1
match traffic-class 1
end-class-map
!
class-map match-any PFC_OUT_CMAP_2
match traffic-class 2
end-class-map
!
```

```
class-map match-any PFC_OUT_CMAP_3
match traffic-class 3
end-class-map
!
class-map match-any PFC_OUT_CMAP_4
match traffic-class 4
end-class-map
!
class-map match-any PFC_OUT_CMAP_5
match traffic-class 5
end-class-map
!
class-map match-any PFC_OUT_CMAP_6
match traffic-class 6
end-class-map
!
class-map match-any PFC_OUT_CMAP_7
match traffic-class 7
end-class-map
!

RP/0/RP1/CPU0:NCS5504#show running-config policy-map PFC_UUT_bwrr_microsoft
Tue Oct 13 10:32:18.009 UTC
policy-map PFC_UUT_bwrr_microsoft
class PFC_OUT_CMAP_1
bandwidth remaining ratio 5
queue-limit 192 us
!
class PFC_OUT_CMAP_2
bandwidth remaining ratio 5
queue-limit 192 us
!
class PFC_OUT_CMAP_3
bandwidth remaining ratio 20
random-detect ecn
random-detect 224 kbytes 275 kbytes
!
class PFC_OUT_CMAP_4
bandwidth remaining ratio 20
random-detect ecn
random-detect 224 kbytes 275 kbytes
!
class PFC_OUT_CMAP_5
bandwidth remaining ratio 20
queue-limit 192 us
!
class PFC_OUT_CMAP_6
bandwidth remaining ratio 1
queue-limit 192 us
!
class PFC_OUT_CMAP_7
shape average percent 10
priority level 1
queue-limit 192 us
!
class class-default
bandwidth remaining ratio 20
queue-limit 192 us
!
end-policy-map
!

RP/0/RP1/CPU0:NCS5504#show running-config policy-map MAIN_OUT_TC_SUPPORTING_UUT
Tue Oct 13 10:32:31.430 UTC
```

```
policy-map MAIN_OUT_TC_SUPPORTING_UUT
class MAIN_IN_CMAP_1
set traffic-class 1
!
class MAIN_IN_CMAP_2
set traffic-class 2
!
class MAIN_IN_CMAP_3
set traffic-class 3
!
class MAIN_IN_CMAP_4
set traffic-class 4
!
class MAIN_IN_CMAP_5
set traffic-class 5
!
class MAIN_IN_CMAP_6
set traffic-class 6
!
class MAIN_IN_CMAP_7
set traffic-class 7
!
class class-default
!
end-policy-map
!
```

### Verification

Incoming PFC frames are displayed on the PFC Rx statistics that are shown in the command below. If the interface is enabled for PFC and has an egress queuing policy applied to it, then the queue corresponding to the incoming CoS value on the PFC frames is paused

When PFC pause thresholds are crossed, PFC Tx frames are sent out of the PFC enabled interfaces sourcing such traffic on that network processing core. This can be verified against the PFC Tx statistics that are shown on this command. After the congestion condition ceases and the buffer resource usage goes below the PFC resume thresholds, a PFC resume frame is sent and the PFC Tx frames are stopped.

```
RP/0/RP1/CPU0:NCS5504#show controllers hundredGigE 0/0/0/0  priority-flow-control statistics

Mon Oct 12 12:22:39.362 UTC

Priority flow control information for interface HundredGigE0/0/0/0:


Priority Flow Control:
    Total Rx PFC Frames: 0
    Total Tx PFC Frames: 1764273
    Rx Data Frames Dropped: 0
    CoS  Status  Rx Frames   Tx Frames
    ---  ------  ----------  ----------
      0  off              0           0
      1  off              0           0
      2  off              0           0
      3  on               0      882032
      4  on               0      882241
      5  off              0           0
      6  off              0           0
      7  off              0           0
```

The status of the traffic-class and the configured parameters under it on the hw-module priority-flow-control profile can be checked using the following command.

```
RP/0/RP1/CPU0:NCS5504#show controllers npu priority-flow-control loc 0/0/CPU0
Mon Oct 12 14:35:17.531 UTC

Location:    0/0/CPU0
PFC:          Enabled
TC    Pause-threshold    Resume-Threshold     Headroom
-------------------------------------------------------
3       403200 bytes        40320 bytes        441600 bytes
4       403200 bytes        40320 bytes        441600 bytes
```

# Priority Flow Control Watchdog Overview

PFC Watchdog is a mechanism to identify any PFC storms (queue-stuck condition) in the network, and to prevent the PFC from propagating on the network and running in a loop. You can configure a PFC watchdog interval to detect whether packets in a no-drop queue are being drained within a specified time period. When the time period is exceeded, all outgoing packets are dropped on interfaces that match the PFC queue that is not being drained.

This requires monitoring PFC receiving on each port and detecting ports seeing an unusual amount of sustained pause frames. Once detected, the watchdog module can enforce several actions on such ports, which include generating a syslog message for network management systems, shutting down the queue, and auto-restoring the queue (after the PFC storm stops).

### Related Topics

- Priority Flow Control Overview, on page 97

# Configure a Priority Flow Control Watchdog Interval

You can configure PFC Watchdog parameters (Watchdog interval, shutdown multiplier, auto-restore multiplier) at the global or interface levels. Note that:

- When global Watchdog mode is disabled or off, Watchdog is disabled on all interfaces. This condition is regardless of the interface level Watchdog mode settings.
- When global Watchdog mode is enabled or on, the interface level Watchdog mode configuration settings override the global Watchdog mode values.
- When you configure interface level Watchdog attributes such as interval, shutdown multiplier, and auto-restore multiplier, they override the global Watchdog attributes.

**Note** Configuring the PFC mode and its policies is a prerequisite for PFC Watchdog.

**Note** PFC Watchdog also monitors, detects, and generates a syslog message every 5 minutes if global pause frames or link level flow control frames are received on a PFC-enabled port.

Such frames are ignored and discarded on the PFC-enabled port.

### Configuration Example

You can configure the Watchdog at the global or at the interface level.

**Note** Watchdog is enabled by default, with system default values of:

Watchdog interval = 100 ms

Shutdown multiplier = 1

Auto-restore multiplier = 10

```
RP/0/RP1/CPU0:NCS5504# show controllers hundredGigE 0/0/0/0 priority-flow-control
watchdog-config
Mon Oct 12 14:32:47.056 UTC

Priority flow control information for interface HundredGigE0/0/0/0:


Priority flow control watchdog configuration:
(D) : Default value
U : Unconfigured
--------------------------------------------------------------------------------
       Configuration Item           Global  Interface Effective
--------------------------------------------------------------------------------
    PFC watchdog state           :        U        U  Enabled(D)
    Poll interval                :        U        U      100(D)
    Shutdown multiplier          :        U        U        1(D)
    Auto-restore multiplier      :        U        U       10(D)
RP/0/RP1/CPU0:NCS5504# show controllers hundredGigE 0/0/0/0 priority-flow-control
watchdog-stat
watchdog-state  watchdog-stats
RP/0/RP1/CPU0:NCS5504# show controllers hundredGigE 0/0/0/0 priority-flow-control
watchdog-state
Mon Oct 12 14:32:56.760 UTC

Priority flow control information for interface HundredGigE0/0/0/0:


Priority flow control watchdog state machine state:
D - Disabled
M - Monitoring
S - Waiting For Shutdown
R - Waiting to Restore
---------------------------------------------------------------
PFC Watchdog      : Enabled
Watchdog SM state :  Traffic Class
                    7 6 5 4 3 2 1 0
                    - - - D D - - -
RP/0/RP1/CPU0:NCS5504# show controllers hundredGigE 0/0/0/0 priority-flow-control
watchdog-stats
Mon Oct 12 14:33:09.321 UTC
```

```
Priority flow control information for interface HundredGigE0/0/0/0:


Priority flow control watchdog statistics:
SAR: Auto restore and shutdown
-----------------------------------------------------------------------------------------------
Traffic Class            :       0        1        2        3        4        5        6
      7
-----------------------------------------------------------------------------------------------
Watchdog Events          :       0        0        0        0        0        0        0
      0
Shutdown Events          :       0        0        0        0        0        0        0
      0
Auto Restore Events      :       0        0        0        0        0        0        0
      0
SAR Events               :       0        0        0        0        0        0        0
      0
SAR Instantaneous Events :       0        0        0        0        0        0        0
      0
Total Dropped Packets    :       0        0        0        0        0        0        0
      0
Dropped Packets          :       0        0        0        0        0        0        0
      0
RP/0/RP1/CPU0:NCS5504#

RP/0/RP1/CPU0:NCS5504#show controllers npu priority-flow-control loc 0/0/CPU0
Mon Oct 12 14:35:17.531 UTC

Location:    0/0/CPU0
PFC:          Enabled
TC    Pause-threshold    Resume-Threshold     Headroom
-------------------------------------------------------
3       403200 bytes        40320 bytes        441600 bytes
4       403200 bytes        40320 bytes        441600 bytes
RP/0/RP1/CPU0:NCS5504#

RP/0/RP1/CPU0:NCS5504#clear controller hundredGigE 0/0/0/0 priority-flow-control
watchdog-stats ?
  traffic-class  Traffic class to be cleared
  <cr>
RP/0/RP1/CPU0:NCS5504#clear controller hundredGigE 0/0/0/0 priority-flow-control
watchdog-stats
Mon Oct 12 14:36:12.407 UTC
RP/0/RP1/CPU0:NCS5504#clear controller hundredGigE 0/0/0/0 priority-flow-control
watchdog-stats traffic-class ?
  <0-7>  Traffic class
RP/0/RP1/CPU0:NCS5504#clear controller hundredGigE 0/0/0/0 priority-flow-control
watchdog-stats traffic-class 3
NOTE: Use the clear commands to clear the statistics displayed by the show commands.
```

To restore a queue manually that has been shut down by the PFC watchdog action, run the command **set controller <> priority-flow-control recover traffic-class [0-7]**. Completion of this operation resets all internal watchdog state machines and the queue is back to a monitoring state.

```
RP/0/RP1/CPU0:NCS5504#show controllers hundredGigE 0/1/0/19 priority-flow-control
watchdog-state

Priority flow control information for interface HundredGigE0/1/0/19:


Priority flow control watchdog state machine state:
D - Disabled
M - Monitoring
```

```
S - Waiting For Shutdown
R - Waiting to Restore
---------------------------------------------------------------
PFC Watchdog : Enabled
Watchdog SM state : Traffic Class
7 6 5 4 3 2 1 0
- - - M R - - -
RP/0/RP1/CPU0:NCS5504#

RP/0/RP1/CPU0:NCS5504#set controller hundredGigE 0/1/0/19 priority-flow-control recover
traffic-class 3

RP/0/RP1/CPU0:NCS5504#show controllers hundredGigE 0/1/0/19 priority-flow-control
watchdog-state

Priority flow control information for interface HundredGigE0/1/0/19:


Priority flow control watchdog state machine state:
D - Disabled
M - Monitoring
S - Waiting For Shutdown
R - Waiting to Restore
---------------------------------------------------------------
PFC Watchdog : Enabled
Watchdog SM state : Traffic Class
7 6 5 4 3 2 1 0
- - - M M - - -
```

### Related Topics

- Priority Flow Control Overview, on page 97

# Monitoring and Logging Packet Drops on Lossless PFC-Enabled Queues

***Table 18: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Monitoring and Logging Packet Drops on Lossless PFC-Enabled Queues | Release 7.3.2 | In case of packet drops on lossless PFC-enabled queues that also have the PFC Watchdog feature enabled, this functionality generates syslogs in 10-minute intervals.<br><br>Such timely alerts help you troubleshoot quickly, isolate issues, and reroute traffic if necessary, with minimal impact on end-user services. |

# A Brief Background

Lossless PFC-enabled queues avoid queue tail drops during temporary congestion by sending PFC pause frames back to the previous network element to transmit lesser packets. Depending on the duration of the congestion, the previous network element may send PFC pause frames further upstream, hence triggering subsequent network elements to pause all the way up to the sender. Suppose that the congestion lasts longer or is persistent. In that case, the buffers on some of these network elements may still overflow, causing some packet drops eventually. PFC watchdog isolates such cases of persistent congestion and excessive PFC pause frames. PFC watchdog (enabled by default on all PFC enabled queues) monitors neighboring network elements that send excessive PFC pause frames and shuts down such queues until the PFC storm subsides. This action isolates the issue to just those queues, stops new PFC pause frame generation, and frees up the device buffers for other traffic flows, thus preventing the PFC storm from propagating throughout the network and affecting all other traffic flows.

From IOS XR Release 7.3.2, the PFC watchdog also monitors and periodically reports packet drops that occur due to excessive pausing, congestion, or PFC watchdog-triggered queue shutdown. While the **show controller** commands continue to provide an on-demand display of the total number of packets dropped for a given PFC enabled queue, the new logging mechanism provides timely alerts for proactive user intervention when drops occur on lossless PFC-enabled queues.

# Highlights of Logging and Monitoring Packet Drops

From Release 7.3.2 onwards, packet drops on lossless PFC-enabled queues trigger syslogs that record the drop statistics, alerting you about the event. These syslogs enable monitoring of such queues and provide timely updates and alerts that help you take proactive action. You can troubleshoot quickly, isolate issues, and reroute traffic if necessary, with minimal impact on end-user services.

Here are some important highlights of the monitoring and logging of packet drops:

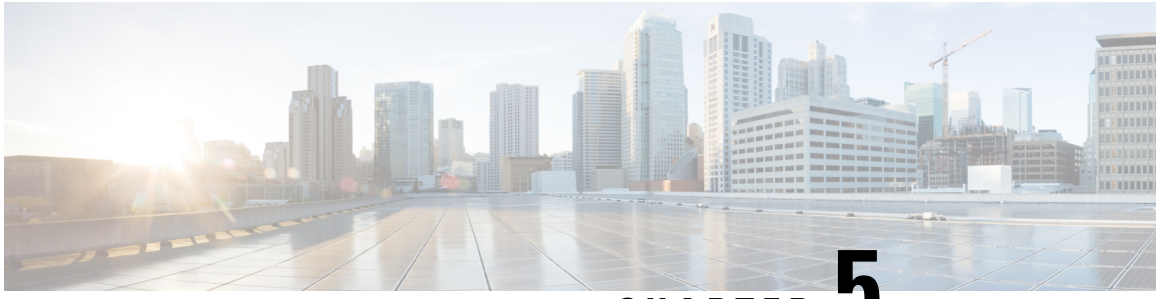
- *Only PFC-enabled queues* that also have the PFC Watchdog enabled are monitored.
- The syslog generation is event-based, where the event trigger is a packet drop on PFC Watchdog-monitored lossless queue between two consecutive measurement polls at a 10-minute interval. If the drop counters increment between such polls within a 10-minute interval, then this monitoring and logging functionality generates a syslog.
- Syslogs are generated for the following monitored parameters:
  - **Rx Data Frames Dropped** under **show controllers priority-flow-control statistics**. This counter accounts for virtual output queue (VOQ) tail drop statistics from all network processors across all ingress line cards in the system for a given egress port. VOQ tail drops occur when the PFC-enabled lossless queues build up beyond the user-configured headroom.
  - **Dropped Packets** under **show controllers priority-flow-control watchdog statistics**. This parameter accounts for all queue drops on the PFC-enabled lossless queues in the network processor on the egress line card.
- The **show controllers** command continues to display the total number of packets dropped.
- A typical syslog output looks like this:

```
%PLATFORM-PFC_WATCHDOG-5-PACKETDROP : PFC watchdog detected 150 packet drops on lossless
 priority 3 of interface HundredGigE0_7_0_30. Total drops 5678 packets.
```

# Restrictions for Monitoring and Logging Packet Drops

The following restrictions apply to the monitoring and logging of packet drops for lossless PFC-enabled queues:

- The 10-minute polling interval is fixed, and you can't modify it.
- The monitoring and logging functionality is enabled by default on all PFC Watchdog-enabled queues, and you can't disable it.



CHAPTER 5

# Configuring Modular QoS Congestion Management

This chapter covers the following topics:


- Congestion Management Overview, on page 111
- Ingress Traffic Management Model, on page 112
- Class-based Weighted Fair Queueing, on page 114
- Low-Latency Queuing with Strict Priority Queuing, on page 116
- Overhead Accounting, on page 118
- Traffic Shaping, on page 121
- 4K Pseudowire on Bundle with QoS Enhancement , on page 124
- Traffic Policing, on page 127
- References for Modular QoS Congestion Management, on page 142


## Congestion Management Overview

Congestion management features allow you to control congestion by determining the order in which a traffic flow (or packets) is sent out an interface based on priorities assigned to packets. Congestion management entails the creation of queues, assignment of packets to those queues based on the classification of the packet, and scheduling of the packets in a queue for transmission.

**Note** From Cisco IOS XR Release 7.3.1 onwards, systems with Cisco NC57 line cards running in compatibility mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

Starting with Cisco IOS XR Release 7.4.1 systems with Cisco NC57 line cards running in native mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

*Table 19: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Set Peak Burst Size for Egress Shaping | Release 7.4.1 | This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.<br><br>With this feature you can configure the burst size along with the existing egress shaper bandwidth, using the "**shape average** *shaper bandwidth burst size burst unit*" command.<br><br>This feature is used to control higher bursts of traffic being transmitted to the devices that have lower queue length configured to receive traffic. For more information, see Configure VOQ-Level Traffic Shaping, on page 121. |
| Absolute Rates for Bundle Interfaces for Traffic Shapers and Traffic Policers | Release 7.4.1 | This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes. For more information, see Configure VOQ-Level Traffic Shaping, on page 121 and Configure Traffic Policing (Two-Rate Three-Color), on page 133. |
| Ingress Policing on BVI (Bridge Group Virtual Interface) and Low-Latency Queueing (LLQ) | Release 7.4.1 | This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes. For more information, see "Restrictions for QoS on BVI" under Restrictions , on page 53 and Low-Latency Queuing with Strict Priority Queuing, on page 116 |

The types of traffic regulation mechanisms supported are:

- Class-based Weighted Fair Queueing, on page 114
- Low-Latency Queuing with Strict Priority Queuing, on page 116
- Traffic Shaping, on page 121
- Traffic Policing, on page 127

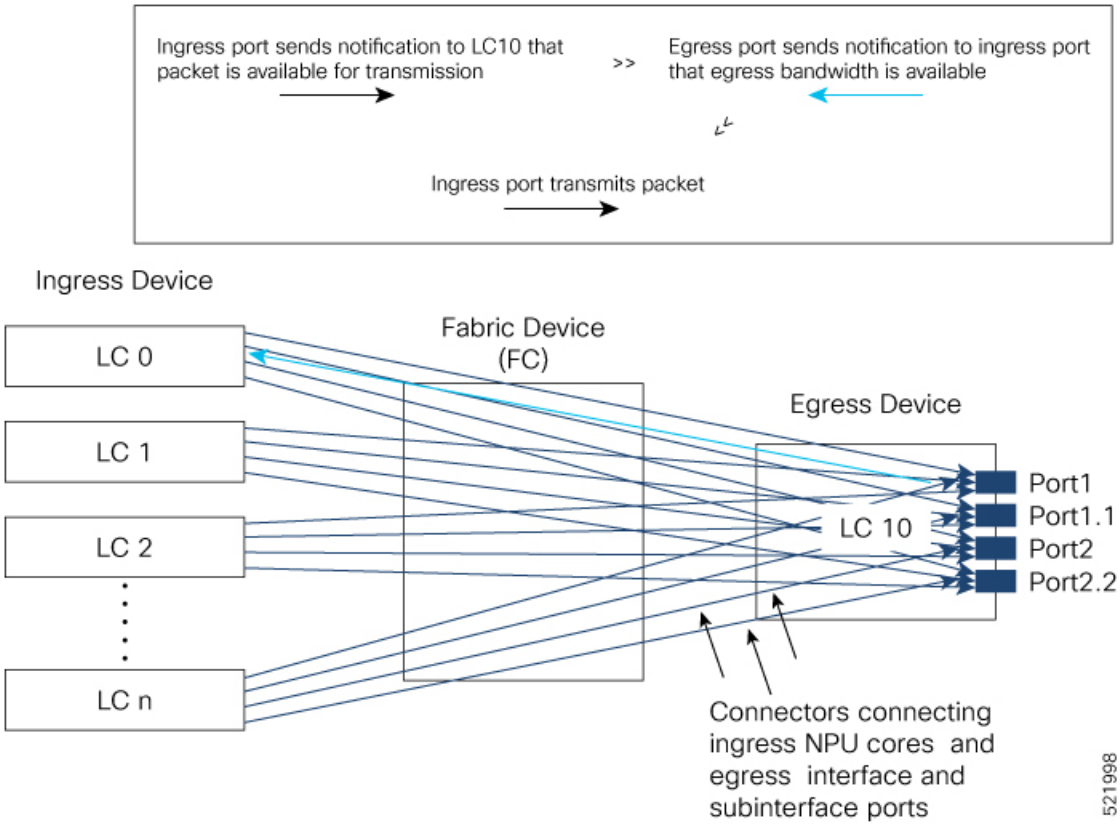
# Ingress Traffic Management Model

The ingress traffic management model relies on packet queueing on the egress interface using Virtual Output Queueing (VOQ) on the ingress. In this model, buffering takes place at ingress. Here's how the VOQ process works.

Your routers support up to eight output queues per main interface or physical port. For every egress output queue, the VOQ model earmarks buffer space on every ingress pipeline. This buffer space is in the form of

dedicated VOQs. These queues are called virtual because the queues physically exist on the ingress interface only when the line card actually has packets enqueued to it. To support the modular model of packet distribution, each network processing unit (NPU) core at the ingress needs connectors to every egress main interface and subinterface. The ingress traffic management model thus requires a mesh of connectors to connect the ingress NPU cores to the egress interfaces, as shown in **The Ingress Traffic Management Model**.

***Figure 5: The Ingress Traffic Management Model***




In the figure, every ingress interface (LC 0 through LC n) port has eight VOQs for the single egress line card LC 10.

Here's how packet transmission takes place:

1. When a packet arrives at a ingress port (say on LC 0), the forwarding lookup on the ingress line card points to the egress interface. Based on the egress interface (say it is on LC10), the packet is enqueued to the VOQ of the LC 10 egress port. The egress interface is always mapped to a physical port.

2. Credits are requested from the egress port based on the scheduling mechanism. Once credits are available, the LC 10 will receive the packets (based on the packet marking and distribution model) and send grants to the ingress ports via the connectors. (The figure shows a separate line for the grant for the sake of visual representation. In reality, the same connector is used for requests, grants, and transmission between an NPU core at the ingress and the egress port on LC 10.)

3. The ingress ports respond to this permission by transmitting the packets via FC to the LC 10 ports. (The time it takes for the ingress ports to request for egress port access, the egress port to grant access, and the packet to travel across FC is the round-trip time.)

The VOQ model thus operates on the principle of storing excess packets in buffers at ingress until bandwidth becomes available. Based on the congestion that builds up and the configured threshold values, packets begin to drop at the ingress itself, instead of having to travel all the way to the egress interface and then getting dropped.

**Hardware Limitation:**

In a scale scenario where 1000+ VoQs (created using egress QoS policies) store packets due to active traffic flows and may consume all the available on-chip buffer (OCB), unexpected traffic drops will be seen even though the traffic rate at the VoQ level is less than that of the VoQ shaper.

# Class-based Weighted Fair Queueing

Class-based Weighted Fair Queueing (CBWFQ) allows definition of traffic classes based on customer match criteria. With CBWFQ you can define traffic classes and assign guaranteed amount of minimum bandwidth to them. CBWFQ also allows for a strict priority queue for delay-sensitive traffic.

## Bandwidth Remaining

The CBWFQ  algorithm derives the weight for each class from the bandwidth remaining value allocated to the class. The **bandwidth remaining** option specifies a weight for the class to the CBWFQ . After the priority-queue is serviced, the leftover bandwidth is distributed as per bandwidth remaining ratio (BWRR) or percentage. If you do not configure this command for any class, the default value of the BWRR is considered as 1 (one). In the case of **bandwidth remaining percent**, the remaining bandwidth is equally distributed among other classes, to make it 100 percentage (100%).

### Restrictions

- The **bandwidth remaining** command is supported only for egress policies.

## Configure Minimum Bandwidth and Bandwidth Remaining

**Guidelines**

- The **priority** command cannot be configured along with **bandwidth**, **bandwidth remaining** commands, but can be configured with **shaping**, **queue-limit** and **random-detect** commands in the same class. Also see Restrictions for Configuring H-QoS, on page 152.

### Configuration Example

You have to accomplish the following to complete the minimum bandwidth and bandwidth remaining configuration:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. Allocating the minimum bandwidth and leftover bandwidth for the class
4. Attaching the policy-map to an output interface

```
Router# configure
Router(config)# policy-map test-bw-bw-rem
Router(config-pmap)# class qos-6
Router(config-pmap-c)# bandwidth percent 60
Router(config-pmap-c)# bandwidth remaining percent 60
Router(config-pmap-c# exit
Router(config-pmap)# exit
Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy output test-bw-bw-rem
Router(config-if)# commit
```

### Running Configuration

```
policy-map test-bw-bw-rem
 class qos-6
  bandwidth percent 60
  bandwidth remaining percent 60
 !
 class qos-5
  bandwidth percent 20
  bandwidth remaining percent 40
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE0/6/0/18
 service-policy input 100g-s1-1
 service-policy output test-bw-bw-rem
!
```

### Verification

```
Router# show qos interface HundredGigE 0/6/0/18 output

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220  -- output policy
NPU Id:                        3
Total number of classes:       3
Interface Bandwidth:           100000000 kbps
VOQ Base:                      11176
VOQ Stats Handle:              0x88550ea0
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   qos-6
Egressq Queue ID                         =   11182 (LP queue)
Queue Max. BW.                           =   100824615 kbps (default)
Queue Min. BW.                           =   60494769 kbps (60 %)
Inverse Weight / Weight                  =   2 (60%)
Guaranteed service rate                  =   71881188 kbps
TailDrop Threshold                       =   90177536 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                             =   qos-5
Egressq Queue ID                         =   11181 (LP queue)
Queue Max. BW.                           =   100824615 kbps (default)
Queue Min. BW.                           =   20164923 kbps (20 %)
```

```
Inverse Weight / Weight                  =   3 (40%)
Guaranteed service rate                  =   27920792 kbps
TailDrop Threshold                       =   35127296 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                             =   class-default
Egressq Queue ID                         =   11176 (Default LP queue)
Queue Max. BW.                           =   101803495 kbps (default)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   120 (BWR not configured)
Guaranteed service rate                  =   198019 kbps
TailDrop Threshold                       =   247808 bytes / 10 ms (default)
WRED not configured for this class
```

### Related Topics

- Bandwidth Remaining, on page 114

### Associated Commands

- bandwidth
- bandwidth remaining

# Low-Latency Queuing with Strict Priority Queuing

The Low-Latency Queuing (LLQ) feature brings strict priority queuing (PQ) to the CBWFQ scheduling mechanism. Priority queuing (PQ) in strict priority mode ensures that one type of traffic is sent, possibly at the expense of all others. For PQ, a low-priority queue can be detrimentally affected, and, in the worst case, never allowed to send its packets if a limited amount of bandwidth is available or the transmission rate of critical traffic is high.

## Configuring Low Latency Queuing with Strict Priority queuing

Configuring low latency queuing (LLQ) with strict priority queuing (PQ) allows delay-sensitive data such as voice to be de-queued and sent before the packets in other queues are de-queued.

**Guidelines**

- Only priority level 1 to 7 is supported, with 1 being the highest priority and 7 being the lowest. However, the default CoSQ 0 has the lowest priority among all.
- Priority level 1 to 7 is supported for non-H-QoS profiles, with 1 being the highest priority and 7 being the lowest. For H-QoS profiles, priority level 1 to 4 is supported. For all profiles, however, the class default is CoSQ 0 and has the lowest priority among all.
- Egress policing is not supported. Hence, in the case of strict priority queuing, there are chances that the other queues do not get serviced.
- You can configure **shape average** and **queue-limit** commands along with **priority**.
- A Priority Queue (PQ) can oversubscribe bandwidth when other queues do not utilize the entire port bandwidth. However, oversubscription of traffic is supported only with a single priority level.

### Configuration Example

You have to accomplish the following to complete the LLQ with strict priority queuing:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed.
3. Specifying priority to the traffic class
4. (Optional) Shaping the traffic to a specific bit rate
5. Attaching the policy-map to an output interface

```
Router# configure
Router(config)#class-map qos-1
Router(config-cmap)#match traffic-class 1
Router(config-cmap)#commit

Router(config)#class-map qos-2
Router(config-cmap)#match traffic-class 2
Router(config-cmap)#commit

Router(config)# policy-map test-priority-1
Router(config-pmap)# class qos1
Router(config-pmap-c)# priority level 7
Router(config-pmap-c)# shape average percent 2
Router(config-pmap-c)# class qos-2
Router(config-pmap-c)# priority level 6
Router(config-pmap-c)# shape average percent 1
Router(config-pmap-c)# commit
Router(config-pmap-c# exit
Router(config-pmap)# exit

Router(config)# interface HundredGigE 0/0/0/20
Router(config-if)# service-policy output test-priority-1
Router(config-if)# commit
```

### Running Configuration

```
policy-map test-priority-1
 class qos-1
  priority level 7
  shape average percent 2
 !
 class qos-2
  priority level 6
  shape average percent 1
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE0/0/0/20
 service-policy output test-priority-1
!
```

### Verification

```
Router#  show qos int hundredGigE 0/0/0/20 output

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/20 ifh 0x38  -- output policy
NPU Id:                        0
Total number of classes:       3
Interface Bandwidth:           100000000 kbps
Policy Name:                   test-priority-1
VOQ Base:                      1184
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class (HP7)                       =   qos-1
Egressq Queue ID                         =   1185 (HP7 queue)
Queue Max. BW.                           =   2000000 kbps (2 %)
Guaranteed service rate                  =   2000000 kbps
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   2499840 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class (HP6)                       =   qos-2
Egressq Queue ID                         =   1186 (HP6 queue)
Queue Max. BW.                           =   1000000 kbps (1 %)
Guaranteed service rate                  =   1000000 kbps
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   1249792 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                             =   class-default
Egressq Queue ID                         =   1184 (Default LP queue)
Queue Max. BW.                           =   no max (default)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Guaranteed service rate                  =   97000000 kbps
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   121249792 bytes / 10 ms (default)
WRED not configured for this class
```

### Related Topics

- Congestion Management Overview, on page 111
- Configure VOQ-Level Traffic Shaping, on page 121
- Bandwidth Remaining, on page 114

### Associated Commands

- priority

# Overhead Accounting

Traffic shapers and policers use packet traffic descriptors to ensure adherence to the service level agreement in QoS. However, when traffic flows from one hop to another in a network, headers added or removed at interim hops affect the packet bytes being accounted for by QoS at each hop. When your end-user network

measures the packet bytes to ensure they receive the payload as agreed, these additional header bytes cause a discrepancy.

QoS overhead accounting provides the flexibility to operators to decide which header bytes can be excluded by the traffic shaper and policer and which can be included, depending on the end user's requirements and device capabilities, to meet the committed payload in units of bytes.

For example, if the QoS commitment includes the additional header bytes, the overhead accounting feature allows your router to account for this overhead and reduces the traffic policing and shaping rates accordingly. This is also called a **positive accounting overhead**.

If however, the committed rate doesn't include the additional bytes, overhead accounting allows your router to adjust the core stream traffic such that the traffic policing and shaping rates are increased. This is also called a **negative accounting overhead**.

To summarize, QoS overhead accounting enables the router to account for packet overhead when shaping and policing traffic to a specific rate. This accounting ensures that the router runs QoS features on the actual bandwidth that the subscriber traffic consumes.

Any interface that supports QoS policies supports overhead accounting.

**Note** You can enable user overhead accounting using the optional configuration of **accounting user-defined** <***overhead size in bytes***> while attaching the service policy on the egress interface.

### Guidelines and Restrictions

- Overhead accounting for ingress shaping is not supported.

The following restrictions apply for routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.

- More than one compensation value can be programmed, provided you configure egress policy maps on different egress ports.
- You must configure a unique compensation value for a main interface and all sub-interfaces belonging to that main interface. You can't program different compensation values on different sub-interfaces sharing a common main interface.
- You can configure different compensation values on different sub-interfaces if they belong to other main interfaces.
- Compensation value programmed on egress queues (but not on VoQs) will remain active until the last egress policy map (with header compensation) is removed from main or sub-interfaces. This may impact traffic flow on main and sub-interfaces even though no compensation is set for them.

The following restrictions apply for routers that have line cards other than Cisco NC57 line cards.

- You can't program more than one compensation value per NPU or router, even if they're on different egress ports.
- You can configure the same egress compensation for different egress ports.
- NPUs can have different compensation values configured on different line cards in a modular system.

- Compensation value programmed on egress queues (but not on VoQs) will remain active until the last egress policy map (with header compensation) is removed from main or sub-interfaces. This may impact traffic flow on main and sub-interfaces even though no compensation is set for them.

### Configuring for Overhead Accounting

To configure overhead accounting, you must:

1. Create a policy map and configure QoS actions for that map.
2. Configure overhead accounting and attach the map to an interface.

```
/* create QoS policy */
Router#configure terminal
Router(config)#policy-map policer
Router(config-pmap)#class class-default
Router(config-pmap-c)#police rate percent 10
Router(config-pmap-c-police)#commit

/* configure account overhead value while attaching the QoS policy to interface */
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#service-policy input policer account user-defined 12
Router(config-if)#commit
Router(config-if)#root
Router(config)#end
```

#### Running Configuration

```
Router#sh run int hundredGigE 0/0/0/2
interface HundredGigE0/0/0/2
service-policy input policer account user-defined 12
!
```

The following example shows how to **configure a negative overhead accounting value**:

```
Router#conf
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#service-policy input policer account user-defined -12
Router(config-if)#commit
```

To **modify an overhead accounting value,** you must:

1. Remove the existing QoS policy and re-add it.
2. Configure the new overhead accounting value.

```
Router#conf
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#no service-policy input policer
Router(config-if)#service-policy input policer account user-defined -20
Router(config-if)#commit
Router#sh run int hundredGigE 0/0/0/2
interface HundredGigE0/0/0/2
service-policy input policer account user-defined -20
!
```

#### Positive Accounting Use Case

If QoS commitment includes Preamble, Frame Delimiter & Interframe Gap and has the following configuration:

```
service-policy input <foo> account user-defined +20
```

For QoS purposes, your router treats this packet as a packet of size = Actual Packet size + 20. Hence, the effective policing and shaping is *reduced* to match the downstream interface.

**Negative Accounting Use Case**

If QoS commitment to your router does not include VLAN header information, and has the following configuration:

```
service-policy input <foo> account user-defined -4
```

For QoS purposes, your router treats this packet as a packet of size = Actual Packet size – 4. Hence, the effective policing and shaping is *increased* to match the downstream interface.

### Associated Commands

service-policy (overhead accounting)

# Traffic Shaping

Traffic shaping allows you to control the traffic flow exiting an interface to match its transmission to the speed of the remote target interface and ensure that the traffic conforms to policies contracted for it. Traffic adhering to a particular profile can be shaped to meet downstream requirements, hence eliminating bottlenecks in topologies with data-rate mismatches.

**Note** If you apply a policy map that has configured traffic policing and traffic shaping on the basis of a percentage of bandwidth available on the interface and you change the speed of the interface, you must delete that policy map and reattach it to the interface. Else, QoS programming for the earlier speed remains in effect and does not change with change in the port speed.

You can configure traffic shaping at the VOQ level (see Ingress Traffic Management Model, on page 112Ingress Traffic Management Model to know how the VOQ model works), meaning you can apply shaping policies individually to each VOQ or output port. You can thus achieve fine-grained control over traffic shaping, enabling differentiated treatment of various types of traffic or destinations.

## Configure VOQ-Level Traffic Shaping

The traffic shaping performed on outgoing interfaces is done at the Layer 1 level and includes the Layer 1 header in the rate calculation.

### Guidelines

- You can configure **shape average** command along with **priority** command.
- From Cisco IOS XR Release 6.6.25 onwards, a shaper on a bundle interface also allows absolute rates apart from the already supported units of percentage, per-thousand and per-million.

  From Cisco IOS XR Release 7.4.1 onwards, this is also supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.

### Configuration Example

You have to accomplish the following to complete the traffic shaping configuration:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. Shaping the traffic to a specific bit rate and set peak burst size
4. Attaching the policy-map to an output interface

```
Router# configure
Router(config)#class-map c5
Router(config-cmap)#match traffic-class 5
Router(config-cmap)#commit

Router(config)# policy-map egress_policy1
Router(config-pmap)# class c5
Router(config-pmap-c)# shape average percent 50 1000
Router(config-pmap-c# exit
Router(config-pmap)# exit
Router(config)# interface HundredGigE 0/1/0/0
Router(config-if)# service-policy output egress_policy1
Router(config-if)# commit
```

### Running Configuration

```
class-map c5
 match traffic-class 5
commit

policy-map egress_policy1
 class c5
  shape average percent 50 1000
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE0/6/0/18
 service-policy output egress_policy1
!
```

### Verification

```
Router# show qos interface tenGigE 0/0/0/0 output

Wed Jul 10 14:18:37.783 UTC
NOTE:- Configured values are displayed within parentheses
Interface TenGigE0/0/0/0 ifh 0x120  -- output policy
NPU Id:                        0
Total number of classes:       1
Interface Bandwidth:           10000000 kbps
Policy Name:                   test
VOQ Base:                      1024
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
```

```
-------------------------------------------------------------------------------
Level1 Class                              =   class-default
Egressq Queue ID                          =   1024 (Default LP queue)
Queue Max. BW.                            =   5031499 kbps (50 %)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   5000000 kbps
Peak burst                                =   2240 bytes (1000 bytes)
TailDrop Threshold                        =   6258688 bytes / 10 ms (default)
```

### Related Topics

- Congestion Management Overview, on page 111

### Associated Commands

- shape average

# Port Shaping

Port shaping, or port-based traffic shaping, is a technique used in QoS to control and manage network traffic on specific ports. Port shaping allows you to prioritize or limit the bandwidth on specific ports to ensure that critical applications receive sufficient network resources while non-critical traffic is restricted.

By implementing port shaping, you can define rules and policies to control the bandwidth allocated to specific ports.

Port shaping can help prevent network congestion, optimize network performance, and ensure that bandwidth is allocated efficiently to meet the requirements of different applications or services.

# Key Differences Between VOQ-Level and Port-Level Shapers

*Table 20: Differences Between VOQ-level and Port-level Shapers*

| Parameter | VOQ-Level Shaper | Port-Level Shaper |
|---|---|---|
| Granularity | Operates at the individual output queue level within a network device. It shapes the traffic on a per-queue basis. | Operates at a network device's interface or port level. It shapes the overall traffic on the port, affecting all the traffic passing through that interface. |
| Operational | Helps manage bandwidth usage for different traffic flows or classes within the device. | Helps manage the aggregate bandwidth usage for the entire port. |
| Traffic control | Allows for granular control and shaping of traffic at the queue level. | Provides high-level control over the shaping of traffic, as it shapes all the traffic passing through the port without differentiating between individual flows or classes. |

| Parameter | VOQ-Level Shaper | Port-Level Shaper |
|---|---|---|
| When to use | Useful when you need to manage and shape traffic per-flow or per-class basis, providing more precise control. | Helpful when you want to manage the overall bandwidth utilization of a port without the need for fine-grained control or differentiation between specific traffic types. |

## Burst Size for VOQ-Level Shaper

Virtual Output Queue (VOQ)-level shaper burst refers to the burst size or capacity set in a VOQ-level shaper.

A VOQ level shaper within a switch or router regulates the rate at which packets are transmitted from the VOQs to the output ports in a given time interval.

The burst size or capacity in a VOQ-level shaper determines the maximum number of packets or bytes transmitted from a VOQ during a traffic burst. It defines the burst size allowed before the shaper starts limiting the traffic to conform to the specified rate.

The burst setting is thus essential in VOQ-level shaping as it affects the burstiness or the ability to handle traffic bursts. A larger burst size allows larger traffic bursts to be transmitted before the shaping mechanism restricts the flow to conform to the desired rate. Similarly, a smaller burst size limits the burstiness of the traffic and ensures a smoother and more controlled transmission.

### Configure Burst Size for VOQ-Level Shaper

See Configure VOQ-Level Traffic Shaping, on page 121 for details.

# 4K Pseudowire on Bundle with QoS Enhancement

*Table 21: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| 4K Pseudowire on Bundle with QoS Enhancement | Release 7.3.1 | With this feature you can configure a desired traffic policy, to which your network complies, by using the bandwidth management technique of two-level traffic shaping. You can also increase the Link Aggregation Group (LAG) sub-interface scale or pseudowires up to 4K.<br>This increased scale value enables you to increase the number of devices connected to your router, resulting in benefits such as increased bandwidth and cost-effective operations. |

Your router supports two modes of egress traffic shaping:

- The **default non-Hierarchical QoS (H-QoS) mode**, where the egress traffic shaping is configured only on main interfaces and no hierarchical policies are supported. In other words, egress traffic shaping on subinterfaces isn't supported in this mode.
- The **H-QoS mode**, where egress traffic shaping is also supported on subinterfaces. This mode also supports hierarchical policies on the main and subinterfaces.

Prior to Cisco IOS XR Release 7.3.1, the H-QoS mode restricted the scale of subinterfaces you could configure. For example, the maximum Link Aggregation Group (LAG) subinterface scale or pseudowires is 1K in the H-QoS mode.

The enhancement described in this section, available from Cisco IOS XR Release 7.3.1 is applicable for the default non-H-QoS mode for egress QoS. It involves configuring two-level traffic shaper policy on the main interface, while enabling you to increase the LAG subinterface scale or pseudowires to as much as 4K. What this means is that you can also scale up the number of access devices you want to connect through LAG to your router, thus enabling:

- Increased reliability and availability
- Better use of physical resources
- Increased bandwidth
- Cost-effective operations

# Restrictions and Guidelines

The following restrictions and guidelines apply while configuring two-level traffic shaper policy on the main interface in (default) non-H-QoS mode:

- The hierarchical egress policy support is only for main interfaces.
- Subinterface behavior remains the same in non-H-QoS mode. No egress QoS support is available for subinterfaces.
- This enhancement is applicable only for egress QoS, and there are no changes in ingress QoS behavior.
- There's no change to the current non-H-QoS flat policy behavior.
- The minimum shaper rate varies between different ASICs.

# Configure Two-Level Shaper Policy on Main Interface

To configure two-level shaper policy on main interface, you must:

1. Enter global configuration mode and create a two-level policy map.
2. Attach this policy map to the main interface.

```
/* Enter the global configuration mode and create the two-level policy map */
Router#configure
Router(config)# policy-map  two-level-pm
Router(config-pmap)#class class-default
Router(config-pmap-c)#shape average percent 20
```

```
Router(config-pmap-c)#service-policy child
Router(config-pmap-c)#root
Router(config)#policy-map child
Router(config-pmap)#class class-default
Router (config-pmap-c)#shape average percent 5
Router(config-pmap-c)#commit

/* Apply policy-map under interface */
Router(config)#interface hundredGigE 0/0/0/3
Router(config-intf)#service-policy output policy
Router(config-intf)#commit
```

### Running Configuration

```
Router#show running-config policy-map two-level-pm
policy-map two-level-pm
 class class-default
  service-policy child
  shape average percent 20
 !
 end-policy-map
!

Router#show running-config interface hundredGigE 0/0/0/3
interface HundredGigE0/0/0/3
service-policy output two-level-pm
!
```

### Verification

Verify that the maximum bandwidth (shaping rate) for the parent policy (Level 1) is greater than the rate for the child policy at Level 2.

```
Router#show qos interface hundredGigE 0/0/0/3 output
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/3 ifh 0x68  -- output policy
NPU Id:                        0
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Policy Name:                  two-level-pm
SPI Id:                        0x0
VOQ Base:                      1048
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   class-default
Queue Max. BW.                            =   20004211 kbps (20 %)
Queue Min. BW.                            =   20000305 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)

   Level2 Class                              =   class-default
   Egressq Queue ID                          =   1048 (Default LP queue)
   Queue Max. BW.                            =   1000000 kbps (5 %)
   Queue Min. BW.                            =   0 kbps (default)
   Inverse Weight / Weight                   =   1 / (BWR not configured)
   Guaranteed service rate                   =   1000000 kbps
   Peak burst                                =   36864 bytes (default)
   TailDrop Threshold                        =   1249792 bytes / 10 ms (default)
   WRED not configured for this class
```

# Traffic Policing

Traffic policing allows you to control the maximum rate of traffic sent or received on an interface and to partition a network into multiple priority levels or class of service (CoS).Traffic policing manages the maximum rate of traffic through a token bucket algorithm. The token bucket algorithm uses user-configured values to determine the maximum rate of traffic allowed on an interface at a given moment in time. The token bucket algorithm is affected by all traffic entering or leaving the interface (depending on where the traffic policy with traffic policing is configured) and is useful in managing network bandwidth in cases where several large packets are sent in the same traffic stream. By default, the configured bandwidth value takes into account the Layer 2 encapsulation that is applied to traffic leaving the interface.

Traffic policing also provides a certain amount of bandwidth management by allowing you to set the burst size (Bc) for the committed information rate (CIR). See, Committed Bursts and Excess Bursts, on page 128.

The router supports the following traffic policing mode(s):

- Single-Rate Two-Color (SR2C) in color-blind mode. See Single-Rate Policer, on page 128.
- Single-Rate Three-Color (SR3C) in color-blind mode.
- Two-Rate Three-Color (2R3C) in color-blind mode. See Two-Rate Policer, on page 132 .

**Note** In NC57-18DD-SE,QoS enhanced stats is enabled by default.

### Restrictions

- If you apply a policy map that has configured traffic policing and traffic shaping on the basis of a percentage of bandwidth available on the interface and you change the speed of the interface, you must delete that policy map and reattach it to the interface. Else, QoS programming for the earlier speed remains in effect and does not change with change in the port speed.
- Traffic policing is supported only in ingress direction, and only color-blind mode is supported.
- The policing rate accuracy may vary up to +/-2% from the configured policer value.
- Ensure that you don’t configure a policer and match criteria for **discard-class** in the same class. Even though the configuration is allowed, the policer doesn’t work and allows all traffic without dropping packets.
- Policer marking is not supported.
- Policers are configured in the interface at the core level and “show qos int <>” value is displayed at the NPU level.

  For policers configured in a bundle interface where bundle members are from the same NPU but different cores (NPU cores), each member sends the traffic up to the core level policer configuration, but “show qos int <>” displays the NPU level policer output.
- Example:

For bundle interface with two 10GE members (same NPU, but one interface from core0, one interface from core1) 2R3C policer applied on bundle interface (1G confirm rate , 1G exceed rate – total 2G policer rate) will be shown on the “show qos int <>” output):

Interface in core0 – 500 Mbps confirm rate, 500 Mbps exceed rate

Interface in core1 – 500 Mbps confirm rate, 500 Mbps exceed rate

For traffic in one out of two interfaces, the policed rate will be 1Gbps. For traffic on two interfaces, policed rate will be 2Gbps.

# Committed Bursts and Excess Bursts

Unlike a traffic shaper, a traffic policer does not buffer excess packets and transmit them later. Instead, the policer executes a “send or do not send” policy without buffering. Policing uses normal or committed burst (bc) values and excess burst values (be) to ensure that the router reaches the configured committed information rate (CIR). Policing decides if a packet conforms or exceeds the CIR based on the burst values you configure. Burst parameters are based on a generic buffering rule for routers, which recommends that you configure buffering to be equal to the round-trip time bit-rate to accommodate the outstanding TCP windows of all connections in times of congestion. During periods of congestion, proper configuration of the excess burst parameter enables the policer to drop packets less aggressively.

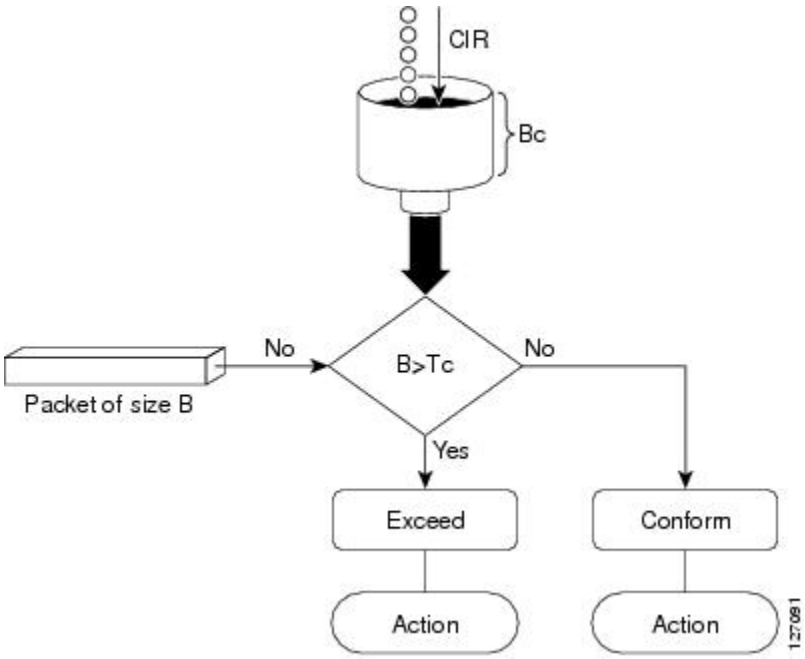
For more details, see Committed Bursts, on page 142 and Excess Bursts, on page 143.

# Single-Rate Policer

### Single-Rate Two-Color Policer

A single-rate two-color (SR2C) policer provides one token bucket with two actions for each packet: a conform action and an exceed action.

*Figure 6: Workflow of Single-Rate Two-Color Policer*




Based on the committed information rate (CIR) value, the token bucket is updated at every refresh time interval. The Tc token bucket can contain up to the Bc value, which can be a certain number of bytes or a period of time. If a packet of size B is greater than the Tc token bucket, then the packet exceeds the CIR value and a default action is performed. If a packet of size B is less than the Tc token bucket, then the packet conforms and a different default action is performed.

### Single-Rate Three-Color Policer

A single-rate three-color (SR3C) policer provides one token bucket with three actions for each packet: a conform action, an exceed action and a violate action. The packet is marked based on the CIR value and the two associated burst size - committed burst size (CBS) and excess burst size (EBS). If a packet does not exceed the CBS, it is marked as conformed packet. The packet is marked as exceeded if it exceeds CBS, but not the EBS. If it exceeds the EBS as well, it is marked as violate packet.

## Configure Traffic Policing (Single-Rate Two-Color)

Traffic policing is often configured on interfaces at the edge of a network to limit the rate of traffic entering or leaving the network. The default conform action for single-rate two color policer is to transmit the packet and the default exceed action is to drop the packet. Users cannot modify these default actions.

### Configuration Example

You have to accomplish the following to complete the traffic policing configuration:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. (Optional) Specifying the marking action
4. Specifying the policy rate for the traffic

5. Attaching the policy-map to an input interface

```
Router# configure
Router(config)# policy-map test-police-1
Router(config-pmap)# class  ipv6-6
Router(config-pmap-c)# set dscp cs2 (optional)
Router(config-pmap-c)# set qos-group 7 (optional)
Router(config-pmap-c)# police rate percent 20 burst 10000 bytes
Router(config-pmap-c-police)# exit
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy input test-police-1
Router(config-if)# commit
```

### Running Configuration

```
class-map match-any ipv6-6
 match precedence 3
 end-class-map
!

policy-map test-police-1
 class ipv6-6
  set dscp cs2
  set qos-group 7
  police rate percent 20 burst 10000 bytes
  !
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE0/6/0/18
 service-policy input test-police-1
 service-policy output test-priority-1
!
```

### Verification

```
Router# show qos interface hundredGigE 0/6/0/18 input

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220  -- input policy
NPU Id:                        3
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   ipv6-6
New dscp                                  =   16
New qos group                             =   7

Policer Bucket ID                         =   0x102a0
Policer Stats Handle                      =   0x8a8090c0
Policer committed rate                    =   19980000 kbps (20 %)
```

```
Policer conform burst                    =   9856 bytes (10000 bytes)

Level1 Class                             =   class-default

Default Policer Bucket ID                =   0x102a1
Default Policer Stats Handle             =   0x8a808e78
Policer not configured for this class
```

### Related Topics

- Traffic Policing, on page 127

### Associated Commands

- police rate

## Configure Traffic Policing (Single-Rate Three-Color)

The default conform action and exceed actions for single-rate three-color policer are to transmit the packet and the default violate action is to drop the packet. User cannot modify these default actions.

### Configuration Example

You have to accomplish the following to complete the traffic policing configuration:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. (Optional) Specifying the marking action
4. Configuring the policy rate for the traffic along with the peak-burst values
5. Attaching the policy-map to an input interface

```
Router# configure
Router(config)# policy-map test-police-1R3C
Router(config-pmap)# class ipv4-5
Router(config-pmap-c)# set qos-group 2 (optional)
Router(config-pmap-c)# police rate percent 20 burst 100000 bytes peak-burst 190000 bytes
Router(config-pmap-c-police)# exit
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy input test-police-1R3C
Router(config-if)# commit
```

### Running Configuration

```
class-map match-any ipv4-5
 match precedence 3
 end-class-map
!

policy-map test-police-1R3C
```

```
 class ipv4-5
  set qos-group 7
  police rate percent 20 burst 100000 bytes peak-burst 190000 bytes
  !
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE0/6/0/18
 service-policy input test-police-1R3C
 service-policy output test-priority-1
!
```

### Verification

```
Router# show qos interface hundredGigE 0/6/0/18 input

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220  -- input policy
NPU Id:                        3
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   ipv4-5
New qos group                             =   2

Policer Bucket ID                         =   0x102a1
Policer Stats Handle                      =   0x8a8090c0
Policer committed rate                    =   19980000 kbps (20 %)
Policer conform burst                     =   99584 bytes (100000 bytes)
Policer exceed burst                      =   188672 bytes (190000 bytes)

Level1 Class                              =   class-default

Default Policer Bucket ID                 =   0x102a1
Default Policer Stats Handle              =   0x8a808e78
Policer not configured for this class
```

### Related Topics

- Traffic Policing, on page 127

### Associated Commands

- police rate

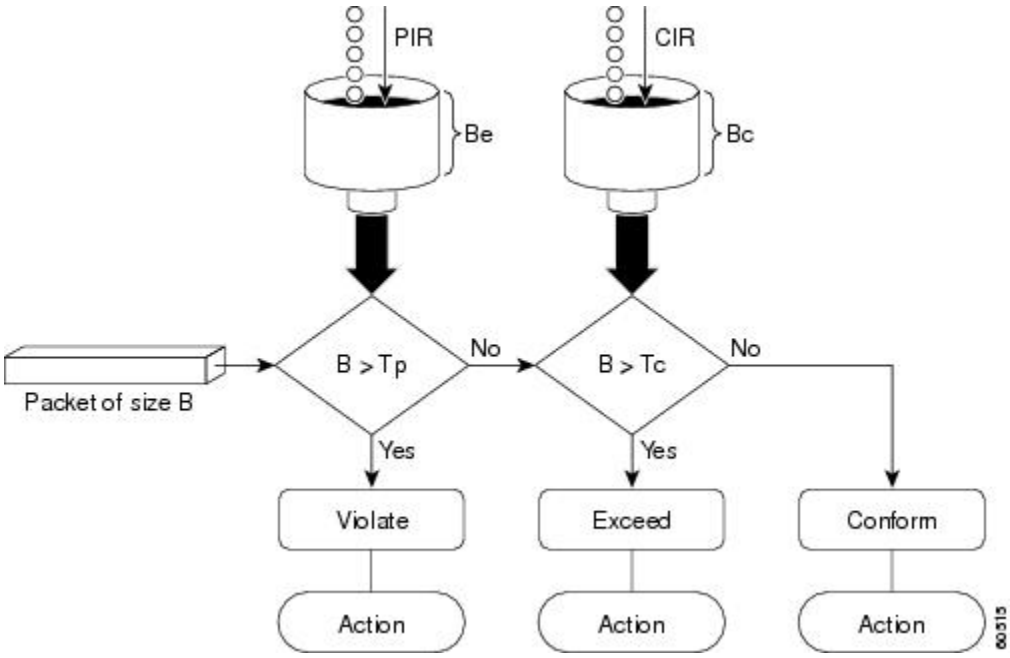
# Two-Rate Policer

The two-rate policer manages the maximum rate of traffic by using two token buckets: the committed token bucket and the peak token bucket. The dual-token bucket algorithm uses user-configured values to determine the maximum rate of traffic allowed on a queue at a given moment. In this way, the two-rate policer can meter traffic at two independent rates: the committed information rate (CIR) and the peak information rate (PIR).

The dual-token bucket algorithm provides users with three actions for each packet—a conform action, an exceed action, and an optional violate action. Traffic entering a queue with the two-rate policer configured is placed into one of these categories. The actions are pre-determined for each category. The default conform and exceed actions are to transmit the packet, and the default violate action is to drop the packet.

This figure shows how the two-rate policer marks a packet and assigns a corresponding action to the packet.

***Figure 7: Marking Packets and Assigning Actions—Two-Rate Policer***




Also, see Two-Rate Policer Details, on page 143.

The router supports Two-Rate Three-Color (2R3C) policer.

## Configure Traffic Policing (Two-Rate Three-Color)

The default conform and exceed actions for two-rate three-color (2R3C) policer are to transmit the packet and the default violate action is to drop the packet. Users cannot modify these default actions.

### Configuration Example

You have to accomplish the following to complete the two-rate three-color traffic policing configuration:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. Specifying the packet marking
4. Configuring two rate traffic policing
5. Attaching the policy-map to an input interface

```
Router# configure
Router(config)# policy-map policy1
```

```
Router(config-pmap)# class ipv4-7
Router(config-pmap-c)# set qos-group 4
Router(config-pmap-c)# police rate percent 20 burst 100000 bytes peak-rate percent 50
peak-burst 200000 bytes
Router(config-pmap-c-police)# exit
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy input policy1
Router(config-if)# commit
```

### Running Configuration

```
policy-map policy1
 class ipv4-7
  set qos-group 4
  police rate percent 20 burst 100000 bytes peak-rate percent 50 peak-burst 200000 bytes
  !
 !

interface HundredGigE 0/6/0/18
 service-policy input policy1
!
```

### Verification

```
Router# show policy-map interface HundredGigE 0/6/0/18

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220  -- input policy
NPU Id:                        3
Total number of classes:       8
Interface Bandwidth:           100000000 kbps
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   ipv4-4
- - -
- - -
Level1 Class                              =   ipv4-7
New qos group                             =   4

Policer Bucket ID                         =   0x102a3
Policer Stats Handle                      =   0x8a8089e8
Policer committed rate                    =   19980000 kbps (20 %)
Policer peak rate                         =   49860000 kbps (50 %)
Policer conform burst                     =   99584 bytes (100000 bytes)
Policer exceed burst                      =   199168 bytes (200000 bytes)

Level1 Class                              =   class-default

Policer Bucket ID                         =   0x102a7
Policer Stats Handle                      =   0x8a7c8510
Policer committed rate                    =   29880000 kbps (30 %)
Policer conform burst                     =   4194304 bytes (default)
```

### Important Notes

- From Cisco IOS XR Release 6.6.25 onwards:
  - a committed information rate of 0 kbps is supported, which is the only exception to the minimum rate of 22 kbps for both commit and excess rates.
  - a policer on a bundle interface also allows absolute rates apart from the already supported units of percent, per-thousand and per-million. From Cisco IOS XR Release 7.4.1 onwards, this is also supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.
- A policer is programmed per NPU core on a bundle interface. So, all members on a bundle interface from the same core share the policer.

### Related Topics

- Two-Rate Policer, on page 132

### Associated Commands

- police rate

# Packets-Per-Second-Based Policer

*Table 22: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Packets-Per-Second-Based Policer | Release 7.4.1 | Prior to this functionality, when configuring policers, the only available option for policer rates was bit-rate measured in units of bits per second (bps). With this release, you can configure policer rates in units of packets per second (pps) as well. pps-based policer is critical in fending off malicious attacks—when attackers target your specific resources with a vast amount of traffic that contain higher number of packets, but move at a slower packet rate. Protection from such attacks is possible because pps-based policers ensure that regardless of the packet size and rate, the policer only accepts a fixed number of packets per second.<br><br>This functionality modifies the **police rate** command. |

- **Policer rates so far**—You used the **police rate** command to configure policers based on two parameters:
  - bit-rates (default unit: bits per second or bps)
  - Burst size (default unit: bytes)
- **packets-per-second (pps)-based policer**—With this additional functionality, you can use the **police rate** command to configure policers in units of packets per second (pps). The pps configuration option is available as **police rate <*pps-value*> pps**. When you configure the pps option, ensure that you configure burst size in packets. (See **Restrictions and guidelines**.) Thus, the parameters for pps-based policer are:
  - packets per second (pps)
  - burst size (packets)
- **Why pps-based-policer**—Networks face newer types of attacks, and these days malicious operators don't necessarily employ aggressive tactics that involve overwhelming your bandwidth with large amount of traffic to cause distributed denial of service (DDoS). Now, some attackers go the 'softer' route, where they send smaller packet sizes at slower traffic rates. During such malicious network activity, a bandwidth-based policer can still aggregate up to many packets to be processed if the packet size is small. Attackers tend to use this behavior to bypass bandwidth-based policers to exploit vulnerabilities or try to hit performance limitations by increasing the packet rates.

  Packets-per-second-based policers ensure that regardless of the packet size and traffic rate, the policer only accepts a fixed number of packets per second.

  **pps-based-policer support cheat-sheet**—Here's a quick look at some key support areas and their details for pps-based policer.

| Support | Details |
|---|---|
| Classification and marking support | Same as that for bps-based-policer |
| Units | Equivalent kbps values display for QoS programming and statistics. |
| H-QoS | Support for parent and child policers |
| Bursts | Support for confirm burst (bc) and exceed burst (be) values in units of packets. The default value is in multiple of 128 bytes equivalent to 10 milliseconds. |
| Minimum pps value | For better granularity, recommended minimum value is 100 pps. |

- **Restriction and guidelines**
  - This functionality is **applicable only for ingress.**
  - When using a pps-based policer, **ensure that you configure the burst-size value in number of packets as well.** This is because a policer burst rate determines whether a specific number of packets out of contract would be subject to the next action (that is, exceed or violate).
  - **Within a QoS policy, configure the parent and child policies policers to either bps or pps.** Else, the configuration displays an error when you try attaching the policy to an interface.

- **For single-level policy maps:** under the same policy map, you can configure one class map with bps-based policer and the other class map with a pps-based policer.
- **For two-level hierarchical policy maps:**
  - The parent and child-level policy maps must use the same unit-based policer. That is, both must have either pps-based or bps-based policers.
  - If you configure the child-level policy map with pps-based policer, ensure that the parent policy-map class default has a pps-based policer.

- **Configure pps-based policer**—To configure pps-based policer, you must:
  1. Configure a class map.
  2. Create a service policy for the map and configure the pps values.
  3. Attach the service policy to an interface.

```
/*Configure a class map*/

Router(config)#class-map prec1
Router(config-cmap)#match precedence 1
Router(config-cmap)# exit
Router(config)# commit

/*Create a service policy map*/

Router(config)# policy-map policy1
Router(config-pmap)# class prec1
Router(config-pmap-c)#police rate 1000 pps burst 300 packets
Router(config-pmap-c-police)#exit
Router(config-pmap-c)#exit
Router(config-pmap)#exit
Router(config)# commit

/*Attach the service policy to an interface*/
Router#int hundredGigE 0/7/0/2
Router(config-if)service-policy input policy1
Router(config-if)#exit
Router(config)#commit
```

### Running Configuration

```
class-map match-any prec1
match precedence 1
 end-class-map
!
policy-map policy1
class prec1
  police rate 1000 pps burst 300 packets
  !
 !
 class class-default
!
 end-policy-map
!
```

### Verification

```
Router#show qos int hundredGigE 0/7/0/2 input
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/7/0/2 ifh 0xe000088  -- input policy
NPU Id:                        0
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Policy Name:                   policy1
SPI Id:                        0x0
Accounting Type:               Layer2 (Include Layer 2 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   prec1

Policer Bucket ID                         =   0x9
Policer Stats Handle                      =   0x0
Policer committed rate                    =   998 kbps (1000 packets/sec)
Policer conform burst                     =   37632 bytes (300 packets)

Level1 Class                              =   class-default

Default Policer Bucket ID                 =   0x8
Default Policer Stats Handle              =   0x0
Policer not configured for this class
```

### Associated Commands

police rate

# Per-thousand and Per-million Units

Shaper and policer rates can be configured in units of per-thousand and per-million on bundle interfaces. This provides the ability to provision shape and police rates down to 100 kbps on bundle or link aggregation (LAG) interfaces even with 100 GE bundle members.

For example, consider a 100GE interface and simple policy.

```
Interface HundredGig0/0/0/0
Service-policy output TEST
Policy-map TEST
Class C
 Shape average per-thousand 5
End-policy
```

Per thousand represents 0.1% of the link bandwidth and per million represents 0.0001% of the link bandwidth.

Which means that for a 100G link, 5 parts per thousand is 0.5% of the link bandwidth. Hence, the shape average per thousand of 5 in the above example enforces a shaper of 500 Mbps.

# Shared Policer

*Table 23: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Shared Policer | Release 7.4.1 | This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes. |

The classification of the incoming packet occurs only once. However, based on the different classification criteria, the shared policer feature allows sharing of the policer bucket amongst two or more classes in a QoS policy map . That is, the same token bucket is used for a traffic flow matching against any of the classes sharing the policer.

For example, let us say a policer of 10 Mbps is shared among two classes C1 and C2. This feature ensures that both C1 and C2 get traffic flow assigned on First Come First Serve (FCFS) basis. Also that, if C2 does not have any traffic, C1 uses all of the 10 Mbps for transmission.

This feature includes two components:

- Policer Bucket Shared
- Policer Bucket Referred

## Policer Bucket Shared

The policer bucket shared feature defines and shares a policer bucket instance among multiple classes.

Here is a sample configuration that defines and shares policer bucket instance sp1 :

```
policy-map parent
        class long-distance
          police bucket shared sp1 rate 1 mbps
```

In this configuration, a policy-map for class long-distance traffic type is created to police at 1Mbps rate and the policer bucket is shared.

## Policer Bucket Referred

The policer bucket referred feature refers a defined policer bucket instance. Shared policer is not supported across policy levels. This means for example, that parent and child policy cannot share a common bucket.

Here is a sample configuration that refers shared policer bucket instance sp1 :

```
policy-map voip-child
        class long-distance-voip
         police bucket referred sp1
```

In this configuration, a policy-map for class long-distance-voip traffic type is created and the shared policer bucket sp1 is referred.

## Shared Policer Statistics

Currently, individual class statistics are not available as a default option for shared policer. You can access statistics in the following modes.

### Aggregate Mode

In this mode the policer bucket is shared among two or more classes. However, statistics are not available for every individual class. You can view the aggregate statistics that combine the numbers for all the classes sharing the policer bucket.

### Per-Class Mode

In this mode the policer bucket is shared among two or more classes, and you can also view individual class statistics. However, when this mode is active, the Policy-Based Tunnel Selection (PBTS) mechanism is

disabled. To enable the per-class mode, you must configure the **hw-module profile qos shared-policer-per-class-stats** command.

## Restrictions and Guidelines

The following restrictions and guidelines apply while configuring the shared policer feature.

- When shared policer is enabled in per-class mode, Policy-Based Tunnel Selection (PBTS) mechanism is disabled. In other words, shared policer-per-class-mode and PBTS are mutually exclusive features.
- Shared policer is not supported across policy levels. This means, for example, that parent and child policies cannot share a common policer bucket.
- Shared policer is not supported in ingress peering mode.
- Shared policer is supported within classes of the same policy. However, cross-policy bucket sharing is not supported.
- There are no limitations on the number of classes that can share policer.
- There are no changes in policer scale numbers in the aggregate and per-class modes.
- All the existing policer types (1R2C, 1R3C and 2R3C) are supported.
- You must reload the affected line card to enable the per-class-stats mode.

## Configuring Shared Policer

To configure shared policer, you must:

1. Create a class map to be used for matching packets to the specified class.
2. Create a policy map to be used for matching packets to the specified class.
3. Specify a class name.
4. Define and share a policer bucket.
5. Specify a class name.
6. Refer a shared policer bucket.

```
RP/0/RSP0/CPU0:ios(config)#class-map match-any c1
RP/0/RSP0/CPU0:ios(config-cmap)#match precedence 1
RP/0/RSP0/CPU0:ios(config-cmap)#end-class-map
RP/0/RSP0/CPU0:ios(config)#class-map match-any c2
RP/0/RSP0/CPU0:ios(config-cmap)#match precedence 2
RP/0/RSP0/CPU0:ios(config-cmap)#end-class-map
RP/0/RSP0/CPU0:ios(config)#policy-map s-pol
RP/0/RSP0/CPU0:ios(config-pmap)#class c1
RP/0/RSP0/CPU0:ios(config-pmap-c)#police bucket shared 1 rate 10 mbps
RP/0/RSP0/CPU0:ios(config-pmap-c-police)#exit
RP/0/RSP0/CPU0:ios(config-pmap-c)#exit
RP/0/RSP0/CPU0:ios(config-pmap)#class c2
RP/0/RSP0/CPU0:ios(config-pmap-c)#police bucket referred 1
RP/0/RSP0/CPU0:ios(config-pmap-c-police)#class class-default
RP/0/RSP0/CPU0:ios(config-pmap-c)#exit
RP/0/RSP0/CPU0:ios(config-pmap)#exit
RP/0/RSP0/CPU0:ios(config)#interface HundredGigE 0/6/0/18
```

```
RP/0/RSP0/CPU0:ios(config-if)#service-policy input s-pol
RP/0/RSP0/CPU0:ios(config-if)#commit
```

### Running Configuration

```
class-map match-any c1
 match precedence 1
 end-class-map

class-map match-any c2
 match precedence 2
 end-class-map

policy-map s-pol
 class c1
  police bucket shared 1 rate 10 mbps
  !
 !
 class c2
  police bucket referred 1
  !
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE0/6/0/18
 service-policy input s-pol
!
```

### Verification

### In Aggregate Mode

```
RP/0/RP0/CPU0:ios#sh policy-map interface tenGigE 0/0/0/0 input
Fri Nov 15 12:55:56.817 UTC

TenGigE0/0/0/0 input: s-pol

Class c1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          1784530245/228419871360         8640780
    Transmitted         :             2067504/264640512            10011
    Total Dropped       :          1782462741/228155230848         8630769
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :             2067504/264640512            10011
    Policed(exceed)     :          1782462741/228155230848         8630769
    Policed(violate)    :                   0/0                    0
    Policed and dropped :          1782462741/228155230848
Class c2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
  Policing statistics                (packets/bytes)     (rate - kbps)

Policed(conform)    :                   0/0                    0
    Policed(exceed)     :                   0/0                    0
    Policed(violate)    :                   0/0                    0
    Policed and dropped :                   0/0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
```

```
    Matched             :                   0/0                     0
    Transmitted         :                   0/0                     0
    Total Dropped       :                   0/0                     0
Policy Bag Stats time: 1573822531986  [Local Time: 11/15/19 12:55:31.986]
```

### In Per-Class Mode

```
RP/0/RP0/CPU0:ios#sh policy-map interface tenGigE 0/0/0/0 input
Fri Nov 15 15:18:18.319 UTC

TenGigE0/0/0/0 input: s-pol

Class c1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          1005369276/128687267328         4320337
    Transmitted         :             1163300/148902400            5013
    Total Dropped       :          1004205976/128538364928         4315324
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :             1163300/148902400            5013
    Policed(exceed)     :          1004205976/128538364928         4315324
    Policed(violate)    :                   0/0                    0
    Policed and dropped :          1004205976/128538364928
Class c2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          1005341342/128683691776         4320335
    Transmitted         :             1166269/149282432            4997
    Total Dropped       :          1004175073/128534409344         4315338
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :             1166269/149282432            4997
    Policed(exceed)     :          1004175073/128534409344         4315338
    Policed(violate)    :                   0/0                    0
    Policed and dropped :          1004175073/128534409344
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :               49159/6292352              0
    Transmitted         :               49159/6292352              0
    Total Dropped       :                   0/0                    0
Policy Bag Stats time: 1573831087338  [Local Time: 11/15/19 15:18:07.338]
```

**Related Commands** hw-module profile qos shared-policer-per-class-stats

# References for Modular QoS Congestion Management

## Committed Bursts

The committed burst (bc) parameter of the police command implements the first, conforming (green) token bucket that the router uses to meter traffic. The bc parameter sets the size of this token bucket. Initially, the token bucket is full and the token count is equal to the committed burst size (CBS). Thereafter, the meter updates the token counts the number of times per second indicated by the committed information rate (CIR).

The following describes how the meter uses the conforming token bucket to send packets:

- If sufficient tokens are in the conforming token bucket when a packet arrives, the meter marks the packet green and decrements the conforming token count by the number of bytes of the packet.

- If there are insufficient tokens available in the conforming token bucket, the meter allows the traffic flow to borrow the tokens needed to send the packet. The meter checks the exceeding token bucket for the number of bytes of the packet. If the exceeding token bucket has a sufficient number of tokens available, the meter marks the packet.

  Green and decrements the conforming token count down to the minimum value of 0.

  Yellow, borrows the remaining tokens needed from the exceeding token bucket, and decrements the exceeding token count by the number of tokens borrowed down to the minimum value of 0.

- If an insufficient number of tokens is available, the meter marks the packet red and does not decrement either of the conforming or exceeding token counts.

**Note** When the meter marks a packet with a specific color, there must be a sufficient number of tokens of that color to accommodate the entire packet. Therefore, the volume of green packets is never smaller than the committed information rate (CIR) and committed burst size (CBS). Tokens of a given color are always used on packets of that color.

# Excess Bursts

The excess burst (be) parameter of the police command implements the second, exceeding (yellow) token bucket that the router uses to meter traffic. The exceeding token bucket is initially full and the token count is equal to the excess burst size (EBS). Thereafter, the meter updates the token counts the number of times per second indicated by the committed information rate (CIR).

The following describes how the meter uses the exceeding token bucket to send packets:

- When the first token bucket (the conforming bucket) meets the committed burst size (CBS), the meter allows the traffic flow to borrow the tokens needed from the exceeding token bucket. The meter marks the packet yellow and then decrements the exceeding token bucket by the number of bytes of the packet.
- If the exceeding token bucket does not have the required tokens to borrow, the meter marks the packet red and does not decrement the conforming or the exceeding token bucket. Instead, the meter performs the exceed-action configured in the police command (for example, the policer drops the packets).

# Two-Rate Policer Details

The committed token bucket can hold bytes up to the size of the committed burst (bc) before overflowing. This token bucket holds the tokens that determine whether a packet conforms to or exceeds the CIR as the following describes:

- A traffic stream is conforming when the average number of bytes over time does not cause the committed token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream green.
- A traffic stream is exceeding when it causes the committed token bucket to overflow into the peak token bucket. When this occurs, the token bucket algorithm marks the traffic stream yellow. The peak token bucket is filled as long as the traffic exceeds the police rate.

The peak token bucket can hold bytes up to the size of the peak burst (be) before overflowing. This token bucket holds the tokens that determine whether a packet violates the PIR. A traffic stream is violating when it causes the peak token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream red.

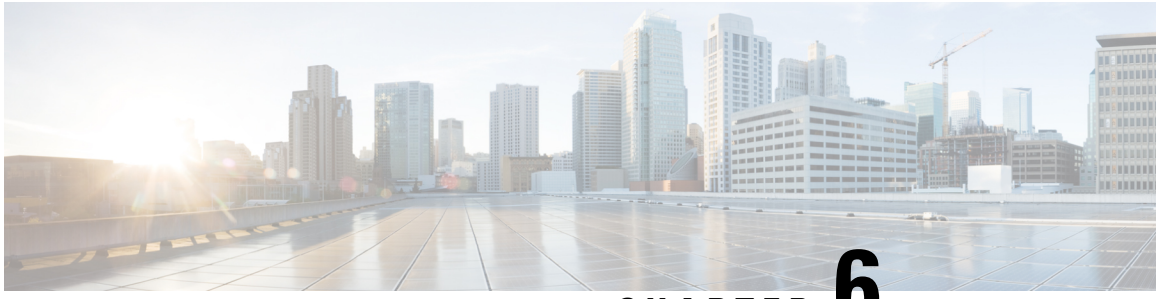
For example, if a data stream with a rate of 250 kbps arrives at the two-rate policer, and the CIR is 100 kbps and the PIR is 200 kbps, the policer marks the packet in the following way:

- 100 kbps conforms to the rate
- 100 kbps exceeds the rate
- 50 kbps violates the rate

The router updates the tokens for both the committed and peak token buckets in the following way:

- The router updates the committed token bucket at the CIR value each time a packet arrives at the interface. The committed token bucket can contain up to the committed burst (bc) value.
- The router updates the peak token bucket at the PIR value each time a packet arrives at the interface. The peak token bucket can contain up to the peak burst (be) value.
- When an arriving packet conforms to the CIR, the router takes the conform action on the packet and decrements both the committed and peak token buckets by the number of bytes of the packet.
- When an arriving packet exceeds the CIR, the router takes the exceed action on the packet, decrements the committed token bucket by the number of bytes of the packet, and decrements the peak token bucket by the number of overflow bytes of the packet.
- When an arriving packet exceeds the PIR, the router takes the violate action on the packet, but does not decrement the peak token bucket.

See Two-Rate Policer, on page 132.



CHAPTER 6

# Configuring Modular QoS on Link Bundles

This chapter covers the following topics:


- QoS on Link Bundles, on page 145


## QoS on Link Bundles

A bundle is a group of one or more ports that are aggregated together and treated as a single link. The router supports Ethernet interfaces and VLAN interfaces (bundle sub-interfaces) bundles. All QoS features currently supported on physical interfaces, are also supported on all link bundle interfaces. Applying QoS on bundle members is not supported.

**Note** From Cisco IOS XR Release 7.3.1 onwards, systems with Cisco NC57 line cards running in compatibility mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

Starting with Cisco IOS XR Release 7.4.1 systems with Cisco NC57 line cards running in native mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

### Restrictions for Link Bundles

- Only Ethernet link bundling is supported.
- A bundle interface can only contain physical interface.
- All links within a single bundle must be configured either to run 802.3ad (LACP) or Etherchannel (non-LACP). Mixed links within a single bundle are not supported.
- MAC accounting is not supported on Ethernet link bundles.
- Maximum number of links supported in each link bundle is 64.

- The maximum number of link bundles supported is 128.

# Load Balancing

Load balancing function is a forwarding mechanism to distribute traffic over multiple links based on Layer 3 routing information in the router. Per-destination load balancing isonly supported on the router, where the router is allowed to distribute packets over one of the links in the bundle. When the per-destination load balancing is enabled, all packets for a certain source-destination pair goes through the same link, though there are multiple links available. In other words, per-destination load balancing can ensure that packets for a certain source-destination pair could arrive in order.

### Layer 3 Load Balancing on Link Bundles

Layer 3 load balancing for link bundles is done on Ethernet Flow Points (EFPs) and is based on the IPv4 source and destination addresses in the packet. When Layer 3 service-specific load balancing is configured, all egress bundles are load balanced based on the IPv4 source and destination addresses. When packets do not have IPv4 addresses, default load-balancing (based on the MAC SA/DA fields in the packet header) is used.

# Configure QoS on Link Bundles

QoS is configured on link bundles in the same way that it is configured on individual interfaces.

### Guidelines

- When a QoS policy is applied on a bundle in the egress direction, it’s also applied at each member interface.
- When a QoS policy is applied on a bundle (ingress direction), it’s replicated at each NPU core.
- If a QoS policy is not applied to a bundle interface, both the ingress and egress traffic use the default queue of the per link member port.
- The shape rate that is specified in the bundle policy-map is not an aggregate for all bundle members. The shape rate applied to the bundle depends on the load balancing of the links. For example, if a policy map with a shape rate of 10 Mbps is applied to a bundle with two member links, and if the traffic is always load-balanced to the same member link, then an overall rate of 10 Mbps applies to the bundle. However, if the traffic is load-balanced evenly between the two links, the overall shape rate for the bundle becomes 20 Mbps.
- If a member is deleted from a bundle, the total bundle statistics changes because the statistics that belongs to the detached link is lost.
- The QoS policy that is applied on bundle is inherited to all its member links and the reference bandwidth that is used to calculate shaper/bandwidth is applied as per the physical member interface bandwidth, and not the bundle as a whole.

### Configuration Example

You have to accomplish the following to complete the QoS configuration on link bundles:

**Note** The policy works only if it is applied on the ingress direction. The egress is supported on COS, DEI and MPLS exp marking. So the below policy may not work when it is applied on egress.

1. Creating a class-map
2. Creating a policy-map and specifying the respective class-map
3. Specifying the action type for the traffic

   Refer Attach a Traffic Policy to an Interface, on page 13 for details on step 1, 2 and 3.
4. Creating a link bundle
5. Applying traffic policy to the link bundle

```
/* Configure Ether-Bundle and apply traffic policy */
Router(config)# interface Bundle-Ether 12000
Router(config-if)# mtu 9100
Router(config-if)# service-policy input ingress
Router(config-if)# service-policy output egress
Router(config-if)# ipv4 address 100.12.0.0 255.255.255.254
Router(config-if)# bundle maximum-active links 64
Router(config-if)# commit
```

### Running Configuration

This example shows how a traffic policy is applied on an Ethernet link bundle. The policy is applied to all interfaces that are members of the Ethernet link bundle.

```
/* Policy-map */

policy-map ingress
 class inet4-classifier-af1
  set qos-group 1
 !
 class inet4-classifier-af2
  set qos-group 2
 !
 class inet4-classifier-af3
  set qos-group 3
 !
 class inet4-classifier-af4
  set qos-group 4
 !
 class inet4-classifier-be1
  set qos-group 5
 !
 class inet4-classifier-nc1
  set qos-group 6
 !
 class class-default
 !
 end-policy-map
!

/* Ether Bundle */
```

```
interface Bundle-Ether12000
 mtu 9100
 service-policy input ingress
 service-policy output egress
 ipv4 address 100.12.0.0 255.255.255.254
 load-interval 30
 flow ipv4 monitor FMM-V4 sampler SM ingress
 flow ipv6 monitor FMM-V6 sampler SM ingress
 flow mpls monitor FMM-MPLS sampler SM ingress
 ipv4 access-group IPV4ACL_101 ingress
 ipv6 access-group IPV6ACL_101 ingress
!
```

### Verification

- Verify that the bundle status is UP.

```
router# show bundle bundle-ether 1200
Wed Dec 16 19:55:49.974 PST

Bundle-Ether12000
  Status:                                      Up
  Local links <active/standby/configured>:     35 / 0 / 35
  Local bandwidth <effective/available>:       3500000000 (3500000000) kbps
  MAC address (source):                        ea3b.745f.c4b0 (Chassis pool)
  Inter-chassis link:                          No
  Minimum active links / bandwidth:            1 / 1 kbps
  Maximum active links:                        64
  Wait while timer:                            2000 ms
  Load balancing:                              Default
  LACP:                                        Operational
    Flap suppression timer:                    Off
    Cisco extensions:                          Disabled
    Non-revertive:                             Disabled
  mLACP:                                       Not configured
  IPv4 BFD:                                    Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Hu0/4/0/0             Local            Active       0x8000, 0x0009   100000000
      Link is Active
  Hu0/4/0/1             Local            Active       0x8000, 0x000a   100000000
      Link is Active
- - -
- - -
  Hu0/4/0/35            Local            Active       0x8000, 0x002b   100000000
      Link is Active
```

- Verify the bundle statistics:

```
router# show policy-map interface bundle-ether 12000

Bundle-Ether12000 input: ingress

Class inet4-classifier-af1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          4647401962/21236124455654       26403040
    Transmitted         :          4647401962/21236124455654       26403040
    Total Dropped       :                   0/0                    0
Class inet4-classifier-af2
```

```
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          4502980177/20576584333939       25571493
    Transmitted         :          4502980177/20576584333939       25571493
    Total Dropped       :                   0/0                    0
Class inet4-classifier-af3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          4647404125/21236213667880       26389086
    Transmitted         :          4647404125/21236213667880       26389086
    Total Dropped       :                   0/0                    0
Class inet4-classifier-af4
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          9291188840/42456120548683       52771168
    Transmitted         :          9291188840/42456120548683       52771168
    Total Dropped       :                   0/0                    0
Class inet4-classifier-be1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          4647413429/21235847852686       26393414
    Transmitted         :          4647413429/21235847852686       26393414
    Total Dropped       :                   0/0                    0
Class inet4-classifier-nc1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          9294887621/42473100149807       52778258
    Transmitted         :          9294887621/42473100149807       52778258
    Total Dropped       :                   0/0                    0

Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0

Bundle-Ether12000 output: egress

Class c1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         16665494532/75878118942463       8760591
    Transmitted         :         16655834643/75834136022017       8760591
    Total Dropped       :             9659889/43982920446          0
  Queueing statistics
    Queue ID                              : None (Bundle)
    Taildropped(packets/bytes)            : 9659889/43982920446
Class c2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         16665421959/75877849543188       8718687
    Transmitted         :         16665421959/75877849543188       8718687
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                              : None (Bundle)
    Taildropped(packets/bytes)            : 0/0
Class c3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         16665247833/75877509455458       8703470
    Transmitted         :         16665187414/75877234624197       8703470
    Total Dropped       :               60419/274831261            0
  Queueing statistics
    Queue ID                              : None (Bundle)
    Taildropped(packets/bytes)            : 60419/274831261
Class c4
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         33330896131/151755393012945      17470745
    Transmitted         :         33330745421/151754709368565      17470745
    Total Dropped       :              150710/683644380            0
  Queueing statistics
    Queue ID                              : None (Bundle)
```

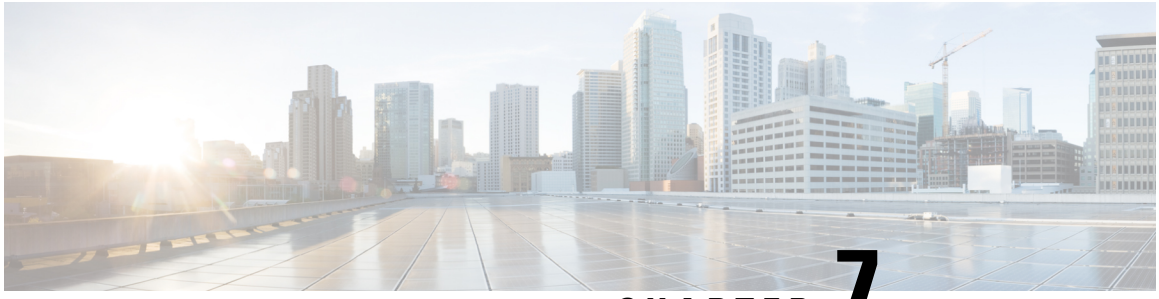
```
    Taildropped(packets/bytes)           : 150710/683644380
Class c5
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         16878910340/76849791869834       8833394
    Transmitted         :         16878849464/76849514633309       8833394
    Total Dropped       :               60876/277236525            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 60876/277236525
Class c6
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         33330898844/151756094112925      17456785
    Transmitted         :         33330752668/151755427708382      17456785
    Total Dropped       :              146176/666404543            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 146176/666404543
Class c7
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :              244106/79922040             74
    Transmitted         :              244106/79922040             74
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 0/0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :        267075066180/1215993441123215     139917482
    Transmitted         :        267075066180/1215993441123215     139917482
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 0/0
```

## Related Topics

- QoS on Link Bundles, on page 145

## Associated Commands

- bundle maximu-active links
- interface Bundle-Ether



CHAPTER 7

# Configuring Hierarchical Modular QoS

Hierarchical QoS (H-QoS) is a QoS model that enables you to specify QoS behavior at multiple levels of hierarchy. This chapter provides information about this feature and the different steps involved in configuring it.

***Table 24: Feature History for Hierarchical Modular QoS***

| Release | Modification |
|---|---|
| Release 6.3.1 | Initial Release |

This chapter covers the following topics:


- Overview of Hierarchical Modular QoS, on page 151
- Restrictions for Configuring H-QoS, on page 152
- Configuring Hierarchical Queuing, on page 153
- Conform Aware Hierarchical Policy Overview, on page 158


## Overview of Hierarchical Modular QoS

Hierarchical QoS (H-QoS) allows you to specify QoS behavior at multiple policy levels, which provides a high degree of granularity in traffic management.

H-QoS is applied on the router interface using nested traffic policies. The first level of traffic policy, the parent traffic policy, is used for controlling the traffic at the main interface or sub-interface level. The second level of traffic policy, the child traffic policy, is used for more control over a specific traffic stream or class. The child traffic policy, is a previously defined traffic policy, that is referenced within the parent traffic policy using the **service-policy** command.

Two-level H-QoS is supported on both ingress and egress directions on all line cards and on physical or bundle main interfaces and sub-interfaces.

Three-level Hierarchical QoS (H-QoS) enables enforcement of class/service, group/ Ethernet Flow Point (EFP), and port level SLAs. You can apply regular two-level egress H-QoS policies on the sub-interfaces to achieve class and EFP SLAs at child and parent levels. In addition, you can apply a port shaper policy on the main interface to achieve an aggregated port level SLA in a 1+2 H-QoS or three-level H-QoS model.

An important point to note is that before Release 6.6.25 (where the three-level H-QoS capability was introduced), when you applied class-default shaper on a main interface, it was enforced *only* on the traffic going through the main interface. With three-level HQoS, a class default shaper that is applied on the main

interface is considered as a port shaper and enforced on *all* traffic going out of that physical port. The advantage of three-level H-QoS is that the parent shaper on the sub-interfaces is allowed to oversubscribe, thus enabling best effort sharing of the aggregate port shaper at the third level.

**Note** From Cisco IOS XR Release 7.3.1 onwards, systems with Cisco NC57 line cards running in compatibility mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

Starting with Cisco IOS XR Release 7.4.1 systems with Cisco NC57 line cards running in native mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

# Restrictions for Configuring H-QoS

The following restrictions are applicable while configuring H-QoS:

1. The parent traffic policy only supports the traffic class of type class-default.
2. The parent traffic policy only supports the class-action **shape** and no other queuing action can be configured in it.
3. While configuring on the router, it is mandatory that the priority class must have traffic shaper in the child traffic policy.
4. The sum of the bandwidth of the child policies must be less than the parent policy's traffic shaper.
5. For congestion avoidance and management, the traffic shaper in the parent traffic policy calculates the queue limit and drop priority.
6. H-QoS profile and ingress peering profile don't work simultaneously. Hence, features requiring a peering profile also do not work with the H-QoS profile enabled.
7. PBTS feature does not work when the H-QoS profile is enabled. This is due to TCAM limitations.
8. A maximum of 896 bundle sub-interfaces are only supported in the system, even if there are no QoS policies applied. This is due to an internal LAG_ID resource consumption in HQoS profile mode for bundle sub-interfaces with or without QoS policies being applied.
9. A maximum of 4 priority levels are only supported in HQoS profile mode unlike the default mode where 7-priority levels are supported. The restriction also applies to physical and bundle main interface policies where 7-level priorities were previously used in non-H-QoS profile mode.
10. Bandwidth and Bandwidth remaining configurations are not supported simultaneously within the same policy-map. If a class has bandwidth (CIR), other classes must also have only bandwidth configuration. If a class-map has bandwidth remaining percent/ratio (EIR), other classes should also have only the bandwidth remaining configuration. Shaping is applied on any class.

11. In HQOS Mode, if multiple queues are configured with BRR and there is high congestion on the LP (Low Priority) queues then one queue with BRR gets more credits than rest of the LP queues. The deviation is proportional to the congestion in LP queues.
12. Priority classes must have rate limit configuration by using a Shaping configuration. The effective shaper value is taken as priority bandwidth reservation. Sum of priority bandwidth reservations across all sub-interfaces and main interfaces must not exceed the network interface (NIF) port speed. This is to avoid over-subscription of priority traffic across the network interface port.

    Rates of non-priority classes and parent shaping can be over-subscribed.
13. The granularity of bandwidth or bandwidth remaining ration (BRR) is 1:64 as compared to 1:4096 in non-hqos mode. So, there could be accuracy differences in bandwidth performance based on the values used.
14. Filtering for egress IPv4 and IPv6 multicast traffic is not supported if H-QoS is configured on the router.

The following restrictions are applicable while configuring three-level H-QoS:

- There is no support for bandwidth action at the EFP parent level. All EFP/sub-interface policies get a fair share of the port shaper.
- Three-level H-QoS does not apply to ingress policies or to egress marking policies.
- Executing **clear qos counters** on the main interface clears only the main interface policy statistics. Use the “all” option to clear all sub-interface statistics or alternately, clear the sub-interface policy statistics individually.
- Main interface policy statistics do not reflect the sub-interface packet / byte counters, although the port shaper is enforced on all logical ports for a given physical interface. The sub-interface policy-map statistics reflect the transmitted and dropped packet/byte count post-port shaper enforcement.

# Configuring Hierarchical Queuing

Before you configure H-QoS, you must enable the H-QoS profile on the router. After enabling H-QoS profile, reload the router, as shown in the following configuration.

```
admin
hw-module location all reload
Router# configure
Router(config)# hw-module profile qos hqos-enable
Router(config)# commit
Router# admin
sysadmin-vm:0_RP0# hw-module location all reload
```

The steps that are involved in configuring hierarchical queuing are as follows:

1. Configure a class-map.
2. Configure a child traffic policy using the class-map that was configured in the previous step.
3. Configure a parent traffic policy and add the child traffic policy in it.

The parent traffic policy is the H-QoS traffic policy and it can be applied on physical or bundle main interfaces and sub-interfaces.

### Configuration Example

Configuration of a class-map is as follows:

```
Router# configure
Router(config)# class-map match-any tc2
Router(config-cmap)# match traffic-class 1
Router(config-cmap)# end-class-map
Router(config)# commit
```

Configuration of a child traffic policy is as follows:

```
Router# configure
Router(config)# policy-map child
Router(config-pmap)# class tc2
Router(config-pmap-c)# shape average percent 20
Router(config-pmap-c)# exit
Router(config-pmap)# class class-default
Router(config-pmap-c)# shape average percent 1
Router(config-pmap)# end-policy-map
Router(config)# commit
```

Configuration of a parent traffic policy is as follows:

```
Router# configure
Router(config)# policy-map parent
Router(config-pmap)# class class-default
Router(config-pmap-c)# service-policy child
Router(config-pmap-c)# shape average percent 50
Router(config-pmap)# end-policy-map
Router(config)# commit
```

### Running Configuration

```
/* Configuration of a Class-map */
class-map match-any tc2
 match traffic-class 1
 end-class-map
!
/* Configuration of a Child Traffic Policy */
policy-map child
 class tc2
  shape average percent 20
 !
 class class-default
  shape average percent 1
 !
 end-policy-map
!
/* Configuration of a Parent Traffic Policy */
policy-map parent
 class class-default
  service-policy child
  shape average percent 50
 !
 end-policy-map
!
```

### Applying the Parent Traffic Policy on a Main Interface

```
Router# configure
Router(config)# Interface TenGigE 0/0/0/10
Router(config-int)# service-policy output parent
Router(config-int)# commit
```

### Applying the Parent Traffic Policy on a Sub-interface

```
Router# configure
Router(config)# Interface TenGigE 0/0/0/10.1
Router(config-int)# service-policy output parent
Router(config-int)# commit
```

### Verification

Verify if the H-QoS traffic policy is applied correctly on the interface using the commands **show qos interface** *interface-name* **output**. In the following example, the **Level1 Class** gives information about the class-map that is associated with the parent traffic policy and the **Level2 Class** gives information about the class-maps that are associated with the child traffic policy.

```
RP/0/RP0/CPU0:ios#show qos interface ten0/0/0/10 output

NOTE:- Configured values are displayed within parentheses
Interface TenGigE0/0/0/10 ifh 0x1e0  -- output policy
NPU Id:                        0
Total number of classes:       3
Interface Bandwidth:           10000000 kbps
VOQ Base:                      1136
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                              =   class-default
Queue Max. BW.                            =   no max (50 %)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   0 / (BWR not configured)
   Level2 Class                              =   tc2
   Egressq Queue ID                          =   1138 (LP queue)
   Queue Max. BW.                            =   1020015 kbps (20 %)
   Queue Min. BW.                            =   0 kbps (default)
   Inverse Weight / Weight                   =   1 / (BWR not configured)
   Guaranteed service rate                   =   1000000 kbps
   TailDrop Threshold                        =   1253376 bytes / 10 ms (default)
   WRED not configured for this class
   Level2 Class                              =   class-default
   Egressq Queue ID                          =   1136 (Default LP queue)
   Queue Max. BW.                            =   50625 kbps (1 %)
   Queue Min. BW.                            =   0 kbps (default)
   Inverse Weight / Weight                   =   1 / (BWR not configured)
   Guaranteed service rate                   =   50000 kbps
   TailDrop Threshold                        =   62720 bytes / 10 ms (default)
   WRED not configured for this class
```

The statistics for the packets that have matched the different traffic classes of the parent and child traffic policies can be viewed using the command **show policy-map interface** *interface-name* **output**. Also, this command also shows the number of packets that are transmitted or dropped when the specified action is applied on the packets that have matched the respective traffic class.

```
Router# show policy-map interface ten0/0/0/10 output

TenGigE0/0/0/10 output: parent
```

```
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          2313578823/296138089344         8494665
    Transmitted         :           232805738/29799134464          854465
    Total Dropped       :          2080773085/266338954880         7640200
  Policy child Class tc2
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :          2313578823/296138089344         8494665
      Transmitted         :           232805738/29799134464          854465
      Total Dropped       :          2080773085/266338954880         7640200
    Queueing statistics
      Queue ID                              : 1138

      Taildropped(packets/bytes)            : 2080773085/266338954880
  Policy child Class class-default
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :                  0/0                     0
      Transmitted         :                  0/0                     0
      Total Dropped       :                  0/0                     0
    Queueing statistics
      Queue ID                              : 1136

      Taildropped(packets/bytes)            : 0/0
```

When using hierarchical policers, there is no independent set of hardware counters to store the parent policer statistics. Instead, parent policer statistics are manipulated in the software to be the sum of all child policers under the same policy-map.

This is shown in the following example where two streams of traffic, with CoS value of 1 and 2 are sent at a speed of 3.5 Gbps each.

```
/*Hierarchical Policy Map Configuration*/
=====================================================
Router# show running-config policy-map Hingress
policy-map Hingress
 class class-default
  service-policy ingress
  police rate 5 gbps peak-rate 9 gbps
  !
 !
 end-policy-map
!
 /*Ingress Policy Map Configuration*/
=====================================
Router#show running-config policy-map ingress
policy-map ingress
 class cos1
  set traffic-class 1
  police rate 5 gbps
  !
 !
 class cos2
  set traffic-class 2
  police rate 5 gbps
  !
 !
 class class-default
 !
 end-policy-map
!
 /*Policy Map applied at TenGigE0/0/0/6.100 Interface*/
=====================================================
Router#show policy-map interface tenGigE 0/0/0/6.100 input
```

```
TenGigE0/0/0/6.100 input: Hingress

Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           856717937/109659895936         6683676
    Transmitted         :           856717937/109659895936         6683676
    Total Dropped       :                   0/0                    0
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :           856717937/109659895936         6683674
    Policed(exceed)     :                   0/0                    0
    Policed(violate)    :                   0/0                    0
    Policed and dropped :                   0/0

  Policy ingress Class cos1
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :           437826303/56041766784          3341838
      Transmitted         :           437826303/56041766784          3341838
      Total Dropped       :                   0/0                    0
    Policing statistics                (packets/bytes)     (rate - kbps)
      Policed(conform)    :           437826303/56041766784          3341838
      Policed(exceed)     :                   0/0                    0
      Policed(violate)    :                   0/0                    0
      Policed and dropped :                   0/0
      Policed and dropped(parent policer)  : 0/0

  Policy ingress Class cos2
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :           418891634/53618129152          3341838
      Transmitted         :           418891634/53618129152          3341838
      Total Dropped       :                   0/0                    0
    Policing statistics                (packets/bytes)     (rate - kbps)
      Policed(conform)    :           418891634/53618129152          3341838
      Policed(exceed)     :                   0/0                    0
      Policed(violate)    :                   0/0                    0
      Policed and dropped :                   0/0
      Policed and dropped(parent policer)  : 0/0

  Policy ingress Class class-default
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :                   0/0                    0
      Transmitted         :                   0/0                    0
      Total Dropped       :                   0/0                    0
Policy Bag Stats time: 0
Policy Bag Stats time: 0
```

### Configuration Example for Three-Level H-QoS

To configure three-level H-QoS:

1. Configure the port shaper or EFP group shaper.
2. Configure the two-level H-QoS policy with an EFP parent shaper and class or service level actions
3. Enable the port or EFP group shaper on the main interface to become the root policy.
4. Enable the two-level H-QoS policy on each EFP instance, thus providing a three-level hierarchy of service, EFP and EFP group or port SLAs.

A sample configuration for three-level H-QoS is as follows.

```
policy-map port_shaper
 class class-default
```

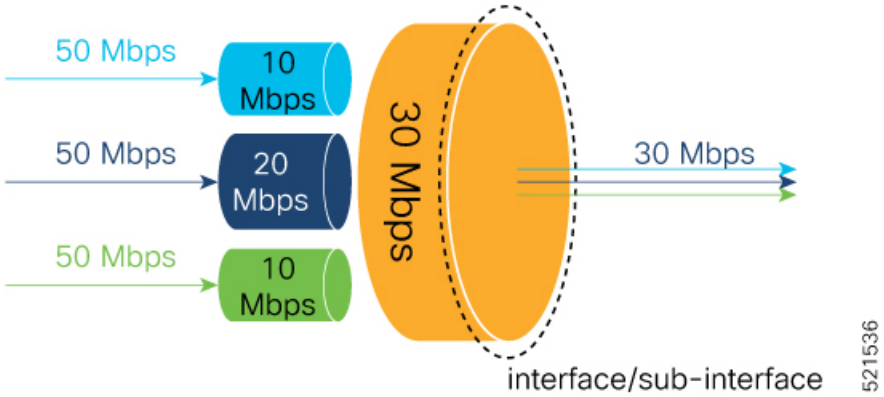
```
  shape average 6 gbps
 !
 end-policy-map
!

policy-map efp_policy
 class class-default
  service-policy efp_policy_child
  shape average 4 gbps
 !
 end-policy-map

!
policy-map efp_policy_child
 class tc1
  shape average 50 mbps
  priority level 1
 !
 class tc2
  bandwidth percent 50
 !
 class tc3
  bandwidth percent 30
 !
 class class-default
 !
 end-policy-map
!

interface TenGigE0/5/0/4
 service-policy output port_shaper
!

interface TenGigE0/5/0/4.1
 service-policy output efp_policy
 encapsulation dot1q 11
!

interface TenGigE0/5/0/4.2
 service-policy output efp_policy
 encapsulation dot1q 12
!
```

### Verification

Run the **show policy-map interface** command in XR EXEC mode to display the packet/byte count and rate post port shaper enforcement on each of the sub-interface/EFP policies.

# Conform Aware Hierarchical Policy Overview

Hierarchical QoS (H-QoS), while allowing for granular and multi-level management of traffic, does not allow for conform traffic from a child-level policy to a parent-level policy to get priority. This means that in case of excess traffic, the parent policer drops conform traffic packets from the child level as well.




The conform-aware hierarchical policy feature enables the parent-level policy to prioritize conform traffic from child-level policy over exceed and violate traffic.

Here is how it works: the child-level policer initially marks its packets as red, yellow, or green. Packets are marked based on the committed information rate (CIR) value and the two associated burst sizes - committed burst size (CBS) and excess burst size (EBS). If a packet does not exceed the CBS, it is marked as conformed packet (green). The packet is marked as exceeded if it exceeds CBS, but not the EBS (yellow). If it exceeds the EBS as well, it is marked as violate packet (red).

When the packets arrive at the parent level policer (which is color aware), the policer prioritizes the packets marked green over the packets marked yellow. After all the conform traffic (green) is transmitted and there are tokens available still, the yellow packets are transmitted next, instead of being marked as violate traffic (red).

To enable the conform-aware hierarchical policy feature run the command.

```
hw-module profile qos conform-aware-policer
```

# Conform Aware Hierarchical Policy Restrictions

The parent-policy traffic police rate must be greater than or equal to the sum of child conform rates.

# Configuring Conform Aware Hierarchy Policy

To enable and configure shared policer:

1. Run the **hw-module profile qos conform-aware-policer** command.
2. Reload the affected line card.
3. Configure class maps to be used for matching packets to the class specified.
4. Create a child policy map.
5. Configure traffic policing for the classes in the child policy map.
6. Attach the child policy-map to the parent's class-default class.
7. Configure traffic policing for the parent policy map.

```
RP/0/RP0/CPU0:ios(config)#hw-module profile qos conform-aware-policer
RP/0/RP0/CPU0:router# reload location 0/0/CPU0
```

```
RP/0/RSP0/CPU0:ios(config)#class-map match-any c1
RP/0/RSP0/CPU0:ios(config-cmap)#match precedence 1
RP/0/RSP0/CPU0:ios(config-cmap)#end-class-map
RP/0/RSP0/CPU0:ios(config)#class-map match-any c2
RP/0/RSP0/CPU0:ios(config-cmap)#match precedence 2
RP/0/RSP0/CPU0:ios(config-cmap)#end-class-map
RP/0/RSP0/CPU0:ios(config-pmap)# policy-map child
RP/0/RSP0/CPU0:ios(config-pmap-c)#class class1
RP/0/RSP0/CPU0:ios(config-pmap-c-police)#police rate 20 mbps peak-rate 50 mbps
RP/0/RSP0/CPU0:ios(config-pmap-c)#class class2
RP/0/RSP0/CPU0:ios(config-pmap-c-police)#police rate 30 mbps peak-rate 60 mbps
RP/0/RSP0/CPU0:ios(config-pmap)#policy-map parent
RP/0/RSP0/CPU0:ios(config-pmap-c)#class class-default
RP/0/RSP0/CPU0:ios(config-pmap-c)#service-policy child
RP/0/RSP0/CPU0:ios(config-pmap-c-police)#police rate 60 mbps
RP/0/RSP0/CPU0:ios(config-pmap-c-police)#exit
RP/0/RSP0/CPU0:ios(config-pmap-c)#exit
RP/0/RSP0/CPU0:ios(config-pmap)#exit
RP/0/RSP0/CPU0:ios(config-if)#commit
```

### Running Configuration

```
class-map match-any class1
match precedence 1
end-class-map

class-map match-any class2
match precedence 2
end-class-map

policy-map child
class class1
  police rate 20 mbps peak-rate 50 mbps
  !
!
class class2
  police rate 30 mbps peak-rate 60 mbps
  !

policy-map parent
class class-default
  service-policy child
  police rate 60 mbps
!
!
```

In this example, the 20 mbps rate in child class1 and 30 mbps rate in child class2 are guaranteed as conform traffic and the parent will not drop them.

### Verification

Run the **show policy-map interface** command in XR EXEC mode to confirm that the committed information rate (CIR) is prioritized over the peak information rate (PIR).

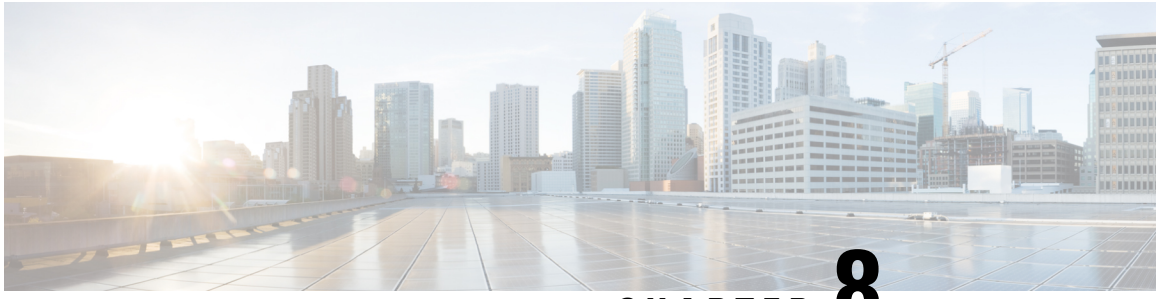
In the example below, **Policed (conform)** or CIR from each class is prioritized over **Policed (exceed)** or PIR.

```
RP/0/RP0/CPU0:ios(config-if)#do show running-config policy-map parent_conform
Tue Apr 28 11:19:09.785 UTC
policy-map parent_conform
class class-default
  service-policy child_conform
  police rate 80 mbps peak-rate 120 mbps
  !
```

```
 !
 end-policy-map
!

RP/0/RP0/CPU0:ios(config-if)#do show running-config policy-map child_conform
Tue Apr 28 11:19:14.219 UTC
policy-map child_conform
class match_prec1
  police rate 30 mbps peak-rate 35 mbps
  !
  set qos-group 3
!
 class match_prec2
  police rate 20 mbps peak-rate 25 mbps
  !
  set qos-group 4
!
 class class-default
  police rate 20 mbps peak-rate 30 mbps
  !
 !
 end-policy-map
!

RP/0/RP0/CPU0:ios(config-if)#do show policy-map interface tenGigE 0/0/0/11/2 input
Tue Apr 28 11:23:48.915 UTC

TenGigE0/0/0/11/2 input: parent_conform

Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           217797200/111512166400         2344847
    Transmitted         :             8314388/4256966656           88089
    Total Dropped       :           209482812/107255199744         2256758

  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :             6602174/3380313088           69926
    Policed(exceed)     :             1712214/876653568            18165
    Policed(violate)    :           209482812/107255199744         2256782
    Policed and dropped :           209482812/107255199744

  Policy child_conform Class match_prec1
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :            54449300/27878041600          586215
      Transmitted         :             3246813/1662368256           34399
      Total Dropped       :            51202487/26215673344          551816
    Policing statistics                (packets/bytes)     (rate - kbps)
      Policed(conform)    :             2818471/1443057152           29851
      Policed(exceed)     :              428342/219311104            4547
      Policed(violate)    :            51202487/26215673344          551816
      Policed and dropped :            51202487/26215673344
      Policed and dropped(parent policer)  : 0/0

  Policy child_conform Class match_prec2
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :            54449300/27878041600          586213
      Transmitted         :             2319731/1187702272           24577
     Total Dropped       :            52129569/26690339328          561636
    Policing statistics                (packets/bytes)     (rate - kbps)
      Policed(conform)    :             1891851/968627712            20037
      Policed(exceed)     :              427880/219074560            4540
      Policed(violate)    :            52129569/26690339328          561636
      Policed and dropped :            52129569/26690339328
      Policed and dropped(parent policer)  : 0/0
```

```
  Policy child_conform Class class-default
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :           108898600/55756083200          1172419
      Transmitted         :             2747844/1406896128           29113
      Total Dropped       :           106150756/54349187072          1143306
    Policing statistics                (packets/bytes)     (rate - kbps)
      Policed(conform)    :             1891852/968628224            20036
      Policed(exceed)     :              855992/438267904            9076
      Policed(violate)    :           106150756/54349187072          1143306
      Policed and dropped :           106150756/54349187072
      Policed and dropped(parent policer)  : 0/0
Policy Bag Stats time: 1588072996538
```

**Related Commands** hw-module profile qos conform-aware-policer



CHAPTER 8

# Configure Egress Traffic Management

This chapter covers the following topics:


- Egress Traffic Management, on page 164
- More About the Egress Traffic Management Model, on page 166
- QoS Policy Configuration Rules, on page 167
- Configure Egress Traffic Management , on page 168
- Low Rate Shaper, on page 184

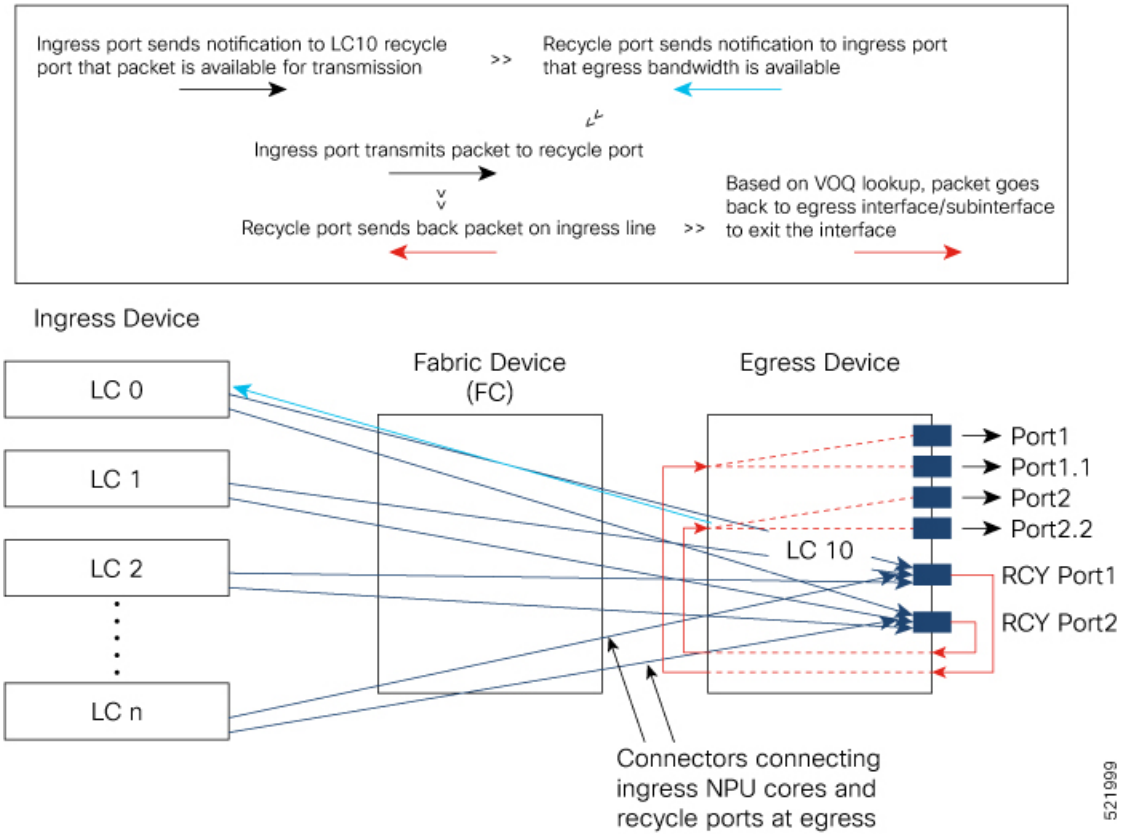
# Egress Traffic Management

*Table 25: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Egress Traffic Management | Release 7.6.1 | The egress traffic management model employs a two-pass model per NPU core to get up to 50% of NPU core bandwidth as recycle bandwidth. VOQ lookup takes place in the second pass to get the outgoing interface or sub-interface VOQ. This lookup at the second pass enhances the egress queuing policy by making the feature-rich ingress pipeline available to the outgoing traffic.<br><br>Also, this functionality is configurable at the port level and not at the global level. This approach limits the usage of VOQ connectors required on all ingress devices to only physical ports or main interfaces, thus enabling scaling of egress QoS policies.<br><br>This functionality introduces the **mode etm** option for the **controller optics** command. |

The egress traffic management model modifies the existing Ingress Traffic Management Model, on page 112 to ensure you can configure higher QoS scales. It does so by using a recycle model at egress NPU so that the packet buffering takes place at egress NPU. (See figure, **The Egress Traffic Management Model**.) Also, egress traffic management is configurable at the port-level and not at the global level, to maximize flexibility.

*Figure 8: The Egress Traffic Management Model*




In the figure, every ingress NPU port (LC 0 through LC n) has eight VOQs for the single egress NPU LC 10.

Here's how packet transmission takes place:

1. When a packet arrives at ingress port (say on LC 0), the forwarding lookup on ingress line card points to the egress interface that could be physical, subinterface, Link aggregation Group (LAG), or LAG subinterface. In the case of LAG, hashing takes place at the ingress line card to select the LAG member from where the packet will be sent out. Based on the egress interface (say it is on LC10), the packet is enqueued to the VOQ of LC 10. The egress interface is always mapped to a physical port.

2. However, unlike in the ingress traffic management model where the ingress NPU ports connect to the egress NPU ports (per traffic management hierarchy), the egress traffic management model connects ingress NPU ports to the recycle port that's mapped to the egress port.

3. Once egress bandwidth is available, the recycle ports (based on the packet marking and distribution model) send grants to the ingress ports via the connectors. (The figure shows a separate line for the grant for the sake of visual representation. In reality, the same connector is used for requests, grants, and transmission between an NPU core at the ingress and the corresponding recycle port.)

4. The ingress NPU responds to this permission by transmitting the packet via the fabric device (FC) to the egress NPU.

5. Once the packet reaches egress NPU, it gets recycled back to the ingress pipeline of the egress NPU. A VOQ lookup takes place at this stage that checks if there's a policy on the subinterface. If there's no policy on the subinterface, the main interface VOQ is taken. This behavior is the same as in the ingress traffic

management model, except that another VOQ lookup for subinterface and the main interface takes place in the second pass (recycle) on the egress NPU core.

6. Based on the lookup in the second pass, the packets go back into the egress to the specified main interface or subinterface. It's also in the second pass that any queueing action kicks in before the packets go out the egress interface.

# More About the Egress Traffic Management Model

- **Which routers and line cards support the egress traffic management model?**

  The egress traffic management model is supported on routers that have the following Cisco NC57 line cards that are installed and that operate in the native mode:

  - NC57-18DD-SE
  - NC57-36H-SE

  This feature is also supported on the following Cisco NCS-57B1 chassis variants:

  - NCS-57B1-5D24H-SE
  - NCS-57B1-5DSE-SYS

- **What benefits does the egress traffic management model offer?**

  - Configuring the egress traffic management functionality offers you **higher scales of egress QoS policies**. And this is because of the way the egress traffic management model is designed, reducing the size of the mesh of connectors to main and subinterface ingress ports.
  - Plus, because of this model (where a second lookup and queuing action takes place in the recycled pass), the **egress queuing policy is enhanced**. An egress traffic management queuing policy contains the same match criteria that ingress policies typically contain. (See the following table for details.) In addition, match on traffic-class and remarking traffic-class is also supported in egress with the **hw-module profile qos ipv6 short-etm** command. (For details, see Configure Egress Traffic Management , on page 168.) This means that the egress traffic management model offers you with even more flexibility in planning your egress policies, while supporting higher scale numbers.

| Flow Type | Egress Classification Type Supported |
|---|---|
| IPv4/IPv6 | • Prec / DSCP<br>• SIP / DIP / SP / DP / Protocol / TCP-CTL via ACLs<br>• COS / DEI (for VLAN)<br>• Fragmentation |
| MPLS | • IPv4 ToS / IPv6 TC (for underlying for IPv4/IPv6)<br>• COS / DEI (for VLAN)<br>• MPLS Exp |

| Flow Type | Egress Classification Type Supported |
|---|---|
| Layer 2 | • COS / DEI (for VLAN)<br>• COS Inner / DEI Inner (for VLAN)<br>• IPv4 ToS<br>• IPv6 TC<br>• MPLS Exp |

- There's another important benefit that the egress traffic management model offers. You need higher QoS scale and features only for access- or edge-facing interfaces, and not on core-facing interfaces. The egress traffic management model provides the **flexibility to configure recycle mode only on access facing ports** while continue using core facing ports in single-pass mode.
- Configuring the egress traffic management functionality enables the low rate shaper feature. For details, see Low Rate Shaper, on page 184.

- **In the recycle pass, packets are sent back on the ingress pipeline. What's the impact?**—The impact is **reduced throughput**. Recycle traffic competes with ingress traffic on the ingress pipeline; this impacts the traffic throughput. Thus, higher scales come at the cost of reduced throughputs. In such contention cases, device throughput drops down to 50%.
- **How do I handle Link Aggregation Group (LAG) configurations?**—All LAG members must have the same mode. This means that you must **configure the egress traffic management mode on all LAG members**.
- **Which MDB profiles are supported for the egress traffic management model?**—All SE-based MDB profiles are supported for the egress traffic management model. (SE-based MDB profiles are the scaled versions that allow for configuring more QoS policies on systems using eTCAM.)
- **Important Note on LOCP Statistics**—When you attach a policy map to a subinterface in the egress traffic management model, the Locally Originated Control Protocol (LOCP) packets exiting that subinterface are accounted for in the subinterface policy counters.

  Without the egress traffic management model, egress inject LOCP packets are not accounted for in sub-interface egress policy counters.

# QoS Policy Configuration Rules

- The **set traffic-class** configuration for all traffic classes is mandatory at egress for egress traffic management ports, regardless of your ingress configuration. This **set traffic-class** *val* represents the VOQ offset that the matching packets select. The only exception to this rule is the class-default configuration.
- The class-default can have a **set traffic-class** *val* other than 0.
- Even if the ingress and egress traffic classes match, you'll still need to configure a **set traffic-class** *val* at egress. If the match and set values differ, then the traffic-class is remarked. For example, say a traffic-class is set to 1 in the ingress policy and the match on that traffic-class is 1 on the egress in the

egress traffic management policy. Then, say, at the second pass, the **set traffic-class** *val* is remarked to 5. The packets that match this class in the egress traffic management policy go to VOQ offset 5.

- You can set traffic-class for multiple traffic classes to the same value. In such a situation, traffic matching those classes head to the same VOQ. Therefore, the statistics for all these classes point to the same VOQ counters. It's important to note that when multiple traffic classes have the same value in the **set traffic-class** *val* action, then the queuing parameters (for example, priority or bandwidth remaining, queue depth, WRED) of these classes MUST be the same.
- In a hierarchical policy, you can configure the **set traffic-class** *val* only for the child traffic classes. The parent classes can't have the **set traffic-class** *val* action.

# Configure Egress Traffic Management

### Before You Begin

Some critical points to note before you begin:

- Before you proceed any further, ensure that you delete your existing **hw-module profile qos hqos-enable** configuration and restart your system. Only then can you enable egress traffic management at the port level. H-QoS profiles and egress traffic management configurations can't coexist.
- The egress traffic management mode is configured on a per-port mode. This ensures that the core-facing ports continue using the current ingress-based VOQ model, while the access ports use the egress traffic management model. Such an approach reduces the consumption of the recycle bandwidth. See the following table for details.

*Table 26: Global H-QoS versus Per Port Configuration*

| **Global H-QoS with hw-module profile qos hqos-enable** | | **Per Port Egress Traffic Management (ETM) without hw-module profile qos hqos-enable** | | | |
|---|---|---|---|---|---|
| **All Ports** | | **ETM ports with controller optics mode etm configured** | | **Non-ETM ports** | |
| **Main interface** | **Subinterface** | **Main interface** | **Subinterface** | **Main interface** | **Subinterface** |
| H-QoS and flat policies | H-QoS and flat policies | H-QoS and flat policies | H-QoS and flat policies | H-QoS and flat policies | No policies, inherit from main |

- The configuration is implemented at the controller level. This means that when you commit the egress traffic management configuration, **all existing configurations are deleted. All existing features and subinterface configurations are deleted and you must reconfigure all of them.**

### Configuration Example for Policy Map with Class Match on traffic-class

To configure an egress traffic management queueing policy and reconfigure policies in **etm** mode for traffic class:

1. Delete the existing **hw-module profile qos hqos-enable** configuration and restart your system.

2. Configure NPU native mode by using the **hw-module profile npu native-mode-enable** command.
3. In the admin mode, reload the router by using the **hw-module reload location all** command.
4. Enable egress traffic management (**etm**) on an interface using the **controller optics** command in the optics controller configuration mode. For 10 GbE interfaces, configure **etm** on the breakout port. After this configuration is enabled, it deletes all subinterfaces under the main interface.
5. Configure the subinterfaces. For policy maps with **match-all** specified for **traffic-class**, run **hw-module profile qos ipv6 short-etm** to truncate IPv6 fields. This configuration frees TCAM space for packet header marking for VOQ lookup in the second pass. Reload the line card.
6. Create the class maps.
7. Specify the match criteria, which can be: match on traffic class, DSCP, CoS, or any combination of these criteria.
8. Set the traffic classes for each class in the policy map.

**Note** The **class-default** need not have a **set-traffic class** value.

9. Specify the traffic class actions.
10. Attach the policy map to the subinterface.

```
/*Delete existing h-qos configuration*/
Router#configure terminal
Router(config)#no hw-module profile qos hqos-enable
In order to activate this new qos profile, you must manually reload the chassis/all line
cards
Router(config)#commit
Router(config)#exit
Router#reload


/* Configure NPU native mode*/
Router#(config)#hw-module profile npu native-mode-enable
In order to activate this new npu profile, you must manually reload the chassis
Router#(config)#commit
Router#(config)#exit
Router#admin
sysadmin-router#hw-module location all reload
Reloading the module will be traffic impacting if not properly drained. Continue to Reload
 hardware module ? [no,yes] yes


/*Enable etm on an interface*/
Router#configure terminal
Router(config)#controller optics 0/5/0/16
Router(config-Optics)#mode etm
Router(config-Optics)#commit
Router(config-Optics)#exit

/*To enable etm for breakout ports, as for 10 GbE*/
Router#configure terminal
Router(config)#controller optics 0/5/0/27/1
Router(config-Optics)#mode etm
Router(config-Optics)#commit
Router(config-Optics)#exit
```

```
/*Match on traffic-class on ETM-enabled interfaces requires running the following hw-module
 command to truncate  IPv6 fields, thus freeing TCAM space for packet header marking for
VOQ lookup in the second pass.*/
Router(config)#hw-module profile qos ipv6 short-etm
In order to activate this profile, you must manually reload the chassis/all line cards
Router(config)#commit
Router(config)#exit
Router#reload

/*Configure etm policies on subinterfaces
Configure class map. Specify the match criteria*/
Router(config)#class-map match-any MAIN_OUT_CMAP_1
Router(config-cmap)#match traffic-class 1
Router(config-cmap)#end-class-map
Router(config-cmap)#exit
Router(config)#class-map match-any MAIN_OUT_CMAP_2
Router(config-cmap)#match traffic-class 2
Router(config-cmap)#end-class-map
Router(config-cmap)#exit
Router(config)#class-map match-any MAIN_OUT_CMAP_3
Router(config-cmap)#match traffic-class 3
Router(config-cmap)#end-class-map
Router(config-cmap)#exit
Router(config)#class-map match-any MAIN_OUT_CMAP_4
Router(config-cmap)#match traffic-class 4
Router(config-cmap)#end-class-map
Router(config-cmap)#exit
Router(config)#class-map match-any MAIN_OUT_CMAP_5
Router(config-cmap)#match traffic-class 5
Router(config-cmap)#end-class-map
Router(config-cmap)#exit
Router(config)#class-map match-any MAIN_OUT_CMAP_6
Router(config-cmap)# match traffic-class 6
Router(config-cmap)#end-class-map
Router(config-cmap)#exit
Router(config)#class-map match-any MAIN_OUT_CMAP_7
Router(config-cmap)#match traffic-class 7
Router(config-cmap)#end-class-map
Router(config-cmap)#exit

/*Create a policy map*/
Router(config)#policy-map MAIN_OUT_PMAP5_PHY_SUB

/*Associate the traffic class with the policy map*/
Router(config-pmap)#class MAIN_OUT_CMAP_1

/*Specify the traffic class actions ('bandwidth remaining ratio, in this example)*/
Router(config-pmap-c)#bandwidth remaining ratio 20
Router(config-pmap-c)#set traffic-class 1
Router(config-cmap-c)#commit
/*Repeat for other classes*/
Router(config-pmap-c)#class MAIN_OUT_CMAP_2
Router(config-pmap-c)#bandwidth remaining ratio 30
Router(config-pmap-c)#set traffic-class 2
Router(config-pmap-c)#exit
Router(config-pmap-c)#class MAIN_OUT_CMAP_3
Router(config-pmap-c)#bandwidth remaining ratio 40
Router(config-pmap-c)#set traffic-class 3
Router(config-pmap-c)#exit
Router(config-pmap-c)#class MAIN_OUT_CMAP_4
Router(config-pmap-c)#bandwidth remaining ratio 50
Router(config-pmap-c)#set traffic-class 4
```

```
Router(config-pmap-c)#exit
Router(config-pmap-c)#class MAIN_OUT_CMAP_5
Router(config-pmap-c)#bandwidth remaining ratio 60
Router(config-pmap-c)#set traffic-class 5
Router(config-pmap-c)#exit
Router(config-pmap-c)#class MAIN_OUT_CMAP_6
Router(config-pmap-c)#bandwidth remaining ratio 60
Router(config-pmap-c)#set traffic-class 6
Router(config-pmap-c)#exit
Router(config-pmap-c)#class MAIN_OUT_CMAP_7
Router(config-pmap-c)#set traffic-class 7
Router(config-pmap-c)#priority level 1
Router(config-pmap-c)#exit
Router(config-pmap-c)#class class-default
Router(config-pmap-c)#bandwidth remaining ratio 10
Router(config-pmap-c)#exit
Router(config-pmap-c)#end-policy-map
Router(config)#exit
Router(config)#policy-map PARENT_MAIN_OUT_PHY_SUB
Router(config-pmap)#class class-default
Router(config-pmap-c)#service-policy MAIN_OUT_PMAP5_PHY_SUB
Router(config-pmap-c)#shape average 55 mbps
Router(config-pmap-c)#exit
Router(config-pmap-c)#end-policy-map
Router(config)#commit

/*Attach the policy map to the subinterface.*/
Router#configure
Router(config)#interface HundredGigE 0/5/0/16.1
Router(config-subif)#service-policy output PARENT_MAIN_OUT_PHY_SUB
Router(config-subif)#commit
```

You have enabled **etm** mode and configured policies in this mode. You are now ready to experience existing QoS features with enhanced egress classification for queueing policy and scale egress queueing policies.

### Running Configuration

```
controller Optics 0/5/0/16
  mode etm
!
class-map match-any MAIN_OUT_CMAP_1
 match traffic-class 1
 end-class-map
!
class-map match-any MAIN_OUT_CMAP_2
 match traffic-class 2
 end-class-map
!
class-map match-any MAIN_OUT_CMAP_3
 match traffic-class 3
 end-class-map
!
class-map match-any MAIN_OUT_CMAP_4
 match traffic-class 4
 end-class-map
!
class-map match-any MAIN_OUT_CMAP_5
 match traffic-class 5
 end-class-map
!
class-map match-any MAIN_OUT_CMAP_6
 match traffic-class 6
 end-class-map
```

```
!
class-map match-any MAIN_OUT_CMAP_7
 match traffic-class 7
 end-class-map
!
policy-map MAIN_OUT_PMAP5_PHY_SUB
 class MAIN_OUT_CMAP_1
  bandwidth remaining ratio 20
  set traffic-class 1
!
 class MAIN_OUT_CMAP_2
  bandwidth remaining ratio 30
  set traffic-class 2
!
 class MAIN_OUT_CMAP_3
  bandwidth remaining ratio 40
  set traffic-class 3
!
 class MAIN_OUT_CMAP_4
  bandwidth remaining ratio 50
  set traffic-class 4
!
 class MAIN_OUT_CMAP_5
  bandwidth remaining ratio 60
  set traffic-class 5
!
 class MAIN_OUT_CMAP_6
  bandwidth remaining ratio 60
  set traffic-class 6
!
 class MAIN_OUT_CMAP_7
  set traffic-class 7
  priority level 1
 !
 class class-default
  bandwidth remaining ratio 10
 !
 end-policy-map
!
policy-map PARENT_MAIN_OUT_PHY_SUB
class class-default
  service-policy MAIN_OUT_PMAP5_PHY_SUB
  shape average 55 mbps
 !
 end-policy-map
```

### Verification

To check if interface is in **etm** mode:

```
Router#show controllers npu voq-usage interface all instance all l$
-------------------------------------------------------------------
Node ID: 0/0/CPU0
Intf Intf NPU NPU PP Sys VOQ Flow VOQ Port
name handle # core Port Port base base port speed
(hex) type
-----------------------------------------------------------------------
Hu0/0/0/0 8 0 0 1 24577 38400 6912 local 100G
Hu0/0/0/0 8 0 0 154 1 1536 22272 local 100G  ------>2nd Entry points to Recycle Port Mapping
```

To verify the output for policy map with class match on traffic class:

```
Router#show qos interface  HundredGigE 0/5/0/16.1 output
NOTE:- Configured values are displayed within parentheses
```

```
Interface HundredGigE0/5/0/16.1 ifh 0xa008002  -- output policy
NPU Id:                        1
Total number of classes:       9
Interface Bandwidth:           100000000 kbps
Policy Name:                   PARENT_MAIN_OUT_PHY_SUB
SPI Id:                        0x0
VOQ Base:                      13328
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   class-default
Queue Max. BW.                           =   70311 kbps (55 mbits/sec)
Queue Min. BW.                           =   58593 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)

   Level2 Class                             =   MAIN_OUT_CMAP_1
   New traffic class                        =   1
   Egressq Queue ID                         =   13329 (LP queue)
   Queue Max. BW.                           =   no max (default)
   Queue Min. BW.                           =   0 kbps (default)
   Inverse Weight / Weight                  =   32 / (20)
   Peak burst                               =   36864 bytes (default)
   TailDrop Threshold                       =   1280 bytes / 19 ms (default)
   WRED not configured for this class

   Level2 Class                             =   MAIN_OUT_CMAP_2
   New traffic class                        =   2
   Egressq Queue ID                         =   13330 (LP queue)
   Queue Max. BW.                           =   no max (default)
   Queue Min. BW.                           =   0 kbps (default)
   Inverse Weight / Weight                  =   21 / (30)
   Peak burst                               =   36864 bytes (default)
   TailDrop Threshold                       =   1280 bytes / 19 ms (default)
   WRED not configured for this class

   Level2 Class                             =   MAIN_OUT_CMAP_3
   New traffic class                        =   3
   Egressq Queue ID                         =   13331 (LP queue)
   Queue Max. BW.                           =   no max (default)
   Queue Min. BW.                           =   0 kbps (default)
   Inverse Weight / Weight                  =   16 / (40)
   Peak burst                               =   36864 bytes (default)
   TailDrop Threshold                       =   1280 bytes / 19 ms (default)
   WRED not configured for this class

   Level2 Class                             =   MAIN_OUT_CMAP_4
   New traffic class                        =   4
   Egressq Queue ID                         =   13332 (LP queue)
   Queue Max. BW.                           =   no max (default)
   Queue Min. BW.                           =   0 kbps (default)
   Inverse Weight / Weight                  =   12 / (50)
   Peak burst                               =   36864 bytes (default)
   TailDrop Threshold                       =   1280 bytes / 19 ms (default)
   WRED not configured for this class

   Level2 Class                             =   MAIN_OUT_CMAP_5
   New traffic class                        =   5
   Egressq Queue ID                         =   13333 (LP queue)
   Queue Max. BW.                           =   no max (default)
   Queue Min. BW.                           =   0 kbps (default)
   Inverse Weight / Weight                  =   10 / (60)
   Peak burst                               =   36864 bytes (default)
   TailDrop Threshold                       =   1280 bytes / 19 ms (default)
   WRED not configured for this class
```

```
Level2 Class                            =   MAIN_OUT_CMAP_6
New traffic class                       =   6
Egressq Queue ID                        =   13334 (LP queue)
Queue Max. BW.                          =   no max (default)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   10 / (60)
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   1280 bytes / 19 ms (default)
WRED not configured for this class

Level2 Class (HP1)                      =   MAIN_OUT_CMAP_7
New traffic class                       =   7
Egressq Queue ID                        =   13335 (HP1 queue)
Queue Max. BW.                          =   no max (default)
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   68608 bytes / 10 ms (default)
WRED not configured for this class

Level2 Class                            =   class-default
Egressq Queue ID                        =   13328 (Default LP queue)
Queue Max. BW.                          =   no max (default)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   64 / (10)
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   1280 bytes / 19 ms (default)
```

### Configuration Example for Policy Map with Class Match on DSCP

```
Router(config)#class-map match-any MATCH_DSCP_CONTROL
Router(config-cmap)#description "Customer Control Traffic"
Router(config-cmap)#match dscp 48-58
Router(config-cmap)#end-class-map
Router(config)#class-map match-any MATCH_DSCP_COS1_RT
Router(config-cmap)#description "Real Time Traffic"
Router(config-cmap)#match dscp 40 46
Router(config-cmap)#end-class-map
Router(config)#class-map match-any MATCH_DSCP_COS2_IC
Router(config-cmap)#description "Bursty High In Contract Traffic
Router(config-cmap)#match dscp 24 26
Router(config-cmap)#end-class-map
Router(config)#class-map match-any MATCH_DSCP_COS2_OC
Router(config-cmap)#description "Bursty High Out of Contract Traffic"
Router(config-cmap)#match dscp 25 27 28 29 30 31
Router(config-cmap)#end-class-map
Router(config)#class-map match-any MATCH_DSCP_COS3_IC
Router(config-cmap)#description "Bursty Low In Contract Traffic"
Router(config-cmap)#match dscp 16 18
Router(config-cmap)#end-class-map
Router(config)#class-map match-any MATCH_DSCP_COS3_OC
Router(config-cmap)#description "Bursty Low Out of Contract Traffic
Router(config-cmap)#match dscp 17 19 20 21 22 23
Router(config-cmap)#end-class-map
Router(config)#commit

Router(config)#policy-map M4_Ex_1G_Child_80:40:30:30_out_ss
Router(config-pmap)#class MATCH_DSCP_CONTROL
Router(config-pmap-c)#shape average percent 5
Router(config-pmap-c)#set traffic-class 7
Router(config-pmap-c)#priority level 2

Router(config-pmap-c)#class MATCH_DSCP_COS1_RT
Router(config-pmap-c)#shape average percent 80
Router(config-pmap-c)#set traffic-class 5
```

```
Router(config-pmap-c)#priority level 1
Router(config-pmap-c)#class MATCH_DSCP_COS2_IC
Router(config-pmap-c)#set traffic-class 2
Router(config-pmap-c)#bandwidth remaining percent 40
Router(config-pmap-c)#queue-limit 4096 packets
Router(config-pmap-c)#class MATCH_DSCP_COS3_IC
Router(config-pmap-c)#bandwidth remaining percent 30
Router(config-pmap-c)#set traffic-class 3
Router(config-pmap-c)#queue-limit 3072 packets
Router(config-pmap-c)#class class-default
Router(config-pmap-c)#bandwidth remaining percent 30
Router(config-pmap-c)#set traffic-class 0
Router(config-pmap-c)#queue-limit 3072 packets
Router(config-pmap-c)#end-policy-map
Router(config)#policy-map M4_Ex_1G_80:40:30:30_out_ss
Router(config-pmap)#class class-default
Router(config-pmap-c)#service-policy M4_Ex_1G_Child_80:40:30:30_out_ss
Router(config-pmap-c)#shape average 100 mbps
Router(config-pmap-c)#commit
Router(config-pmap-c)#exit
Router(config-pmap)#exit
Router(config)#interface HundredGigE 0/5/0/16.1
Router(config-subif)#service-policy output M4_Ex_1G_80:40:30:30_$
Router(config-subif)#commit
```

### Running Configuration

```
class-map match-any MATCH_DSCP_CONTROL
 description "Customer Control Traffic"
 match dscp 48-58
 end-class-map
!
class-map match-any MATCH_DSCP_COS1_RT
 description "Real Time Traffic"
 match dscp 40 46
 end-class-map
!
class-map match-any MATCH_DSCP_COS2_IC
 description "Bursty High In Contract Traffic"
 match dscp 24 26
 end-class-map
!
class-map match-any MATCH_DSCP_COS3_IC
 description "Bursty Low In Contract Traffic"
 match dscp 16 18
 end-class-map
!
policy-map M4_Ex_1G_Child_80:40:30:30_out_ss
 class MATCH_DSCP_CONTROL
  shape average percent 5
  set traffic-class 7
  priority level 2
 !
 class MATCH_DSCP_COS1_RT
  shape average percent 80
  set traffic-class 5
  priority level 1
 !
 class MATCH_DSCP_COS2_IC
  set traffic-class 2
  bandwidth remaining percent 40
  queue-limit 4096 packets
 !
```

```
 class MATCH_DSCP_COS3_IC
  bandwidth remaining percent 30
  set traffic-class 3
  queue-limit 3072 packets
 !
 class class-default
  bandwidth remaining percent 30
  set traffic-class 0
  queue-limit 3072 packets
 !
 end-policy-map
!
policy-map M4_Ex_1G_80:40:30:30_out_ss
 class class-default
  service-policy M4_Ex_1G_Child_80:40:30:30_out_ss
  shape average 100 mbps
 !
 end-policy-map
```

### Verify the Output for Policy Map with Class Match on DSCP

```
Router#show qos interface HundredGigE 0/5/0/16.1 output
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/5/0/16.1 ifh 0xa008002  -- output policy
NPU Id:                        1
Total number of classes:       6
Interface Bandwidth:           100000000 kbps
Policy Name:                   M4_Ex_1G_80:40:30:30_out_ss
SPI Id:                        0x0
VOQ Base:                      13328
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   class-default
Queue Max. BW.                           =   113282 kbps (100 mbits/sec)
Queue Min. BW.                           =   101564 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)

   Level2 Class (HP2)                       =   MATCH_DSCP_CONTROL
   New traffic class                        =   7
   Egressq Queue ID                         =   13335 (HP2 queue)
   Queue Max. BW.                           =   7812 kbps (5 %)
   Guaranteed service rate                  =   5000 kbps
   Peak burst                               =   36864 bytes (default)
   TailDrop Threshold                       =   6144 bytes / 10 ms (default)
   WRED not configured for this class

   Level2 Class (HP1)                       =   MATCH_DSCP_COS1_RT
   New traffic class                        =   5
   Egressq Queue ID                         =   13333 (HP1 queue)
   Queue Max. BW.                           =   82031 kbps (80 %)
   Guaranteed service rate                  =   80000 kbps
   Peak burst                               =   36864 bytes (default)
   TailDrop Threshold                       =   99840 bytes / 10 ms (default)
   WRED not configured for this class

   Level2 Class                             =   MATCH_DSCP_COS2_IC
   New traffic class                        =   2
   Egressq Queue ID                         =   13330 (LP queue)
   Queue Max. BW.                           =   no max (default)
   Queue Min. BW.                           =   0 kbps (default)
   Inverse Weight / Weight                  =   2 / (40%)
   Guaranteed service rate                  =   6000 kbps
   Peak burst                               =   36864 bytes (default)
   TailDrop Threshold                       =   1048576 bytes / 1398 ms (4096 packets)
   WRED not configured for this class
```

```
Level2 Class                            =   MATCH_DSCP_COS3_IC
New traffic class                       =   3
Egressq Queue ID                        =   13331 (LP queue)
Queue Max. BW.                          =   no max (default)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   2 / (30%)
Guaranteed service rate                 =   4500 kbps
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   786432 bytes / 1398 ms (3072 packets)
WRED not configured for this class

Level2 Class                            =   class-default
New traffic class                       =   0
Egressq Queue ID                        =   13328 (Default LP queue)
Queue Max. BW.                          =   no max (default)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   2 / (30%)
Guaranteed service rate                 =   4500 kbps
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   786432 bytes / 1398 ms (3072 packets)
WRED not configured for this class
```

### Configuration Example for Policy Map with Class Match on COS

```
Router(config)#class-map match-any cos2
 Router(config-cmap)#match cos 2
Router(config-cmap)#end-class-map
Router(config)#class-map match-all cos3
Router(config-cmap)#match cos 3
Router(config-cmap)#end-class-map
Router(config)#class-map match-all cos4
Router(config-cmap)#match cos 4
Router(config-cmap)#end-class-map
Router(config)#class-map match-all cos5
Router(config-cmap)#match cos 5
Router(config-cmap)#end-class-map
Router(config)#class-map match-all cos6
Router(config-cmap)#match cos 6
Router(config-cmap)#end-class-map
Router(config)#class-map match-all cos7
Router(config-cmap)#match cos 7
Router(config-cmap)#end-class-map
Router(config)#class-map match-all cos1
Router(config-cmap)#match cos 1
Router(config-cmap)#end-class-map
Router(config)#policy-map L2VPN-OUT
Router(config-pmap)#class cos2
Router(config-pmap-c)#set traffic-class 2
Router(config-pmap-c)#shape average 45 mbps
Router(config-pmap-c)#class cos3
Router(config-pmap-c)#set traffic-class 3
Router(config-pmap-c)#shape average 50 mbps
Router(config-pmap-c)#class cos4
Router(config-pmap-c)#set traffic-class 4
Router(config-pmap-c)#shape average 50 mbps
Router(config-pmap-c)#class cos5
Router(config-pmap-c)#set traffic-class 5
Router(config-pmap-c)#shape average 50 mbps
Router(config-pmap-c)#class cos6
Router(config-pmap-c)#set traffic-class 6
Router(config-pmap-c)#shape average 50 mbps
```

```
Router(config-pmap-c)#class cos7
Router(config-pmap-c)#set traffic-class 7
Router(config-pmap-c)#shape average 50 mbps
Router(config-pmap-c)#class cos1
Router(config-pmap-c)#set traffic-class 1
Router(config-pmap-c)#shape average 50 mbps
Router(config-pmap-c)#class class-default
Router(config-pmap-c)#shape average 50 mbps
Router(config-pmap-c)#end-policy-map
Router(config)#commit
Router(config)#exit

Router(config)#interface HundredGigE 0/5/0/16.1
Router(config-subif)#no service-policy output
Router(config-subif)#commit

Router(config-subif)#service-policy output L2VPN-OUT
Router(config-subif)#commit
Router(config-subif)#end
```

### Running Configuration

```
class-map match-any cos2
 match cos 2
 end-class-map
!
class-map match-all cos3
 match cos 3
 end-class-map
!
class-map match-all cos4
 match cos 4
 end-class-map
!
class-map match-all cos5
 match cos 5
 end-class-map
!
class-map match-all cos6
 match cos 6
 end-class-map
!
class-map match-all cos7
 match cos 7
 end-class-map
!
class-map match-all cos1
 match cos 1
 end-class-map
!
class-map match-any class-default
end-class-map
!
policy-map L2VPN-OUT
 class cos2
  set traffic-class 2
  shape average 45 mbps
 !
 class cos3
  set traffic-class 3
  shape average 50 mbps
 !
 class cos4
  set traffic-class 4
```

```
 shape average 50 mbps
!
class cos5
 set traffic-class 5
 shape average 50 mbps
!
class cos6
 set traffic-class 6
 shape average 50 mbps
!
class cos7
 set traffic-class 7
 shape average 50 mbps
!
class cos1
 set traffic-class 1
 shape average 50 mbps
!
class class-default
 shape average 50 mbps
!
end-policy-map
```

### Verify the Output for Policy Map with Class Match on COS

```
Router#show qos interface HundredGigE 0/5/0/16.1 output
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/5/0/16.1 ifh 0xa008002  -- output policy
NPU Id:                        1
Total number of classes:       8
Interface Bandwidth:           100000000 kbps
Policy Name:                   L2VPN-OUT
SPI Id:                        0x0
VOQ Base:                      13328
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   cos2
New traffic class                        =   2
Egressq Queue ID                         =   13330 (LP queue)
Queue Max. BW.                           =   46875 kbps (45 mbits/sec)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Guaranteed service rate                  =   45000 kbps
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   56064 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                             =   cos3
New traffic class                        =   3
Egressq Queue ID                         =   13331 (LP queue)
Queue Max. BW.                           =   50781 kbps (50 mbits/sec)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Guaranteed service rate                  =   50000 kbps
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   62464 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                             =   cos4
New traffic class                        =   4
Egressq Queue ID                         =   13332 (LP queue)
Queue Max. BW.                           =   50781 kbps (50 mbits/sec)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Guaranteed service rate                  =   50000 kbps
```

```
Peak burst                                =   36864 bytes (default)
TailDrop Threshold                        =   62464 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                              =   cos5
New traffic class                         =   5
Egressq Queue ID                          =   13333 (LP queue)
Queue Max. BW.                            =   50781 kbps (50 mbits/sec)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   50000 kbps
Peak burst                                =   36864 bytes (default)
TailDrop Threshold                        =   62464 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                              =   cos6
New traffic class                         =   6
Egressq Queue ID                          =   13334 (LP queue)
Queue Max. BW.                            =   50781 kbps (50 mbits/sec)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   50000 kbps
Peak burst                                =   36864 bytes (default)
TailDrop Threshold                        =   62464 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                              =   cos7
New traffic class                         =   7
Egressq Queue ID                          =   13335 (LP queue)
Queue Max. BW.                            =   50781 kbps (50 mbits/sec)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   50000 kbps
Peak burst                                =   36864 bytes (default)
TailDrop Threshold                        =   62464 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                              =   cos1
New traffic class                         =   1
Egressq Queue ID                          =   13329 (LP queue)
Queue Max. BW.                            =   50781 kbps (50 mbits/sec)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   50000 kbps
Peak burst                                =   36864 bytes (default)
TailDrop Threshold                        =   62464 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                              =   class-default
Egressq Queue ID                          =   13328 (Default LP queue)
Queue Max. BW.                            =   50781 kbps (50 mbits/sec)
Queue Min. BW.                            =   0 kbps (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
Guaranteed service rate                   =   50000 kbps
Peak burst                                =   36864 bytes (default)
TailDrop Threshold                        =   62464 bytes / 10 ms (default)
WRED not configured for this class
```

### Configuration Example for Policy Map with Class Match on Traffic Class and DSCP

```
Router(config)#class-map match-any cm-dscp-cs1
Router(config-cmap)#match dscp cs1
Router(config-cmap)#class-map match-any cm-dscp-cs2
```

```
Router(config-cmap)#match dscp cs2
Router(config-cmap)#class-map match-any cm-dscp-af11
Router(config-cmap)#match dscp af11
Router(config-cmap)#class-map match-any cm-tc-5
Router(config-cmap)#match traffic-class 5
Router(config-cmap)#class-map match-all cm-tc-6-and-dscp-cs3
Router(config-cmap)#match traffic-class 6
Router(config-cmap)#match dscp cs3
Router(config-cmap)#policy-map pm-out-etm-queueing
Router(config-pmap)#class cm-dscp-cs1
Router(config-pmap-c)#set traffic-class 1
Router(config-pmap-c)#shape average percent 45
Router(config-pmap-c)#class cm-dscp-cs2
Router(config-pmap-c)#set traffic-class 1
Router(config-pmap-c)#shape average percent 45
Router(config-pmap-c)#class cm-dscp-af11
Router(config-pmap-c)#set traffic-class 2
Router(config-pmap-c)#shape average percent 10
Router(config-pmap-c)#class cm-tc-5
Router(config-pmap-c)#shape average percent 5
Router(config-pmap-c)#set traffic-class 3
Router(config-pmap-c)#class cm-tc-6-and-dscp-cs3
Router(config-pmap-c)#set traffic-class 7
Router(config-pmap-c)#priority
Router(config-pmap-c)#commit
Router(config-pmap-c)#exit
Router(config-pmap)#exit
Router(config)#interface HundredGigE 0/5/0/16.1
Router(config-subif)#service-policy output pm-out-etm-queueing
Router(config-subif)#commit
Router(config-subif)#end
```

### Running Configuration

```
class-map match-any cm-dscp-cs1
match dscp cs1
 end-class-map
!
class-map match-any cm-dscp-cs2
match dscp cs2
 end-class-map
!
class-map match-any cm-dscp-af11
match dscp af11
 end-class-map
!
class-map match-any cm-tc-5
match traffic-class 5
 end-class-map
!
class-map match-all cm-tc-6-and-dscp-cs3
match traffic-class 6
 match dscp af22 cs3
 end-class-map
!
policy-map pm-out-etm-queueing
class cm-dscp-cs1
  set traffic-class 1
  shape average percent 45
 !
 class cm-dscp-cs2
  set traffic-class 1
  shape average percent 45
 !
```

```
 class cm-dscp-af11
  set traffic-class 2
  shape average percent 10
 !
 class cm-tc-5
  shape average percent 5
  set traffic-class 3
!
 class cm-tc-6-and-dscp-cs3
  set traffic-class 7
  priority level 1
 !
 class class-default
!
 end-policy-map
```

### Notes on the Configuration Example

- Both the traffic classes **cm-dscp-cs1** and **cm-dscp-cs2** have the same action for set traffic-class in the EM mode, which is **set traffic-class 1** . This means that both these classes have matching traffic going to the same VOQ, and hence their statistics refer to the statistics for VOQ=1.
- The traffic class **cm-dscp-af11** has the action set traffic-class 2 at the recycle pass in the ETM mode. Hence, its matching traffic and statistics are from VOQ=2.
- The traffic **class cm-tc-6-and-dscp-cs3** is an example of remarking traffic-class. This class has traffic-class=6 configured at the ingress, and matches all traffic that has incoming traffic-class=6 AND incoming DSCP=CS3. However, at egress, the traffic-class is remarked to 7. Hence, matching traffic goes to VOQ=7, and the statistics for this class are taken from VOQ=7.

### Verify the Output for Policy Map with Class Match on Traffic Class and DSCP

```
Router#show qos interface HundredGigE 0/5/0/16.1 output
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/5/0/16.1 ifh 0xa008002  -- output policy
NPU Id:                        1
Total number of classes:       6
Interface Bandwidth:           100000000 kbps
Policy Name:                   pm-out-etm-queueing
SPI Id:                        0x0
VOQ Base:                      13328
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   cm-dscp-cs1
New traffic class                        =   1
Egressq Queue ID                         =   13329 (LP queue)
Queue Max. BW.                           =   45003261 kbps (45 %)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   1249792 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                             =   cm-dscp-cs2
New traffic class                        =   1
Egressq Queue ID                         =   13329 (LP queue)
Queue Max. BW.                           =   45003261 kbps (45 %)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 / (BWR not configured)
Peak burst                               =   36864 bytes (default)
TailDrop Threshold                       =   1249792 bytes / 10 ms (default)
WRED not configured for this class
```

```
Level1 Class                            =   cm-dscp-af11
New traffic class                       =   2
Egressq Queue ID                        =   13330 (LP queue)
Queue Max. BW.                          =   10000152 kbps (10 %)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   1 / (BWR not configured)
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   1249792 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                            =   cm-tc-5
New traffic class                       =   3
Egressq Queue ID                        =   13331 (LP queue)
Queue Max. BW.                          =   5000076 kbps (5 %)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   1 / (BWR not configured)
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   1249792 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class (HP1)                      =   cm-tc-6-and-dscp-cs3
New traffic class                       =   7
Egressq Queue ID                        =   13335 (HP1 queue)
Queue Max. BW.                          =   no max (default)
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   124999936 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class                            =   class-default
Egressq Queue ID                        =   13328 (Default LP queue)
Queue Max. BW.                          =   no max (default)
Queue Min. BW.                          =   0 kbps (default)
Inverse Weight / Weight                 =   1 / (BWR not configured)
Peak burst                              =   36864 bytes (default)
TailDrop Threshold                      =   1249792 bytes / 10 ms (default)
WRED not configured for this class
```

## Associated Commands

### controller optics

# Low Rate Shaper

*Table 27: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Low Rate Shaper | Release 7.6.1 | This functionality allows your hardware to program low shaper bandwidth with accuracy. Earlier, if one of your customers wanted, say, a bandwidth of 500 kbps, the Cisco NC 57 line card ASIC could only program a minimum bandwidth of 3.9 Mbps for a particular traffic class, resulting in unnecessary wastage of bandwidth. With low shaper rates now made possible, you can limit bandwidth for specific subscribers whose requirements don't call for excessive bandwidth. This functionality thus allows you to save on your bandwidth costs.<br><br>This functionality is only available for and automatically enabled in the egress traffic management mode. |

- **What is the low rate shaper functionality all about?**—As the name suggests, the functionality is about providing accurate low shaper rate (less than 5 Mbps). But the powerful aspect of this feature is that the low shaper rates that it enables are far lesser what your existing Cisco NC 57 line card ASICs allow.

  This functionality is available only when you configure your egress interface in the Egress Traffic Management (Egress TM) mode. It's automatically enabled in this mode.

- **What are the benefits of this functionality?**—This functionality is supported on Cisco NC 57 line cards. The forwarding ASIC in this line card is capable of supporting only 3.9 Mbps as the lowest rate for a particular traffic class, even if you configure a shaping bandwidth less than 3.9 Mbps. When you have a typical network set up that has hundreds of edge-facing routers and thousands of interfaces, this restriction meant that excess bandwidth was distributed even to those subscribers who didn't require it, and hence didn't pay for it.

  With this functionality, you can save on your bandwidth costs by limiting the bandwidth for specific subscribers who require a lower spectrum of capacities. Hence, the low-rate shaper also provides you with more accuracy in shaping your hardware's bandwidth. The result is optimal bandwidth distribution and savings for your operations.

- **How does this functionality work? Any important points?**

  - When you configure a shaper rate of any value below 5 MBps for the virtual output queue (VOQ) of an ETM-enabled egress interface, the system automatically shifts into low-rate shaper mode.

- Once in low-rate shaper mode, your system supports traffic shaping rates to a particular VOQ in a specific range. This range is from 122 kbps through 80 Gbps.
- To support low-shaping capability, you need to reserve low-rate connectors. The system has 64 connector regions and each region contains 1024 connectors.
- Low connectors get allocated statically during system initialization, and the allocation can’t be changed dynamically. You can't use connectors that are reserved and configured as low connectors, as normal connectors, and the converse holds good as well.
- You have the flexibility to reserve the required connectors for low and normal regions before applying all QoS policies at egress.
- A connector switches from low to normal or the other way around if:
  - a low shaper rate (less than 5 Mbps) is detected on any of the traffic classes (VOQ) of an interface, where egress QoS policy map is applied. In this case, all the eight normal connectors are replaced by low connectors.
  - post-policy update or removal, no low shaper rate (less than 5 Mbps) is detected on an egress QoS Policy Map that is applied on VOQs attached with low connectors. In this case, low connectors are replaced by normal connectors.

- **How many egress policy maps can I configure with low rate shaper?**

  By default, 2 connector regions per core are reserved for low connectors.

  Therefore, per core, the number of low rate connectors are: number of low rate connectors per region x number of connector regions per core = 1024 x 2 = **2048**

  Each interface VOQ set requires **8** connectors (or 8 traffic classes).

  Therefore, the number of interface VOQ sets (or policy maps) that you can configure per core = number of low rate connectors/number of connectors per interface VOQ set = 2048/8 = **256**

- **Any limitations or restrictions?**
  - A low connector can't support more than 80 Gbps traffic shaping rate. Which means that a VOQ attached to a low connector can't forward more than 80 Gbps of traffic.
  - Attaching and detaching low connector sets to and from egress interfaces impacts traffic flows in unrelated VOQs. This could affect traffic flows when connectors are switched.
  - This functionality is applicable only for local switching on an ETM-enabled egress interface. This is why low-rate shaper benefits do not accrue to ingress traffic from a remote line card.

- **How do I verify the low shaper rate action?**—Run the **show qos interface** command to see the low shaper statistics. For example:

```
router#show qos interface GigabitEthernet 0/2/0/0 output
Level2 Class = exp1
New traffic class = 1
Egressq Queue ID = 1793 (LP queue)
Queue Max. BW. = 3051 kbps (3 mbits/sec)
Queue Min. BW. = 0 kbps (default)
Inverse Weight / Weight = 1 / (BWR not configured)
Guaranteed service rate = 625 kbps
Peak burst = 32832 bytes (default)
TailDrop Threshold = 1280 bytes / 16 ms (default)
```

```
LOW SHAPER = Enabled

WRED not configured for this class

Level2 Class = exp2
New traffic class = 2
Egressq Queue ID = 1794 (LP queue)
Queue Max. BW. = 5004 kbps (5 mbits/sec)
Queue Min. BW. = 0 kbps (default)
Inverse Weight / Weight = 1 / (BWR not configured)
Guaranteed service rate = 625 kbps
Peak burst = 32832 bytes (default)
TailDrop Threshold = 1280 bytes / 16 ms (default)

LOW SHAPER = Enabled

WRED not configured for this class
```