

Quality of Service Configuration Guide for Cisco NCS 4000 Series

First Published: 2018-07-23

Last Modified: 2026-03-23

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)

© 2021 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1 **QoS Overview** 1

- Information About Modular Quality of Service Overview 1
- QoS Techniques 1
- General QoS Terminology 2
- Modular QoS Command-Line Interface 2
- Traffic Class Elements 2
- Traffic Policy Elements 3
- Default Traffic Class 4
- Creating a Traffic Policy 4
  - Attaching a Traffic Policy to an Interface 5
- In-Place Policy Modification 6
  - Recommendations for Using In-Place Policy Modification 6

CHAPTER 2 **QoS Classification** 9

- Packet Classification Overview 9
- Specification of the CoS for a Packet with IP Precedence 10
  - IP Precedence Bits Used to Classify Packets 10
  - Classification based on DEI 11
- How to Configure Modular QoS Packet Classification 12
  - Creating a Classification Criterion 12
- QoS over Bundle Interfaces 14
  - Restrictions 15
  - Configuring QoS on a Bundle Interface 15
  - Configuring a Service Policy on a Bundle Interface 16

CHAPTER 3 **QoS Marking** 17

Marking Overview 17
Default Marking Behavior 18
IP Precedence Value Settings 18
TCP Establishment DSCP marking / Set IP Precedence 18
IP Precedence compared to IP DSCP marking 19
Conditional and Unconditional Marking 19
Understanding Unconditional Marking 19
Understanding Conditional Marking 21

CHAPTER 4 QoS Policing 25
Policing Overview 25
Regulation of Traffic with the Policing Mechanism 26
Single-Rate Policer 26
Two-Rate Policer 27
Committed Bursts and Excess Bursts 29
Deciding if Packets Conform or Exceed the Committed Rate 30
Two-Rate Three-Color (2R3C) Policer 31
Configuring Traffic Policing (Two-Rate Color-Blind) 31
Configuring Traffic Policing (2R3C) 33

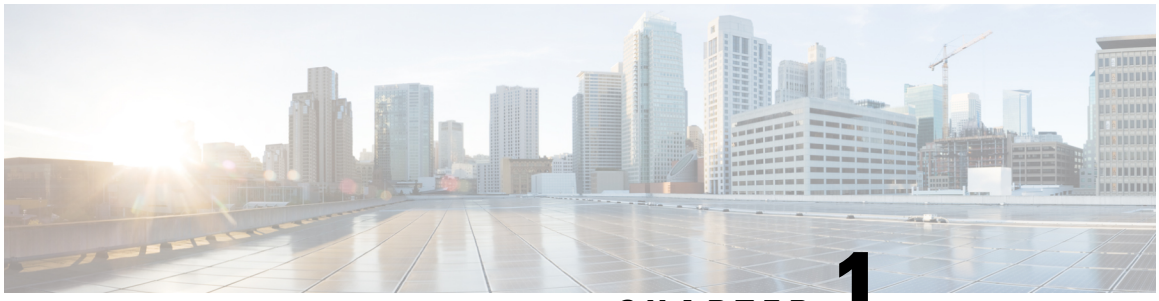
CHAPTER 5 Congestion Avoidance 37
Prerequisites for Configuring Modular QoS Congestion Avoidance 37
Random Early Detection and TCP 37
Queue-limit for WRED 38
Tail Drop 38
Configuring Random Early Detection 38
Configuring Weighted Random Early Detection 40
Configuring Tail Drop 42

CHAPTER 6 Congestion Management 47
Prerequisites for Configuring QoS Congestion Management 47
Information About Configuring Congestion Management 47
Congestion Management Overview 47
Low-Latency Queueing with Strict Priority Queueing 48

Traffic Shaping 48
Regulation of Traffic with the Shaping Mechanism 49
How to Configure QoS Congestion Management 49
Configuring Guaranteed and Remaining Bandwidths 49
Configuring Guaranteed Bandwidth 49
Configuring Bandwidth Remaining 52
Configuring Low-Latency Queueing with Strict Priority Queueing 55
How to Mitigate Control Packet Loss during Traffic Congestion at Core Interface 57
Configuring Traffic Shaping 58

CHAPTER 7 **Hierarchical QoS** 61
Information About Hierarchical QoS 61
Two-Level Hierarchical Policies 61
Configuring Hierarchical Policing 61

CHAPTER 8 **Dual Policy** 67
Dual Policy 67



CHAPTER 1

QoS Overview

Quality of Service (QoS) is the technique of prioritizing traffic flows and providing preferential forwarding for higher-priority packets. The fundamental reason for implementing QoS in your network is to provide better service for certain traffic flows. A traffic flow can be defined as a combination of source and destination addresses, source and destination socket numbers, and the session identifier. A traffic flow can more broadly be described as a packet moving from an incoming interface that is destined for transmission to an outgoing interface. The traffic flow must be classified, and prioritized on all routers and passed along the data forwarding path throughout the network to achieve end-to-end QoS delivery. The terms *traffic flow* and *packet* are used interchangeably throughout this module.

This module contains overview information about modular QoS features within a service provider network.

- Information About Modular Quality of Service Overview, on page 1
- QoS Techniques, on page 1
- General QoS Terminology , on page 2
- Modular QoS Command-Line Interface, on page 2
- Traffic Class Elements, on page 2
- Traffic Policy Elements, on page 3
- Default Traffic Class, on page 4
- Creating a Traffic Policy, on page 4
- In-Place Policy Modification, on page 6

Information About Modular Quality of Service Overview

Before configuring modular QoS on your network, you must understand these concepts:

QoS Techniques

QoS on Cisco IOS XR relies on these techniques to provide end-to-end QoS delivery across a heterogeneous network:

- QoS classification - classification techniques identify the traffic flow, and provide the capability to partition network traffic into multiple priority levels or classes of service. Identification of a traffic flow can be performed by using several methods within a single router, such as IP precedence, IP differentiated service code point (DSCP), MPLS EXP bit, or Class of Service (CoS).

- Qos Marking - after classification, the traffic is marked to indicate the required level of QoS for that traffic. Packets marked as priority are met with preferential treatment.
- QoS Policing - allows the user 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).
- Queueing - is a congestion management technique. Cisco NCS 4000 supports default queue selection for layer2 and layer3. Use the set traffic-class command (ingress side) to explicitly choose a queue and override the default queue selection.
- H-QoS - allows the user to specify QoS behavior at multiple policy levels, which provides a high degree of granularity in traffic management.
- Dual Policy - enables support for two output policies.
- Congestion avoidance - includes techniques that monitor network traffic flows, in an effort to anticipate and avoid congestion at common network and internetwork bottlenecks before problems occur.

Before implementing the QoS features for these techniques, you should identify and evaluate the traffic characteristics of your network because not all techniques are appropriate for your network environment.

General QoS Terminology

This section provides a summary of the frequently used QoS terminologies.

- Dropping technologies (Tail Drop and WRED) - Tail drop is a congestion avoidance technique that drops packets when an output queue is full until congestion is eliminated. WRED drops packets selectively based on IP precedence.
- Shapers and policers - are needed to ensure that a packet adheres to a contract and service. A policer typically drops traffic flow, when the traffic flow exceeds the policer rate. A shaper delays excess traffic flow using a buffer, or queuing mechanism, to hold the traffic for transmission at a later time.
- DEI - incase of congestion, a packet marked with DEI (Drop Eligible Indicator) is dropped.
- DSCP - the DSCP (Differentiated Services Code Point) bit in the IP header is used for packet classification.

Modular QoS Command-Line Interface

QoS features are enabled through the Modular QoS command-line interface (MQC) feature. The *MQC* is a command-line interface (CLI) structure that allows you to create policies and attach these policies to interfaces. A traffic policy contains a traffic class and one or more QoS features. A traffic class is used to classify traffic, whereas the QoS features in the traffic policy determine how to treat the classified traffic. One of the main goals of MQC is to provide a platform-independent interface for configuring QoS across Cisco platforms.

Traffic Class Elements

The purpose of a traffic class is for queuing and classification of traffic on the 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, and, if more than one **match** command exists in the traffic class, an instruction on how to evaluate these **match** commands. The traffic class is named in the **class-map** command. For example, if you use the word *cisco* with the **class-map** command, the traffic class would be named *cisco*.

Note The **class-map** command supports the **match-any** keyword only. The **match-all** keyword is not supported.

The **match** commands are used to specify various criteria for classifying packets. Packets are checked to determine whether they match the criteria 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 the match types supported on the Cisco 4000 Series router.

Supported Match Type	Min, Max	Max Entries	Support for Ranges	Direction supported for interfaces
IPv4 DSCP DSCP	(0, 63)	64	yes	Ingress
IPv4 Precedence Precedence	(0,7)	8	no	Ingress
MPLS Experimental Topmost	(0,7)	8	no	Ingress
QoS-group	(1,7)	7	no	Egress

Traffic Policy Elements

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. The **policy-map** command is used to create a traffic policy. A traffic policy contains three elements: a name, a traffic class (specified with the **class** command), and the QoS policies. The name of a traffic policy is specified in the policy map MQC (for example, the **policy-map** *policy1* command creates a traffic policy named *policy1*). The traffic class that is used to classify traffic to the specified traffic policy is defined in class map configuration mode. After choosing the traffic class that is used to classify traffic to the traffic policy, the user can enter the QoS features to apply to the classified traffic. The **class-map** command is used to define a traffic class and the associated rules that match packets to the class.

The MQC does not necessarily require that users associate only one traffic class to one traffic policy. When packets match to more than one match criterion, as many as 8 traffic classes can be associated to a single traffic policy. The 8 class maps include the default class and the classes of the child policies, if any.

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.

Creating a Traffic Policy

To create a traffic policy (supported on egress), use the **policy-map** command to specify the traffic policy name.

The traffic class is associated with the traffic policy when the **class** command is used. The **class** command must be issued after you enter the policy map configuration mode. After entering the **class** command, the router is automatically in policy map class configuration mode, which is where the QoS policies for the traffic policy are defined.

These class-actions are supported:

- bandwidth—Configures the bandwidth for the class.
- bandwidth remaining ratio—Configures the bandwidth remaining ratio for the class.
- police—Police traffic.
- priority—Assigns priority to the class.
- queue-limit—Configures queue-limit (tail drop threshold) for the class.
- random-detect—Enables Random Early Detection.
- service-policy—Configures a child service policy.
- set—Configures marking for this class.
- shape—Configures shaping for the class.

Restrictions

A maximum of 8 classes for Level 1 and 1 for Level 2.

Procedure

Step 1 **configure**

Step 2 **policy-map** [**type qos**] *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3 **class** class-name

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change.

Step 4 **commit**

Running configuration example for Creating a traffic policy

```
policy-map ingress_POLICER_POLICY
class CLASS_1_POLICERIPV4PREC
  set traffic-class 7
!
```

Attaching a Traffic Policy to an Interface

After the traffic class and traffic policy are created, you must use the service-policy interface configuration command to attach a traffic policy to an interface, and to specify the direction in which the policy should be applied (either on packets coming into the interface or packets leaving the interface).

Prerequisites

A traffic class and traffic policy must be created before attaching a traffic policy to an interface.

Procedure

Step 1 **configure**

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface HundredGigE 0/7/0/1
```

Configures an interface and enters the interface configuration mode.

Step 3 **service-policy** {**input** | **output**} *policy-map*

Example:

```
RP/0/(config-if)# service-policy output policy1
```

Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

Step 4 **commit**

Step 5 **show policy-map interface** *type interface-path-id* [**input** | **output**]

Example:

```
RP/0/# show policy-map interface HundredGigE 0/7/0/1
```

(Optional) Displays statistics for the policy on the specified interface.

Running configuration example for attaching a traffic policy to an interfac e

```
interface TenGigE0/5/0/1/3.100
service-policy input ingress_POLICER_POLICY
ipv4 address 10.0.0.1 255.255.255.0
encapsulation dot1q 100
!
```

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.

Verification

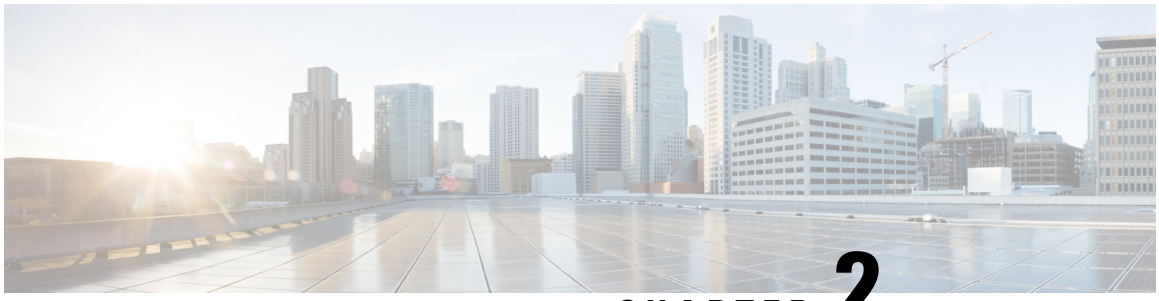
If unrecoverable errors occur during in-place policy modification, the policy is put into an inconsistent state on target interfaces. No new configuration is possible until the configuration session is unblocked. It is recommended to remove the policy from the interface, check the modified policy and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

Recommendations for Using In-Place Policy Modification

For a short period of time while a QoS policy is being modified, no QoS policy is active on the interface. In the unlikely event that the QoS policy modification and rollback both fail, the interface is left without a QoS policy.

For these reasons, it is best to modify QoS policies that affect the fewest number of interfaces at a time. Use the **show policy-map targets** command to identify the number of interfaces that will be affected during policy map modification.

For a short period of time while a QoS policy is being modified, there might not be any policy in effect on the interfaces in which the modified policy is used. For this reason, modify QoS policies that affect the fewest number of interfaces at a time. Use the **show policy-map targets** command to identify the number of interfaces that will be affected during policy map modification.



CHAPTER 2

QoS Classification

QoS classification identifies and marks traffic flows that require congestion management or congestion avoidance on a data path. The Modular Quality of Service (QoS) command-line interface (MQC) is used to define the traffic flows that should be classified, where each traffic flow is called a class of service, or class. Subsequently, a traffic policy is created and applied to a class. All traffic not identified by defined classes falls into the category of a default class.

This chapter provides the conceptual and configuration information for QoS packet classification.

- Packet Classification Overview, on page 9
- Specification of the CoS for a Packet with IP Precedence, on page 10
- How to Configure Modular QoS Packet Classification, on page 12
- QoS over Bundle Interfaces , on page 14

Packet Classification Overview

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. For example, by using the three precedence bits in the type of service (ToS) field of the IP packet header, you can categorize packets into a limited set of up to eight traffic classes. 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 maximum transmission unit (MTU) for interfaces (both bundle and physical) is 9644, that is, interfaces can accept packets with a maximum of 9644 bytes.

Ingress classification techniques:

- match cos
- match dscp
- match ip precedence

- match exponential
- match dei

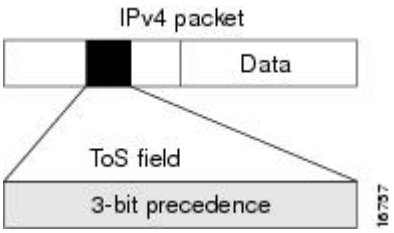
Egress classification techniques:

- match traffic-class
- match discard-class
- match qos-group

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 1: 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. As mentioned earlier, 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.

For historical reasons, each precedence corresponds to a name. These names are defined in RFC 791.

This table lists the numbers and their corresponding names, from least to most important.

Table 1: IP Precedence Values

Number	Name
0	routine

Number	Name
1	priority
2	immediate
3	flash
4	flash-override
5	critical
6	internet
7	network

The IP precedence feature allows you considerable flexibility for precedence assignment. That is, you can define your own classification mechanism. For example, you might want to assign precedence based on application or access router.



Note IP precedence bit settings 6 and 7 are reserved for network control information, such as routing updates.

Classification based on DEI

You can classify traffic based on the Drop Eligible Indicator (DEI) bit that is present in 802.1ad frames and in 802.1ah frames. Default DEI marking is supported. The set DEI action in policy maps is supported on 802.1ad packets for:

- Ingress
- Layer 2 sub-interfaces

Default DEI marking

Incoming packet	Default DEI on imposed 802.1ad headers
802.1 q packet	0
802.1 ad packet	DEI of the topmost tag of the incoming packet
802.1q packet translated to 802.1ad packet or 802.1ad packet	0 or 1 ; based on the DEI value of the set action

How to Configure Modular QoS Packet Classification

Creating a Classification Criterion

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



Note 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.

Restrictions

- All **match** commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.

Procedure

Step 1 **configure**

Step 2 **class-map** [**type qos**] [**match-any**] *class-map-name*

Example:

```
RP/0/(config)# class-map class201
```

Creates a class map to be used for matching packets to the class whose name you specify and enters the class map configuration mode.

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.

Step 3 **match cos** [*cos-value*] [*class-name*]

Example:

```
RP/0/(config-cmap)# match cos 5
```

(Optional) Specifies a *cos-value* in a class map to match packets. The *cos-value* arguments are specified as an integer from 0 to 7.

Step 4 **match traffic-class** *class-name*

Example:

```
RP/0/(config-cmap)# match traffic-class c1
```

(Optional) Specifies a *traffic class name*in a class map to match packets on an egress queuing policy. The arguments are specified as an integer from 0 to 7.

Step 5 **match dscp** [**ipv4** *dscp-value* [*dscp-value ... dscp-value*]

Example:

```
RP/0/(config-cmap)# match dscp ipv4 15
```

(Optional) Identifies a specific DSCP value as a match criterion.

- Value range is from 0 to 63.
- Reserved keywords can be specified instead of numeric values.
- Up to eight values or ranges con be used per match statement.

Step 6 **match mpls experimental topmost** *exp-value* [*exp-value1 ... exp-value7*]

Example:

```
RP/0/(config-cmap)# match mpls experimental topmost 3
```

(Optional) Configures a class map so that the three-bit experimental field in the topmost Multiprotocol Label Switching (MPLS) labels are examined for experimental (EXP) field values. The value range is from 0 to 7.

Step 7 **match mpls experimental imposition** *value*

Example:

```
RP/0/(config-cmap)# match mpls experimental imposition 5
```

(Optional) Configures a class map for the three-bit experimental field in the topmost Multiprotocol Label Switching (MPLS) imposition labels. The value range is from 0 to 7.

Step 8 **match precedence** [**ipv4**] *precedence-value* [*precedence-value1 ... precedence-value6*]

Example:

```
RP/0/(config-cmap)# match precedence ipv4 5
```

(Optional) Identifies IP precedence values as match criteria.

- Value range is from 0 to 7.
- Reserved keywords can be specified instead of numeric values.

Step 9 **match qos-group** [*qos-group-value1 ... qos-group-value8*]

Example:

```
RP/0/(config-cmap)# match qos-group 0 1 2 3 4 5 6 7
```

(Optional) Specifies service (QoS) group values in a class map to match packets.

- *qos-group-value* identifier argument is specified as the exact value or range of values from 0 to 7.
- Up to eight values (separated by spaces) can be entered in one match statement.
- **match qos-group** command is supported only for an egress marking policy.

Step 10 **commit**

Running configuration example for traffic class

```
class-map match-any CLASS_1_POLICERIPV4PREC
match precedence 1
end-class-map

class-map match-any class2
match qos-group 7
end-class-map
!
```

QoS over Bundle Interfaces

Table 2: Feature History

Feature Name	Release Information	Feature Description
Increase to 250 Policy maps	Cisco IOS XR Release 6.5.31	The Cisco NCS 4000 supports up to 250 unique QoS policies distributed across 512 physical sub interfaces by default.

The Cisco NCS 4000 supports up to 250 unique policies distributed across 512 physical sub interfaces by default. While configuring QoS policies on bundle interfaces, the policer id is replicated on both the cores.

Scale Mode

The user can choose bundle or non-bundle scale requirements based on the three profiles. The following table displays the number of supported sub-interfaces, with QoS ingress policy, for physical and bundle interfaces.

Table 3: Supported Bundles based on Scale Mode

	Physical		LAG	
	Core	Card per Network Processor Unit	Core	Card per Network Processor Unit
Default	512	1024	0	0
High QoS LAG	64	128	448	448
Medium QoS LAG	256	512	256	256
Low QoS LAG	448	896	64	64

Restrictions

Restrictions for QoS on bundle interfaces:

- All ingress QoS configurations on a line card must be removed before the scale profile can be added or modified or deleted. If this is not done, the system can go in to an unpredictable and inconsistent state, and any such configuration attempt is not supported. To recover, if the currently configured ingress QoS scale is within the scale allowed by the newly configured scale mode, a LC hw-module reset can be done to recover the system.
- For bundle sub-interfaces whose parent interfaces are across multiple line cards, the maximum possible scale is limited by the least applicable scale-mode. If a bundle and its sub-interfaces have one or more members going over a line card which does not have a valid scale-mode configured, then, ingress QoS configuration applied to it will be denied.

Configuring QoS on a Bundle Interface

The following configuration procedure enables the user to configure the qos policy over a bundle with a specific scale mode.

Procedure

Step 1 **configure**

Step 2 **hw-module profile qos bundle** [*scale-mode*] **location** *location-id*

Example:

```
RP/0/RP0:router(config)#hw-module profile qos bundle high-scale location 0/13
```

Enables a bundle on the router with the specified scale mode. The available options for scale mode are high-scale, medium-scale and low-scale. The number of supported sub-interfaces is based on the configured scale mode.

Note
The **hw-module profile qos bundle** command is service-affecting. The following warning is displayed - *QoS profile CLI on any Line Card should be used only when no ingress QoS is configured on the same Line Card. Failure to do so can result in functionality breakage and system wide inconsistencies. In case of any misconfiguration, the scale mode config has to be reverted followed by LC hw-module reset to get the system back into original state.*

Step 3 **hw-module location** *location-id* **reload**

Example:

```
RP/0/RP0:router(config)#hw-module location 0/1 reload
```

Reloads the line card.

Step 4 **commit**

Configuring a Service Policy on a Bundle Interface

The following configuration procedure enables the user to attach a service policy to a bundle interface.

Procedure

Step 1 **config**

Step 2 **interface** *type interface-id* **l2transport**

Example:

```
RP/0/RP0:router(config)#interface bundle-ether 2000.1 l2 transport
```

Enters the interface configuration mode.

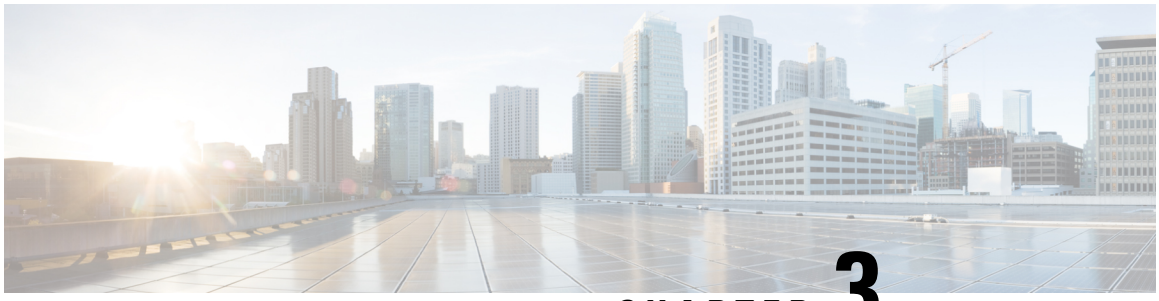
Step 3 **service-policy** [**input** | **output**] *policy-name*

Example:

```
RP/0/RP0:router(config-subif)#service-policy input policy-1
```

Attaches a service policy to the bundle interface.

Step 4 **commit**



CHAPTER 3

QoS Marking

Marking is used to indicate the priority in traffic. The packets are marked to indicate higher priority traffic over lower priority traffic.

This chapter provides conceptual and configuration details for the QoS marking.

- Marking Overview, on page 17
- IP Precedence Value Settings, on page 18
- TCP Establishment DSCP marking / Set IP Precedence , on page 18
- IP Precedence compared to IP DSCP marking , on page 19
- Conditional and Unconditional Marking, on page 19

Marking Overview

Marking is a process, which helps to modify the quality of service for incoming and outgoing packets. You can use marking commands in traffic classes, which are referenced in the policy map.

Ingress marking techniques:

- set cos
- set dei
- set precedence
- set dscp
- set exponential
- set traffic-class
- set qos-group
- set discard-class
- unconditional marking

Egress marking techniques:

- set cos
- set dei

- conditional marking

Default Marking Behavior

When an ingress or egress interface adds VLAN tags or MPLS labels, it requires a default value for the CoS and EXP values that go into those tags and labels. The default value can be then overridden based on the policy map. The default value for CoS and EXP is based on a trusted field in the packet upon ingress to the system. The router implements an implicit trust of certain fields based on the packet type and ingress interface forwarding type (Layer 2 or Layer 3).

By default, the router does not modify the IP precedence or DSCP without a policy-map being configured. The default behavior is described below.

On an ingress or egress Layer 2 interface, such as xconnect, the outermost CoS value is used for any field that gets added in the ingress interface. If there is a VLAN tag that gets added due to a Layer 2 rewrite, the incoming outermost CoS value is used for the new VLAN tag. If an MPLS label is added, the CoS value would be used for the EXP bits in the MPLS tag.

On an ingress or egress Layer 3 interface (routed or label weighted for IPv4 packets), the three DSCP and precedence bits are identified in the incoming packet. For MPLS packets, the outermost label's EXP bit is identified, and this value is used for any new field that gets added at the ingress interface. If an MPLS label is added, then the identified precedence, DSCP, or MPLS EXP value is used for the EXP bits in the newly added MPLS tag.

IP Precedence Value Settings

By default, Cisco IOS XR software leaves 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.

TCP Establishment DSCP marking / Set IP Precedence

The Differentiated Services Code Point (DSCP) field in an IP packet which helps enables different levels of service to be assigned to network traffic. Marking is a process, which helps to modify QOS fields incoming and outgoing packets. You can use marking commands in traffic classes, which are referenced in the policy map.

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 and Unconditional Marking

The class-based, Unconditional Packet Marking feature provides users with a means for efficient packet marking by which the users can differentiate packets based on the designated markings. The unconditional marking feature allows users to:

- Mark packets by setting the layer-2 Class-Of-Service (COS) value.
- Mark packets by setting the Drop Eligible Indicator (DEI) value based on the policy map.
- Mark packets by setting the IP precedence bits or the IP differentiated services code point (DSCP) in the IP ToS or DSCP field.
- Mark Multiprotocol Label Switching (MPLS) packets by setting the EXP bits within the imposed or topmost label.

Use QoS unconditional packet marking to assign packets to a QoS group. To set the QoS group identifier on MPLS packets, use the **set qos-group** command in policy map.

Packet marking as a policer action is conditional marking.

Understanding Unconditional Marking

Unconditional marking on Xconnect

Support Matrix for L2 Xconnect

QoS on Xconnect	COS	DEI
COS	Supported	Supported
DEI	Supported	Supported

Provided below, is the work flow to configure Unconditional Marking on Xconnect. A simple topology consisting of a router (NCS 4000) is considered, with ingress traffic.

1. Configure LANPHY mode

```
controller Optics0/5/0/6
 Port-mode ethernet framing packet rate 10GIG
controller Optics0/3/0/3
 Port-mode ethernet framing packet rate 10GIG
```

2. Configure L2VPN Xconnect

```
interface TenGigE0/5/0/6
no shutdown
 l2transport
interface TenGigE0/3/0/3
 no shutdown
 l2transport
l2vpn
 xconnect group test
  p2p test
   interface TenGigE0/5/0/6
   interface TenGigE0/3/0/3
```

3. Configure Class Map

```
class-map match-any COS7
match cos 7
end-class-map
```

4. Configure Policy Map

```
policy-map ingress_marking
class COS7
police rate 2 gbps
!
set cos 2
set dei 1
!
class class-default
!
end-policy-map
!
```

5. Apply policy-map on an interface

```
interface TenGigE0/5/0/6
l2transport
service-policy input ingress_marking
```

Unconditional Marking on VPWS

Support Matrix for VPWS

QoS on VPWS	COS	DEI	EXP	PRECEDENCE	DSCP
COS	Not supported	Not supported	Supported	Not supported	Not supported
DEI	Not supported	Not supported	Supported	Not supported	Not supported
EXP	Not supported	Not supported	Not supported	Not supported	Not supported
PRECEDENCE	Not supported	Not supported	Not supported	Not supported	Not supported
DSCP	Not supported	Not supported	Not supported	Not supported	Not supported

Provided below is the workflow to configure unconditional marking on VPWS. The topology consists of two NCS 4000 series routers (R1 and R2), with standard ingress and egress traffic, connected to each other by a

Flex LSP tunnel. From R1 to R2, COS-1 is marked to MPLS EXP 7. From R2 to R1, MPLS EXP 7 is marked to COS-5.

- Configuration on R1

```
class-map match-any COS1
match cos 1
end-class-map
!

policy-map POLICY1
class COS1
  set traffic-class 1
  set mpls experimental imposition 7
  police rate 10 mbps peak-rate 20 mbps


interface HundredGigE0/6/0/5.100 l2transport
encapsulation dot1q 100
service-policy input POLICY1
```

- Configuration on R2

```
class-map match-any mpls_experimental_topmost_7
match mpls experimental topmost 7

policy-map POLICY2
class mpls_experimental_topmost_7
  set traffic-class 7
  set cos 5


interface HundredGigE0/2/0/1
service-policy input POLICY2
ipv4 address 1.76.1.2 255.255.255.0
```

Understanding Conditional Marking

Conditional Marking on VPWS

Support Matrix for VPWS

QoS on VPWS	COS	DEI	EXP	PRECEDENCE	DSCP
COS	Not supported	Not supported	Supported	Not supported	Not supported
DEI	Not supported	Not supported	Supported	Not supported	Not supported
EXP	Not supported	Not supported	Not supported	Not supported	Not supported
PRECEDENCE	Not supported	Not supported	Supported	Not supported	Not supported
DSCP	Not supported	Not supported	Supported	Not supported	Not supported

Provided below is the workflow to configure conditional marking on VPWS. The topology consists of two NCS 4000 series routers (R1 and R2), with standard ingress and egress traffic, connected to each other by a Flex LSP tunnel. From R1 to R2, COS 1 is marked to EXP 7 for CIR and EXP 1 for PIR. From R2 to R1, EXP 7 is marked to COS 7.

- Configuration on R1

```
class-map match-any COS1
match cos 1
end-class-map


policy-map POLICY1
class COS1
  set qos-group 1
  police rate 10 mbps peak-rate 20 mbps


interface HundredGigE0/6/0/5.100 l2transport
encapsulation dot1q 100
service-policy input POLICY1


class-map match-all qos-group-1-discard-class-0
match qos-group 1
match discard-class 0
end-class-map
!
class-map match-all qos-group-1-discard-class-1
match qos-group 1
match discard-class 2
end-class-map
!

policy-map POLICY-CONDITIONAL-EGRESS-1
class qos-group-1-discard-class-0
  set mpls experimental imposition 7

class qos-group-1-discard-class-1
  set mpls experimental imposition 1
!

interface HundredGigE0/3/0/0
service-policy output POLICY-CONDITIONAL-EGRESS-1
ipv4 address 1.76.1.1 255.255.255.0
!
```

- Configuration on R2

```
class-map match-any mpls_experimental_topmost_7
match mpls experimental topmost 7
end-class-map
!

policy-map POLICY2
class mpls_experimental_topmost_7
  set qos-group 7
  set discard-class 0
!

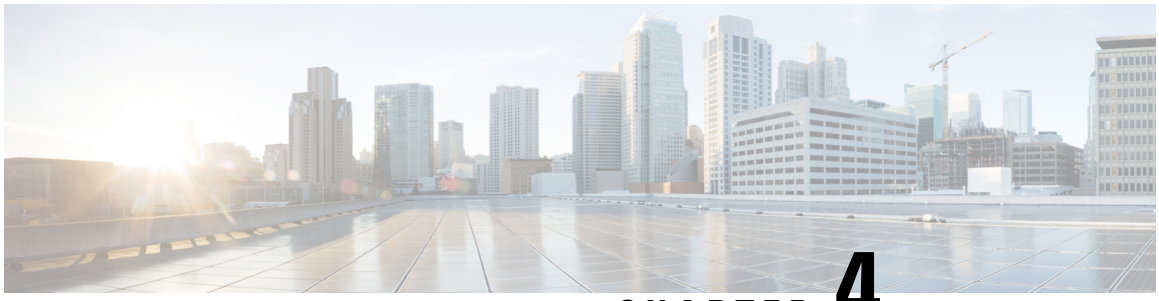
interface HundredGigE0/2/0/1
service-policy input POLICY2
ipv4 address 1.76.1.2 255.255.255.0
!

class-map match-all qos-group-1-discard-class-1
match qos-group 1
match discard-class 1
```

```
end-class-map
!

policy-map POLICY-CONDITIONAL-EGRESS-2
class qos-group-1-discard-class-1
  set cos 5
!

interface TenGigE0/2/0/7/1.100 l2transport
encapsulation dot1q 100
service-policy output POLICY-CONDITIONAL-EGRESS-2
!
```



CHAPTER 4

QoS 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).

This chapter provides conceptual and configuration details for QoS Traffic Policing.

- Policing Overview, on page 25
- Configuring Traffic Policing (Two-Rate Color-Blind), on page 31
- Configuring Traffic Policing (2R3C), on page 33

Policing Overview

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.

Traffic entering the interface with traffic policing configured is placed into one of these categories. Within these three categories, users can decide packet treatments. For instance, packets that conform can be configured to be sent, packets that exceed can be configured to be sent with a decreased priority, and packets that violate can be configured to be dropped.

Traffic policing is often configured on interfaces at the edge of a network to limit the rate of traffic entering or leaving the network. In the most common traffic policing configurations, traffic that conforms to the CIR is sent and traffic that exceeds is sent with a decreased priority or is dropped. Users can change these configuration options to suit their network needs.



Note Configured values take into account the Layer 1 encapsulation applied to traffic. This applies to both ingress and egress policing. For Ethernet, the encapsulation is 34 bytes; whereas for 802.1Q, the encapsulation is 38 bytes.

Traffic policing is often configured on interfaces at the edge of a network to limit the rate of traffic entering or leaving the network. In the most common traffic policing configurations, traffic that conforms to the CIR is sent and traffic that exceeds is sent with a decreased priority or is dropped. Users can change these configuration options to suit their network needs. 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). When the

peak information rate (PIR) is supported, a second token bucket is enforced and then the traffic policer is called a two-rate policer.

The supported policing features are:

- Single-rate policers
- Two-rate policers

Regulation of Traffic with the Policing Mechanism

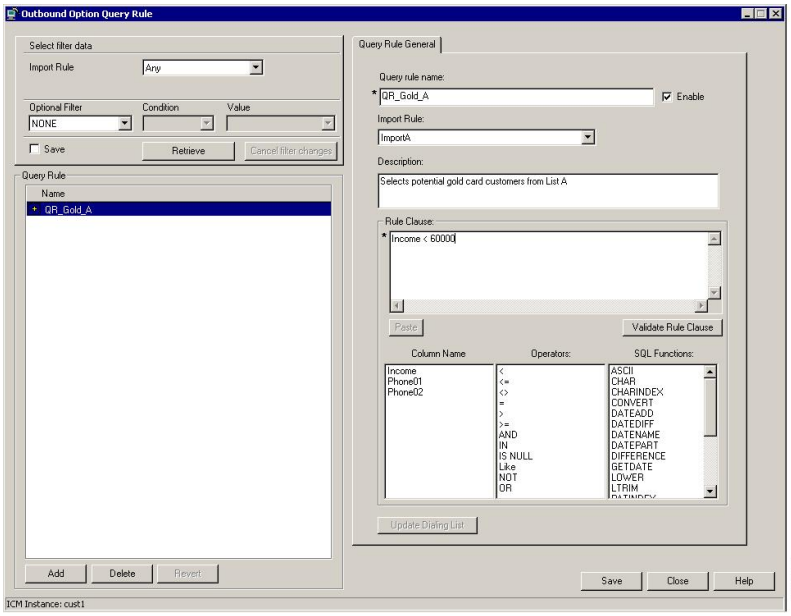
This section describes the single-rate and two-rate policing mechanisms.

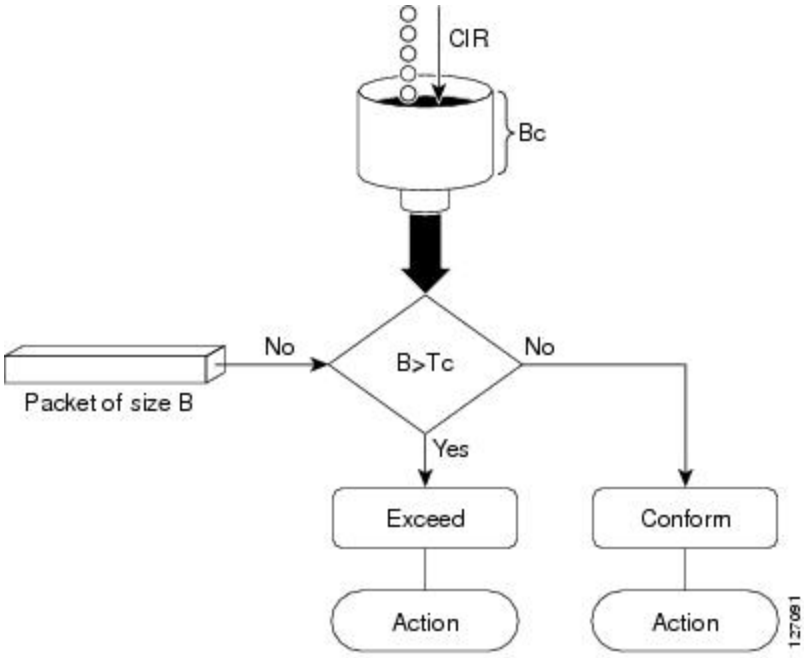
Single-Rate Policer

A single-rate, two-action policer provides one token bucket with two actions for each packet: a conform action and an exceed action.

This figure illustrates how a single-rate token bucket policer marks packets as either conforming or exceeding a CIR, and assigns an action.

Figure 2: Marking Packets and Assigning Actions—Single-Rate Policer





The time interval between token updates (Tc) to the token bucket is updated at the CIR value each time a packet arrives at the traffic policer. 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 configured action is performed. If a packet of size B is less than the Tc token bucket, then the packet conforms and a different configured action is performed.

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 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:

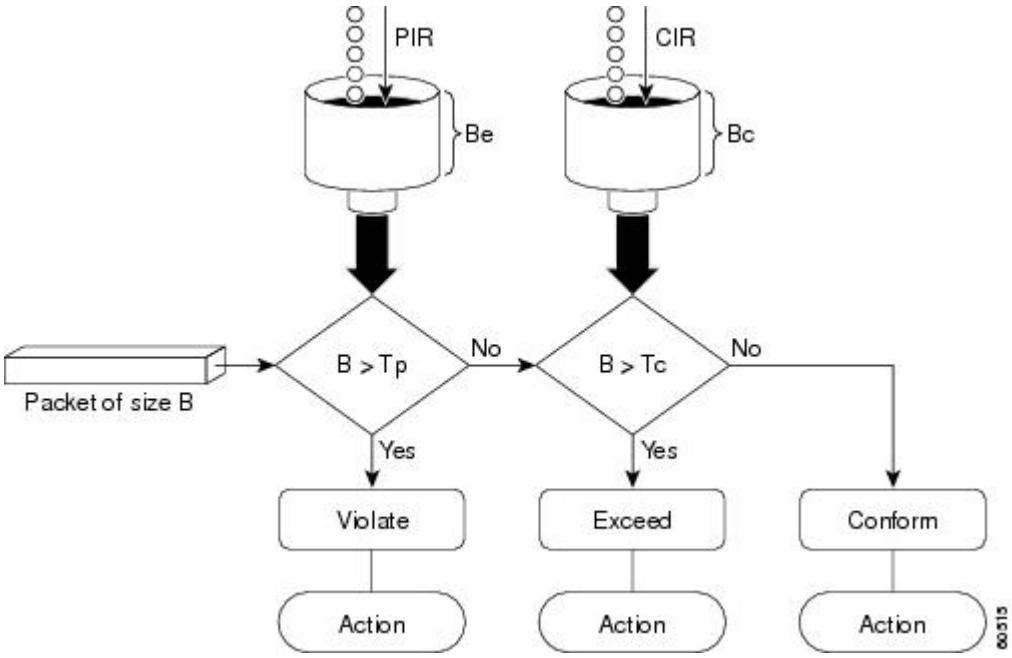
- A traffic stream is conforming when the average number of bytes over time does not cause the committed token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream green.
- A traffic stream is exceeding when it causes the committed token bucket to overflow into the 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.

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. Within these three categories, users can decide packet treatments. For

instance, packets that conform can be configured to be sent; packets that exceed can be configured to be sent with a decreased priority; and packets that violate can be configured to be dropped.

Figure 3: Marking Packets and Assigning Actions—2-Rate Policer



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.

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. During periods of congestion, proper configuration of the excess burst parameter enables the policer to drop packets less aggressively. Therefore, it is important to understand how policing uses the committed (normal) and excess burst values to ensure the router reaches the configured committed information rate (CIR).

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.

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.

The default committed burst size is the greater of 2 milliseconds of bytes at the police rate or the network maximum transmission unit (MTU).

Committed Burst Calculation

To calculate committed burst, use the following formula:

bc = CIR bps * (1 byte) / (8 bits) * 1.5 seconds

Note 1.5 seconds is the typical round-trip time.

For example, if the committed information rate is 512000 bps, then using the committed burst formula, the committed burst is 96000 bytes.

bc = 512000 * 1/8 * 1.5

bc = 64000 * 1.5 = 96000

Note When the be value equals 0, we recommend that you set the egress bc value to be greater than or equal to the ingress bc value plus 1. Otherwise, packet loss can occur. For example: be = 0 egress bc >= ingress bc + 1

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).

Excess Burst Calculation

To calculate excess burst, use the following formula:

be = 2 * committed burst

For example, if you configure a committed burst of 4000 bytes, then using the excess burst formula, the excess burst is 8000 bytes.

be = 2 * 4000 = 8000

The default excess burst size is 0.

Deciding if Packets Conform or Exceed the Committed Rate

Policing uses normal or committed burst (bc) and excess burst (be) values to ensure that the configured committed information rate (CIR) is reached. Policing decides if a packet conforms or exceeds the CIR based on the burst values you configure. Several factors can influence the policer's decision, such as the following:

- Low burst values—If you configure burst values too low, the achieved rate might be much lower than the configured rate.

- Temporary bursts—These bursts can have a strong adverse impact on throughput of Transmission Control Protocol (TCP) traffic.

It is important that you set the burst values high enough to ensure good throughput. If your router drops packets and reports an exceeded rate even though the conformed rate is less than the configured CIR, use the show interface command to monitor the current burst, determine whether the displayed value is consistently close to the committed burst (bc) and excess burst (be) values, and if the actual rates (the committed rate and exceeded rate) are close to the configured committed rate. If not, the burst values might be too low. Try reconfiguring the burst rates using the suggested calculations in the Committed Burst Calculation, on page 29 and the Excess Burst Calculation, on page 30.

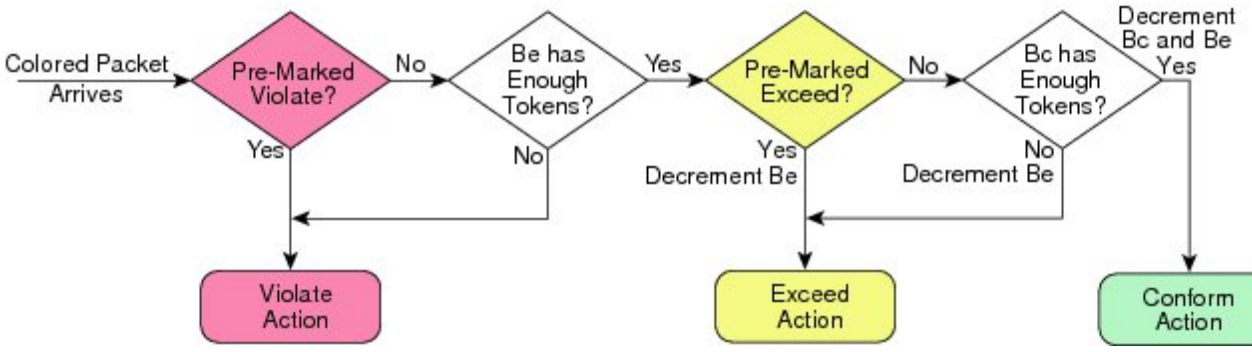
Two-Rate Three-Color (2R3C) Policer

The policer reads a preexisting marking—the frame-relay discard-eligibility (FRDE) bit in the packet header—that was set by a policer on a previous network node. By default the FRDE bit is set to 0. At the receiving node, the system uses this bit to determine the appropriate color-aware policing action for the packet:

- To classify the FRDE bit value 0 as conform color, create a conform-color class-map for frde=0 packets. This causes packets to be classified as color green, and the system applies the conform action.
- To classify the FRDE bit value 1 as exceed color, create an exceed-color class-map for frde=1 packets. This causes packets to be classified as color yellow, and the system applies the exceed action.

Note Color-aware policing is not supported for hierarchical QoS.

Figure 4: 2R3C Policing Process Flowchart



Configuring Traffic Policing (Two-Rate Color-Blind)

Traffic policing allows you to control the maximum rate of traffic sent or received on an interface. This section provides the procedure for configuring two-rate color-blind traffic policing.

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 4 **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]] | **percent** *percentage*]

Example:

```
RP/0/(config-pmap-c)# police rate 250000
```

Configures traffic policing and enters policy map police configuration mode. The traffic policing feature works with a token bucket algorithm.

Step 5 **exit**

Example:

```
RP/0/(config-pmap-c-police)# exit
```

Returns the router to policy map class configuration mode.

Step 6 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 7 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to the global configuration mode.

Step 8 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface HundredGigE 0/7/0/0
```

Enters configuration mode and configures an interface.

Step 9 **service-policy** {**input** | **output**} *policy-map*

Example:

```
RP/0/(config-if)# service-policy output policy1
```

Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

Step 10 **commit**

Step 11 **show policy-map interface** *type interface-path-id* [**input** | **output**]

Example:

```
RP/0/# show policy-map interface HundredGigE 0/7/0/0
```

(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing (2R3C)

This section provides the procedure for configuring two-rate three-color traffic policing.

Procedure

Step 1 **configure**

Step 2 **class-map** [**match-any**] *class-map-name*

Example:

```
RP/0/(config)# class-map match-all
```

Creates or modifies a class map that can be attached to one or more interfaces to specify a matching policy and enters the class map configuration mode.

Step 3 **match precedence ipv4***precedence_value***fr-de***fr-de-bit-value*

Example:

```
RP/0/(config)# match  precedence ipv4 5
```

Specifies a precedence value that is used as the match criteria against which packets are checked to determine if they belong to the class specified by the class map.

Specifies the matching condition:

- Match fr-de 1 is typically used to specify an exceed-color packet.

Step 4 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 5 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 6 **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]]

Example:

```
RP/0/(config-pmap-c)# police rate 768000 burst 288000 peak-rate 1536000 peak-burst 576000
```

Configures traffic policing and enters policy map police configuration mode. The traffic policing feature works with a token bucket algorithm.

Step 7 **exit**

Example:

```
RP/0/(config-pmap-c-police)# exit
```

Returns the router to policy map class configuration mode.

Step 8 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 9 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to global configuration mode.

Step 10 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface HundredGigE 0/7/0/0
```

Enters configuration mode and configures an interface.

Step 11 **service-policy** *policy-map*

Example:

```
RP/0/(config-if)# service-policy policy1
```

Attaches a policy map to an input interface to be used as the service policy for that interface.

Step 12 **commit**

Step 13 **show policy-map interface** *type interface-path-id*

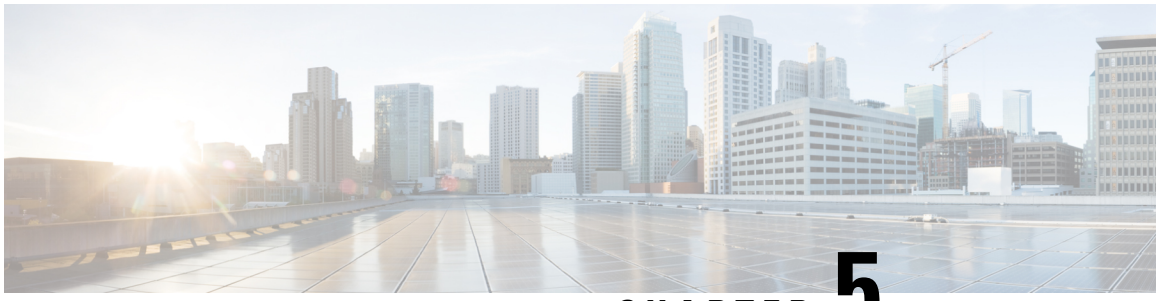
Example:

```
RP/0/# show policy-map interface HundredGigE 0/7/0/0
```

(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Running configuration for policer

```
policy-map ingress_POLICER_POLICY
class CLASS_1_POLICERIPV4PREC
  set qos-group 7
  police rate 1 gbps peak-rate 2 gbps
!
```



CHAPTER 5

Congestion Avoidance

Congestion avoidance techniques monitor traffic flow in an effort 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.

This chapter provides details regarding congestion avoidance techniques.

- Prerequisites for Configuring Modular QoS Congestion Avoidance, on page 37
- Random Early Detection and TCP, on page 37
- Tail Drop, on page 38
- Configuring Random Early Detection, on page 38
- Configuring Weighted Random Early Detection, on page 40
- Configuring Tail Drop, on page 42

Prerequisites for Configuring Modular QoS Congestion Avoidance

This prerequisite is required for configuring QoS congestion avoidance on your network:

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

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.

Queue-limit for WRED

Queue-limit is used to fine-tune the number of buffers available for each queue. It can only be used on a queuing class. Default queue limit is --------- ms of the service rate for the given queue. The service rate is the sum of minimum guaranteed bandwidth and bandwidth remaining assigned to a given class either implicitly or explicitly.

The queue-limit is rounded up to one of the following values: 8 KB, 16 KB, 24 KB, 32 KB, 48 KB, 64 KB, 96 KB, 128 KB, 192 KB, 256 KB, 384 KB, 512 KB, 768 KB, 1024 KB, 1536 KB, 2048 KB, 3072 KB, 4196 KB, 8192 KB, 16394 KB, 32768 KB, 65536 KB, 131072 KB, or 262144 KB.

Tail Drop

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.

Configuring Random Early Detection

This configuration task is similar to that used for WRED except that the **random-detect precedence** command is not configured and the **random-detect** command with the **default** keyword must be used to enable RED.

Restrictions

If you configure the **random-detect default** command on any class including class-default, you must configure one of the following commands:

- **shape average or bandwidth**

For the **random-detect** command to take effect, you must configure either the **shape average or bandwidth** command in the user defined policy map class. This dependency is not applicable to the policy map class-default.

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-map-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 4 **random-detect** {**cos** *value* | **default** | **discard-class** *value* | **dscp** *value* | **exp** *value* | **precedence** *value* | *min-threshold* [*units*] *max-threshold* [*units*] }

Example:

```
RP/0/(config-pmap-c)# random-detect default
```

Enables RED with default minimum and maximum thresholds.

Step 5 **bandwidth** {*bandwidth* [*units*] | **percent** *value*}

Example:

```
RP/0/(config-pmap-c)# bandwidth percent 30
```

(Optional) Specifies the bandwidth allocated for a class belonging to a policy map.

or

(Optional) Specifies how to allocate leftover bandwidth to various classes.

Step 6 **shape average** {**percent** *percentage* | *value* [*units*]}

Example:

```
RP/0/(config-pmap-c)# shape average percent 50
```

(Optional) Shapes traffic to the specified bit rate or a percentage of the available bandwidth.

Note
This step can be used if the bandwidth is not configured (step 5).

Step 7 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 8 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to the global configuration mode.

Step 9 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface TenGigE 0/2/0/0
```

Enters the configuration mode and configures an interface.

Step 10 **service-policy** {**input** | **output**} *policy-map*

Example:

```
RP/0/(config-if)# service-policy output policy1
```

Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

Step 11 **commit**

Configuring Weighted Random Early Detection

WRED drops packets selectively based on IP precedence. IP precedences are assigned to packets as they enter the network. WRED uses these precedences to determine how to treat different types of traffic.

Configure WRED using the **random-detect** command and different CoS, DSCP, EXP, and 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.

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.

Restrictions

You cannot configure WRED in a class that has been set for priority queueing (PQ).

You cannot use the **random-detect** command in a class configured with the **priority** command.

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 4 **random-detect discard-class** *class-id* [*min-packets max-packets*] | **default**

Example:

```
RP/0/(config-pmap-c)# random-detect discard-class 1 1 packet 2 packets
```

Modifies the minimum and maximum packet thresholds for the discard class.

Step 5 **bandwidth** {*bandwidth* [*units*] | **percent** *value*}

Example:

```
RP/0/(config-pmap-c)# bandwidth percent 30
```

(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. This example guarantees 30 percent of the interface bandwidth to class class1.

Step 6 **shape average** {**percent** *percentage* | *value* [*units*]}

Example:

```
RP/0/(config-pmap-c)# shape average percent 50
```

(Optional) Shapes traffic to the specified bit rate or a percentage of the available bandwidth.

Note
This step can be used if the bandwidth was not configured earlier (using step 5).

Step 7 **queue-limit** *value* [*units*]

Example:

```
RP/0/(config-pmap-c)# queue-limit 50 ms
```

(Optional) Changes queue-limit to fine-tune the amount of buffers available for each queue. The default queue-limit is 100 ms of the service rate for a non-priority class and 10ms of the service rate for a priority class.

Step 8 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to the global configuration mode.

Step 9 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface 0/2/0/0
```

Enters the configuration mode and configures an interface.

Step 10 **service-policy** {**input** | **output**} *policy-map*

Example:

```
RP/0/(config-if)# service-policy output policy1
```

Attaches a policy map to an input or output interface to be used as the service policy for that interface.

- In this example, the traffic policy evaluates all traffic leaving that interface.
- Ingress policies are not valid; the **bandwidth** and **bandwidth remaining** commands cannot be applied to ingress policies.

Step 11 **commit**

Running configuration for WRED

```
policy-map egress_WRED_POLICY_L3
class class2
  shape average 5 gbps
  random-detect discard-class 0 0 bytes 10 bytes
!
```

Configuring 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, enqueued packets to the class queue result in tail drop (packet drop).

The **queue-limit** value uses the guaranteed service rate (GSR) of the queue as the reference value for the **queue_bandwidth**. If the class has bandwidth percent associated with it, the **queue-limit** is set to a proportion of the bandwidth reserved for that class.

If the GSR for a queue is zero, use the following to compute the default **queue-limit**:

- 1 percent of the interface bandwidth for queues in a nonhierarchical policy.
- 1 percent of parent maximum reference rate for hierarchical policy.

  The parent maximum reference rate is the minimum of parent shape, policer maximum rate, and the interface bandwidth.

Note The default **queue-limit** is set to bytes of 100 ms of queue bandwidth. The following formula is used to calculate the default queue limit (in bytes):??bytes = (100 ms / 1000 ms) * queue_bandwidth kbps)) / 8

The default **queue-limit** is set as follows:

default queue limit (in bytes) = (<200|100|10> ms * queue_bandwidth kbps) / 8

Note You can configure the queue limit in all the priority classes.

The default **queue-limit** is set as follows:

- **For priority class**

  default queue limit (in bytes) = (<10> ms * queue_bandwidth kbps) / 8

- **For non-priority class**

  default queue limit (in bytes) = (<100> ms * queue_bandwidth kbps) / 8



Note You can configure a maximum queue threshold up to 1GB, which translates to 80ms of buffering for 100Gbps queue.

Restrictions

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

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and also enters the policy map configuration mode.

Step 3 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 4 (optional) **queue-limit** *value* [*units*]

Example:

```
RP/0/(config-pmap-c)# queue-limit 1000000 bytes
```

Specifies or modifies the maximum the queue can hold for a class policy configured in a policy map. The default value of the *units* argument is **packets**. In this example, when the queue limit reaches 1,000,000 bytes, enqueued packets to the class queue are dropped.

Step 5 **priority** [**level** *priority-level*]

Example:

```
RP/0/(config-pmap-c)# priority level 1
```

Specifies priority to a class of traffic belonging to a policy map.

Step 6 **police rate percent** *percentage*

Example:

```
RP/0/(config-pmap-c)# police rate percent 30
```

Configures traffic policing.

Step 7 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class2
```

Specifies the name of the class whose policy you want to create or change. In this example, class2 is configured.

Step 8 **bandwidth** {*bandwidth* [*units*] | **percent** *value*}

Example:

```
RP/0/(config-pmap-c)# bandwidth percent 30
```

(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. This example guarantees 30 percent of the interface bandwidth to class class2.

Step 9 (optional) **queue-limit** *value* [*units*]

Example:

```
RP/0/(config-pmap-c)# queue-limit 1000000 bytes
```

Specifies or modifies the maximum the queue can hold for a class policy configured in a policy map. The default value of the *units* argument is **packets**. In this example, when the queue limit reaches 1,000,000 bytes, enqueued packets to the class queue are dropped.

Step 10 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class2
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 11 **bandwidth remaining percent** *value*

Example:

```
RP/0/(config-pmap-c)# bandwidth remaining percent 20
```

(Optional) Specifies how to allocate leftover bandwidth to various classes. This example allocates 20 percent of the leftover interface bandwidth to class class2.

Step 12 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class3
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 13 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 14 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to the global configuration mode.

Step 15 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface HundredGigE 0/7/0/0
```

Enters the configuration mode and configures an interface.

Step 16 **service-policy** {**input** | **output**} *policy-map*

Example:

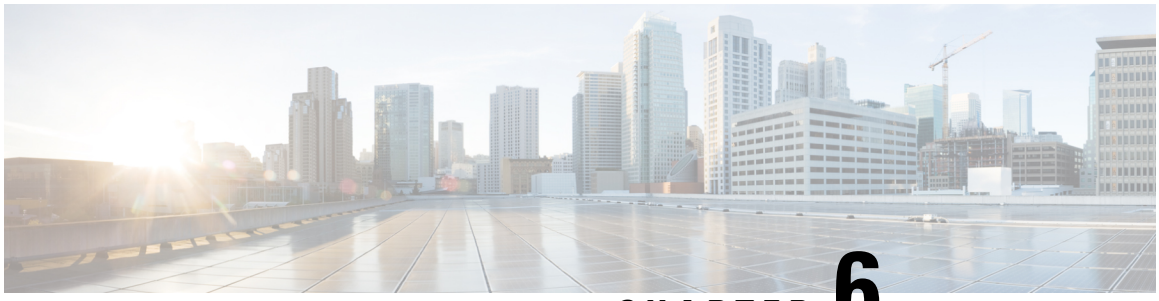
```
RP/0/(config-if)# service-policy output policy1
```

Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

Step 17 **commit**

Running configuration for tail drop

```
policy-map egress_BRRpriority_POLICY
class CLASS_3_egress_BRRpriorityIPV4DSCP
  bandwidth remaining ratio 1
!
class CLASS_1_egress_BRRpriorityIPV4DSCP
  bandwidth remaining ratio 10
!
class class-default
!
end-policy-map
!
```



CHAPTER 6

Congestion Management

Congestion management controls congestion after it has occurred on a network. Congestion is managed by using packet queuing methods and by shaping the packet flow through use of traffic regulation mechanisms.

This chapter provides details about congestion management techniques.

- Prerequisites for Configuring QoS Congestion Management, on page 47
- Information About Configuring Congestion Management, on page 47
- Congestion Management Overview, on page 47
- Traffic Shaping, on page 48
- How to Configure QoS Congestion Management, on page 49

Prerequisites for Configuring QoS Congestion Management

These prerequisites are required for configuring QoS congestion management on your network:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You must be familiar with Cisco IOS XR QoS configuration tasks and concepts.

Information About Configuring Congestion Management

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. The congestion management features allow you to specify creation of a different number of queues, affording greater or lesser degree of differentiation of traffic, and to specify the order in which that traffic is sent.

During periods with light traffic flow, that is, when no congestion exists, packets are sent out the interface as soon as they arrive. During periods of transmit congestion at the outgoing interface, packets arrive faster than

the interface can send them. If you use congestion management features, packets accumulating at an interface are queued until the interface is free to send them; they are then scheduled for transmission according to their assigned priority and the queuing method configured for the interface. The router determines the order of packet transmission by controlling which packets are placed in which queue and how queues are serviced with respect to each other.

In addition to queuing methods, QoS congestion management mechanisms, such as policers and shapers, are needed to ensure that a packet adheres to a contract and service. Both policing and shaping mechanisms use the traffic descriptor for a packet.

Policers and shapers usually identify traffic descriptor violations in an identical manner through the token bucket mechanism, but they differ in the way they respond to violations. A policer typically drops traffic flow; whereas, a shaper delays excess traffic flow using a buffer, or queuing mechanism, to hold the traffic for transmission at a later time.

Traffic shaping and policing can work in tandem. For example, a good traffic shaping scheme should make it easy for nodes inside the network to detect abnormal flows.

Low-Latency Queueing with Strict Priority Queueing

Strict Priority Queueing (PQ) allows delay-sensitive data, such as voice, to be dequeued and sent before packets in other queues are dequeued. Upto four priority queues are supported. The highest priority queue is referred to as PQ1; the lowest priority queue is referred to as PQ4.

Low-Latency Queueing (LLQ) enables the use of a single, strict priority queue at the class level, allowing you to direct traffic belonging to a class. To rank class traffic to the strict priority queue, you specify the named class within a policy map and then configure the **priority** command for the class. (Classes to which the **priority** command is applied are considered priority classes.) Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is enqueued to the same, single, strict priority queue.

Through use of the **priority** command, you can assign a strict PQ to any of the valid match criteria used to specify traffic. These methods of specifying traffic for a class include matching on protocols, IP precedence, and IP differentiated service code point (DSCP) values.

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, thereby eliminating bottlenecks in topologies with data-rate mismatches.

To match the rate of transmission of data from the source to the target interface, you can limit the transfer of data to one of the following:

- A specific configured rate
- A derived rate based on the level of congestion

The rate of transfer depends on these three components that constitute the token bucket: burst size, mean rate, and time (measurement) interval. The mean rate is equal to the burst size divided by the interval.

When traffic shaping is enabled, the bit rate of the interface does not exceed the mean rate over any integral multiple of the interval. In other words, during every interval, a maximum of burst size can be sent. Within the interval, however, the bit rate may be faster than the mean rate at any given time.

When the peak burst size equals 0, the interface sends no more than the burst size every interval, achieving an average rate no higher than the mean rate. However, when the peak burst size is greater than 0, the interface can send as many as the burst size plus peak burst bits in a burst, if in a previous time period the maximum amount was not sent. Whenever less than the burst size is sent during an interval, the remaining number of bits, up to the peak burst size, can be used to send more than the burst size in a later interval.

Regulation of Traffic with the Shaping Mechanism

When incoming packets arrive at an interface, the packets are classified using a classification technique, such as the setting of the IP Precedence bits through the Modular QoS CLI (MQC). If the packet matches the specified classification, the traffic-shaping mechanism continues. Otherwise, no further action is taken.

How to Configure QoS Congestion Management

Configuring Guaranteed and Remaining Bandwidths

The **bandwidth** command allows you to specify the minimum guaranteed bandwidth to be allocated for a specific class of traffic.

The **bandwidth remaining** command specifies a weight for the class. If you do not configure the **bandwidth remaining** command for any class, the leftover bandwidth is allocated equally to all classes for which **bandwidth remaining** is not explicitly specified.

Guaranteed Service rate of a queue is defined as the bandwidth the queue receives when all the queues are congested. It is defined as:

Guaranteed Service Rate = minimum bandwidth + excess share of the queue

Restrictions

- The bandwidth reservation should be at the rate configured at Layer 1.
- The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.
- A policy map can have all class bandwidths specified in kilobits per second or percentages but not a mixture of both in the same class.

The **bandwidth** command is supported only on policies configured on outgoing interfaces.

Configuring Guaranteed Bandwidth

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Enters policy map configuration mode.

- Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change.

Step 4 **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}

Example:

```
RP/0/(config-pmap-c)# bandwidth percent 40
```

Enters policy map class configuration mode.

- Specifies the bandwidth allocated for a class belonging to a policy map.
- In this example, class class1 is guaranteed 40 percent of the interface bandwidth.

Step 5 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 6 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class2
```

Specifies the name of the class whose policy you want to create or change.

Step 7 **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}

Example:

```
RP/0/(config-pmap-c)# bandwidth percent 40
```

Enters policy map class configuration mode.

- Specifies the bandwidth allocated for a class belonging to a policy map.
- In this example, class class2 is guaranteed 40 percent of the interface bandwidth.

Step 8 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 9 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class-default
```

Specifies the name of the class whose policy you want to create or change.

Step 10 **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}

Example:

```
RP/0/(config-pmap-c)# bandwidth percent 20
```

Enters policy map class configuration mode.

- Specifies the bandwidth allocated for a class belonging to a policy map.
- In this example, class class-default is guaranteed 20 percent of the interface bandwidth.

Step 11 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 12 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to global configuration mode.

Step 13 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface tengige 0/2/0/0
```

Enters interface configuration mode and configures an interface.

Step 14 **service-policy output** *policy-map*

Example:

```
RP/0/ (config-if)# service-policy output policy1
```

Attaches a policy map to an output interface to be used as the service policy for that interface.

- In this example, the traffic policy evaluates all traffic leaving that interface.

Step 15 **end** or **commit**

Example:

```
RP/0/(config-if)# end
```

or

```
RP/0/(config-if)# commit
```

Saves configuration changes.

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

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

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

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

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

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

Step 16 **show policy-map interface** *type interface-path-id* **output**

Example:

```
RP/0/ # show policy-map interface tengige 0/2/0/0
```

(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Bandwidth Remaining

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Enters policy map configuration mode.

- Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change.

Step 4 **bandwidth remaining percent** *percentage-value*

Example:

```
RP/0/(config-pmap-c)# bandwidth remaining percent 40
```

Specifies how to allocate leftover bandwidth for class class1.

Step 5 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 6 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class2
```

Specifies the name of the class whose policy you want to create or change.

Step 7 **bandwidth remaining percent** *percentage-value*

Example:

```
RP/0/(config-pmap-c)# bandwidth remaining percent 40
```

Specifies how to allocate leftover bandwidth for class class2.

Step 8 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 9 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class-default
```

Specifies the name of the class whose policy you want to create or change.

Step 10 **bandwidth remaining percent** *percentage-value*

Example:

```
RP/0/(config-pmap-c)# bandwidth remaining percent 20
```

Specifies how to allocate leftover bandwidth for class class-default.

Step 11 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 12 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to global configuration mode.

Step 13 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface tengige 0/2/0/0
```

Enters interface configuration mode and configures an interface.

Step 14 **service-policy output** *policy-map*

Example:

```
RP/0/(config-if)# service-policy output policy1
```

Attaches a policy map to an output interface to be used as the service policy for that interface.

- In this example, the traffic policy evaluates all traffic leaving that interface.

Step 15 **end** or **commit**

Example:

```
RP/0/(config-if)# end
```

or

```
RP/0/(config-if)# commit
```

Saves configuration changes.

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

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

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

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

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

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

Step 16 **show policy-map interface** *type interface-path-id* **output**

Example:

```
RP/0/ # show policy-map interface tengige 0/2/0/0
```

(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Low-Latency Queueing with Strict Priority Queueing

The **priority** command configures LLQ with strict priority queuing (PQ) that allows delay-sensitive data such as voice to be dequeued and sent before packets in other queues are dequeued. When a class is marked as high priority using the **priority** command, you must configure a policer to limit the priority traffic. This configuration ensures that the priority traffic does not constrain all the other traffic on the line card, which protects low priority traffic from limitations. Use the **police** command to explicitly configure the policer.

Note Five levels of priorities are supported: priority level 1, priority level 2, priority level 3, priority level 4, and the priority level normal. If no priority level is configured, the default is priority level normal.

Restrictions

- Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is queued to the same single priority queue.
- The **shape average**,**bandwidth**, and **random-detect**commands cannot be configured in the same class with the **priority** command.

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map voice
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class voice
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 4 **priority**[**level** *priority_level*]

Example:

```
RP/0/(config-pmap-c)# priority level 1
```

Specifies priority to a class of traffic belonging to a policy map. If no priority level is configured, the default is priority 1.

Step 5 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to policy map class configuration mode.

Step 6 **police rate** {[*units*] | **percent** *percentage*}} [**burst**] [**peak-burst** *peak-burst* [*burst-units*]] [**peak-rate** *value* [*units*]] | **percent** *percentage*]

Example:

```
RP/0/(config-pmap-c)# police rate 250
```

Configures traffic policing and enters policy map police configuration mode. In this example, the low-latency queue is restricted to 250 kbps to protect low-priority traffic from starvation and to release bandwidth.

Note
For Link Aggregation Group (LAG), only the **percent** keyword is supported.

Step 7 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 8 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to the global configuration mode.

Step 9 **interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface
```

Enters interface configuration mode, and configures an interface.

Step 10 **service-policy output** *policy-map*

Example:

```
RP/0/(config-if)# service-policy output policy1
```

Attaches a policy map to an output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

Step 11 **commit**

Step 12 **show policy-map interface** *type interface-path-id* **output**

Example:

```
RP/0/# show policy-map interface tengige POS 0/2/0/0HundredGigE 0/7/0/0
```

(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Running configuration for priority

```
policy-map egress_BRRpriority_POLICY
class CLASS_2_egress_BRRpriorityIPV4DSCP
  priority level 4
  police rate 8 gbps
  !
!
class CLASS_3_egress_BRRpriorityIPV4DSCP
  bandwidth remaining ratio 1
!
class CLASS_1_egress_BRRpriorityIPV4DSCP
  bandwidth remaining ratio 10
!
class class-default
!
end-policy-map
!
```

How to Mitigate Control Packet Loss during Traffic Congestion at Core Interface

During a network congestion at the core interfaces, the high priority control packets may be lost randomly. To avoid this loss, we recommend employing egress queueing. This approach involves creating an egress queing policy map that prioritizes Traffic Class (TC) 6 and 7 control packets, thus ensuring uninterrupted transmission while shaping the data traffic to a maximum of 90% of the available bandwidth.

The following sample summarizes how egress queueing prioritizes the TC6 and TC7 control packets.

```
class-map match-any TC6
 match traffic-class 6
 end-class-map
!
class-map match-any TC7
 match traffic-class 7
 end-class-map
!
policy-map out_p
 class TC6
  priority level 1
 !
 class TC7
```

```
 priority level 1
!
class class-default
 shape average percent 90
!
end-policy-map
!
```

Configuring Traffic Shaping

Traffic shaping allows you to control the traffic 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.

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3 **class** *class-name*

Example:

```
RP/0/(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 4 **shape average** {**percent** *value* | *rate* [*units*]} [*burst-size* [*burst-units*]]

Example:

```
RP/0/(config-pmap-c)# shape average percent 50
```

Shapes traffic to the indicated bit rate according to average rate shaping in the specified units or as a percentage of the bandwidth.

Note
For Link Aggregation Group (LAG), only the **percent** keyword is supported.

Step 5 **exit**

Example:

```
RP/0/(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 6 **exit**

Example:

```
RP/0/(config-pmap)# exit
```

Returns the router to global configurationmode.

Step 7 Specifies the name of the class whose policy you want to create or change.**interface** *type interface-path-id*

Example:

```
RP/0/(config)# interface tengige POS 0/2/0/0
```

Enters interface configuration mode and configures an interface.

Step 8 **service-policy output** *policy-map*

Example:

```
RP/0/(config-if)# service-policy output policy1
```

Attaches a policy map to an output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

Step 9 **commit**

Step 10 **show policy-map interface** *type interface-path-id* **output**

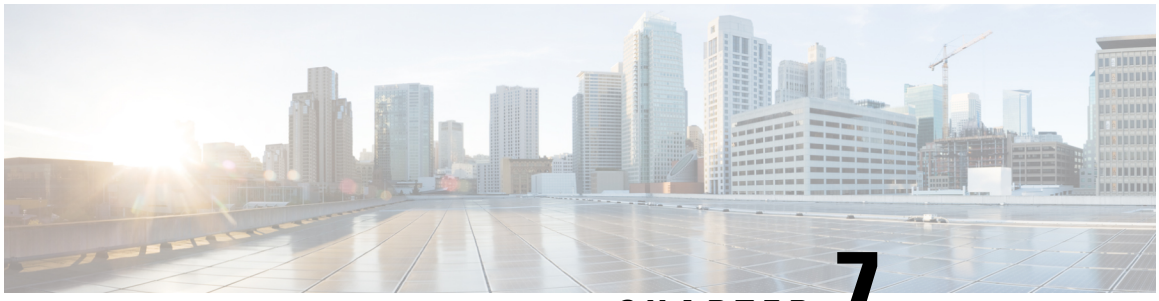
Example:

```
RP/0/# show policy-map interface HundredGigE 0/7/0/0
```

(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Running configuration for traffic shaping

```
policy-map egress_SHAPER_POLICY
class CLASS_1_egress_SHAPERIPV4PREC
  shape average 1000 mbps
!
```



CHAPTER 7

Hierarchical QoS

This chapter includes details of hierarchical QoS.

- Information About Hierarchical QoS, on page 61
- Two-Level Hierarchical Policies, on page 61

Information About Hierarchical QoS

Hierarchical QoS allows you to specify QoS behavior at multiple policy levels, which provides a high degree of granularity in traffic management.

HQoS is not supported on Link Aggregation Group (LAG).

Two-Level Hierarchical Policies

Two-level hierarchical policies, also called *nested polices*, can be illustrated with a parent-level policy for the top level of the hierarchy and a child-level for the bottom level of the hierarchy. A two-level hierarchical policy can have queueing or marking or policing at child level and policing or shaping or bandwidth at parent level.

Four levels of priority are supported —priority level 1, 2, 3 and 4. These priority levels can be used along with the normal-priority queues. The normal-priority queues are scheduled by a different scheduler that does not give any priority treatment to the packets. Priority levels are supported only in the egress direction.

Note Whenever a policy with unsupported combination is applied, a failure message is displayed.

Configuring Hierarchical Policing

Hierarchical policing provides support at two levels:

- Parent level
- Child level

In the hierarchical ingress policy, **policer** command is supported at the parent level. In the hierarchical egress policy, **policer** command is not supported.

Procedure

Step 1 **configure**

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/(config)# policy-map policy1
```

Enters policy map configuration mode.

- Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3 **class default**

Example:

```
RP/0/(config-pmap)# class default
```

Enters policy map class configuration mode.

Step 4 **service-policy** *policy-map-name*

Example:

```
RP/0/(config-pmap-c)# service-policy child
```

Attaches a policy map to an input or output interface to be used as the service policy for that interface.

Step 5 **police rate percent** *percentage*

Example:

```
RP/0/(config-pmap-c)# police rate percent 50
```

Configures traffic policing and enters policy map police configuration mode.

Step 6 **end** or **commit**

Example:

```
RP/0/(config-if)# end
```

or

```
RP/0/(config-if)# commit
```

Saves configuration changes.

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

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

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

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

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

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

Configuration example for HQoS

Configuration example for Egress HQoS on L3 sub-interface

```
class-map match-any CLASS_1_IPV4PREC
 match precedence 6
 end-class-map

policy-map child_POLICY_
 class CLASS_1_IPV4PREC
  set qos-group 6
  police rate percent 17 peak-rate percent 25
end-policy-map

policy-map parent_POLICY
 class class-default
  service-policy child_POLICY
 end-policy-map

interface TenGigE0/3/0/2
 service-policy input parent_POLICY
 ipv4 address 90.0.0.1 255.255.255.0
```

```
class-map match-any match_exp_4
match mpls experimental topmost 4
 end-class-map
!
class-map match-any match_exp_5
match mpls experimental topmost 5
 end-class-map
!

class-map match-any match_exp_1
match mpls experimental topmost 1
 end-class-map
!
class-map match-any match_exp_2
match mpls experimental topmost 2
 end-class-map
!
class-map match-any match_exp_3
match mpls experimental topmost 3
 end-class-map
!
class-map match-any control_class_6
match dscp cs6
```

```
 match precedence 6
 match mpls experimental topmost 6
 end-class-map
!
class-map match-any control_class_7
match dscp cs7
 match precedence 7
 match mpls experimental topmost 7
 end-class-map
!


class-map match-any match_qos_group1
match qos-group 1
 end-class-map
!


class-map match-any match_qos_group2
match qos-group 2
 end-class-map
!


class-map match-any match_qos_group3
match qos-group 3
 end-class-map
!


class-map match-any match_qos_group4
match qos-group 4
 end-class-map
!


class-map match-any match_qos_group5
match qos-group 5
 end-class-map
!


class-map match-any match_qos_group6
match qos-group 6
 end-class-map
!

class-map match-any match_qos_group7
match qos-group 7
 end-class-map
!

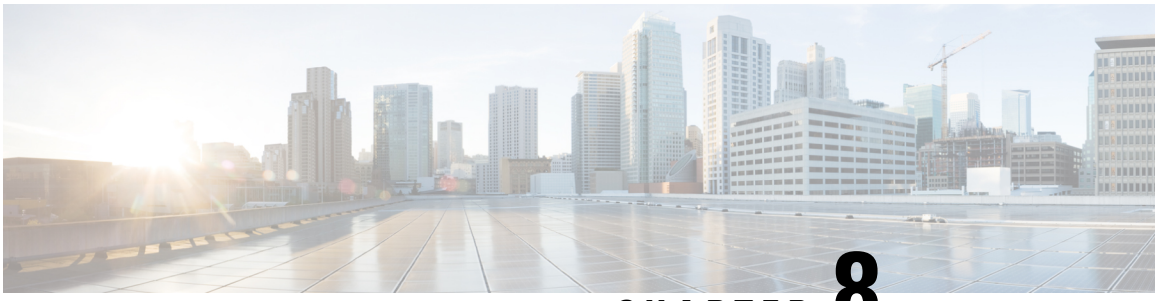


policy-map policy_classify_exp_new
class match_exp_1
  set qos-group 1
!
 class match_exp_2
  set qos-group 2
!
 class match_exp_3
  set qos-group 3
!
```

```
 class match_exp_4
  set qos-group 4
!
 class match_exp_5
  set qos-group 5
!
 class control_class_6
  set qos-group 6
!
 class control_class_7
  set qos-group 7
!
 class class-default
!
 end-policy-map
!


policy-map hqos_child_policy_42xx
class match_qos_group1
  bandwidth 1 gbps
 !
 class match_qos_group2
  bandwidth 1 gbps
 !
 class match_qos_group3
  bandwidth 1 gbps
 !
 class match_qos_group4
  bandwidth 1 gbps
 !
 class match_qos_group5
  bandwidth 1 gbps
 !
 class match_qos_group6
  priority level 1
  police rate 1 gbps
  !
 !
 class match_qos_group7
  priority level 2
  police rate 1 gbps
  !
 !
 class class-default
!
 end-policy-map
!


policy-map hqos_parent_policy_42xx
class class-default
  service-policy hqos_child_policy_42xx
  shape average 5 gbps
 !
 end-policy-map
!
```



CHAPTER 8

Dual Policy

Dual policy enables support for two output policies. The output policies are based on traffic class and qos-group. Traffic class is used for queue selection and queuing policy. Qos-group is used for marking policies.

This chapter provides the conceptual and configuration details for dual policy.

- Dual Policy , on page 67

Dual Policy

To achieve QoS Egress marking/queuing, the Cisco NCS 4000 Series Routers utilize the dual policy model on egress with independent policies for marking and queuing.

Egress marking can be achieved by applying a policy-map on the ingress interface. Similarly, egress queuing can be achieved by applying a policy-map on the ingress interface by setting the traffic-class. Then the traffic-class is used by the egress-policy map to perform queuing actions.

Cisco NCS 4000 Series router supports dual policies on a single interface (egress only).

QoS Egress Marking and Queueing is summarised as below:

- Configure an ingress policy-map

```
Create a class-map
Router#config
Router(config)#class-map match-any cos2
Router(config-cmap)#match cos 2
Router(config-cmap)#commit
Router(config)#class-map match-any cos3
Router(config-cmap)#match cos 3
Router(config-cmap)#commit
Router(config)#class-map match-any cos4
Router(config-cmap)#match cos 4
Router(config-cmap)#commit

Create classification policies
Router#config
Router(config)#policy-map ingress-classification
Route(config-pmap)#class cos 2
Router(config-pmap-c)#set qos-group 1
Router(config-pmap-c)#set traffic-class 3
Router(config-pmap-c)#class cos3
Router(config-pmap-c)#set qos-group 2
Router(config-pmap-c)#set traffic-class 5
Router(config-pmap-c)#class cos4
```

```
Router(config-pmap-c)#set qos-group 3
Router(config-pmap-c)#set traffic-class 4
Router(config-pmap-c)#class class-default
Router(config-pmap-c)#set qos-group 7
Router(config-pmap-c)#set traffic-class 6
Router(config-pmap-c)#commit
```

- Configure an egress policy-map

```
Create egress marking policy
Router#config
Router(config)#class-map match-any qos1
Router(config-cmap)#match qos-group 1
Router(config-cmap)#commit
Router(config)#class-map match-any qos2
Router(config-cmap)#match qos-group 2
Router(config-cmap)#commit
Router(config)#class-map match-any qos3
Router(config-cmap)#match qos-group 3
Router(config-cmap)#commit
Router#config
Router(config)#policy-map egress-marking
Route(config-pmap)#class qos1
Router(config-pmap-c)#set cos 1
Router(config-pmap-c)#class qos2
Router(config-pmap-c)#set cos 2
Router(config-pmap-c)#set dei 1
Router(config-pmap-c)#class qos3
Router(config-pmap-c)#set cos 3
Router(config-pmap-c)#class class-default
Router(config-pmap-c)#set cos 7
Router(config-pmap-c)#commit

Create Egress queueing policy
Router#config
Router(config)#class-map match-any tc3
Router(config-cmap)#match traffic-class 3
Router(config-cmap)#commit
Router(config)#class-map match-any tc4
Router(config-cmap)#match traffic-class 3
Router(config-cmap)#commit
Router(config)#class-map match-any tc5
Router(config-cmap)#match traffic-class 3
Router(config-cmap)#commit
Router#config
Router(config)#policy-map egress-queuing
Route(config-pmap)#class tc3
Router(config-pmap-c)#shape average 2 mbps
Router(config-pmap-c)#class tc4
Router(config-pmap-c)#shape average 5 mbps
Router(config-pmap-c)#class tc5
Router(config-pmap-c)#shape average 7 mbps
Router(config-pmap-c)#class class-default
Router(config-pmap-c)#commit
```

- Attach the policies to an interface

```
Router#config
Router(config)#interface tenGigE 0/0/1/0/0
Router(config-if)#service-policy input ingress-classification
Router(config-if)#service-policy output egress-marking
Router(config-if)#service-policy output egress-queuing
Router(config-if)#commit
```