Quality of Service Configuration Guide, Cisco IOS XE Amsterdam 17.1.x (Catalyst 9400 Switches)

First Published: 2019-11-26

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: www.cisco.com go trademarks. 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)

© 2019 Cisco Systems, Inc. All rights reserved.

CONTENTS

CHAPTER 1

Configuring Auto-QoS 1

Prerequisites for Auto-QoS 1

Restrictions for Auto-QoS 1

Information About Configuring Auto-QoS 1

Auto-QoS Overview 1

Auto-QoS Compact Overview 2

Auto-QoS Global Configuration Templates 2

Auto-QoS Policy and Class Maps 2

Effects of Auto-QoS on Running Configuration 3

Effects of Auto-Qos Compact on Running Configuration 3

How to configure Auto-QoS 3

Configuring Auto-QoS 3

Upgrading Auto-QoS 6

Enabling Auto-Qos Compact 8

Monitoring Auto-QoS 9

Troubleshooting Auto-QoS 9

Configuration Examples for Auto-QoS 9

Example: auto qos trust cos 9

Example: auto qos trust dscp 12

Example: auto qos video cts 14

Example: auto qos video ip-camera 16

Example: auto qos video media-player 18

Example: auto qos voip trust 20

Example: auto qos voip cisco-phone 22

Example: auto qos voip cisco-softphone 25

auto qos global compact 29

Where to Go Next for Auto-QoS 30
Feature History for Auto-QoS 30

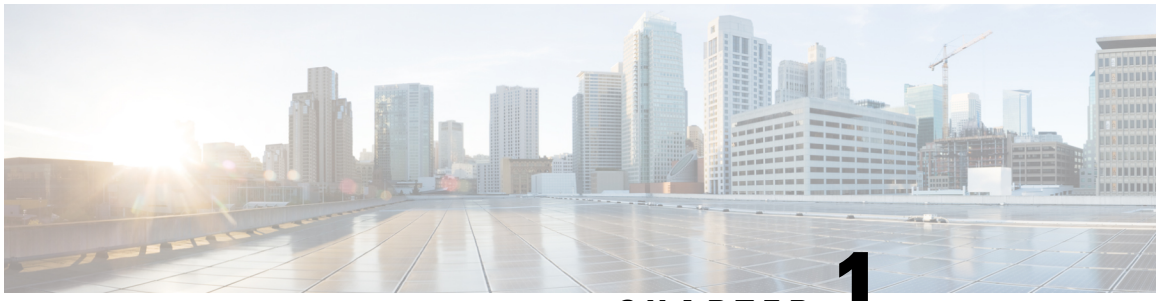
CHAPTER 2 **Configuring QoS** 31
Prerequisites for QoS 31
QoS Components 32
QoS Terminology 32
Information About QoS 32
Modular QoS CLI 33
Supported QoS Features for Wired Access 33
Hierarchical QoS 34
QoS Implementation 34
Layer 2 Frame Prioritization Bits 35
Layer 3 Packet Prioritization Bits 36
End-to-End QoS Solution Using Classification 36
Packet Classification 36
Classification Based on Information that is Propagated with a Packet 37
Classification Based on Information that is Device Specific 38
QoS Wired Model 38
Ingress Port Activity 39
Egress Port Activity 39
Classification 39
Access Control Lists 39
Class Maps 40
Time-to-Live Classification 41
Layer 3 Packet Length Classification 41
Layer 2 SRC-Miss or DST-Miss Classification 42
Policy Maps 43
Policy Map on Physical Port 44
Policy Map on VLAN 44
QoS Profile 45
Security Group Classification 45
SGT Based QoS 45
Sharing DGID with SGACL 45

Restrictions for SGT Based QoS 46
Ingress Port FIFO Parser 46
Policing 48
Token-Bucket Algorithm 48
Marking 49
Packet Header Marking 49
Switch-Specific Information Marking 49
Table Map Marking 49
Traffic Conditioning 50
Policing 51
Single-Rate Two-Color Policing 52
Dual-Rate Three-Color Policing 52
Shaping 52
Class-Based Traffic Shaping 52
Queuing and Scheduling 53
Bandwidth 54
Bandwidth Percent 54
Bandwidth Remaining Percent 55
Weighted Tail Drop 55
Weighted Tail Drop Default Values 56
Priority Queues 56
Priority Queue Policer 56
Queue Buffer 57
Queue Buffer Allocation 58
Dynamic Threshold and Scaling 58
Weighted Random Early Detection 58
Trust Behavior 59
Port Security on a Trusted Boundary for Cisco IP Phones 59
Trust Behavior for Wired Ports 59
Standard QoS Default Settings 60
Default Wired QoS Configuration 60
DSCP Maps 60
Restrictions for QoS on Wired Targets 61
How to Configure QoS 64

Configuring Class, Policy, and Maps 64
Creating a Traffic Class 64
Creating a Traffic Policy 66
Configuring Class-Based Packet Marking 70
Attaching a Traffic Policy to an Interface 75
Classifying, Policing, and Marking Traffic on Physical Ports by Using Policy Maps 77
Classifying and Marking Traffic by Using Policy Maps 81
Configuring Table Maps 84
Configuring QoS Features and Functionality 86
Configuring Bandwidth 86
Configuring Police 88
Configuring Priority 90
Configuring SGT based QoS 92
Configuring Queues and Shaping 94
Configuring Egress Queue Characteristics 94
Configuring Queue Buffers 94
Configuring Queue Limits 97
Configuring Shaping 99
Configuring Sharped Profile Queuing 100
Monitoring QoS 103
Configuration Examples for QoS 103
Examples: TCP Protocol Classification 103
Examples: UDP Protocol Classification 104
Examples: RTP Protocol Classification 105
Examples: Classification by Access Control Lists 105
Examples: Class of Service Layer 2 Classification 106
Examples: Class of Service DSCP Classification 106
Examples: VLAN ID Layer 2 Classification 106
Examples: Classification by DSCP or Precedence Values 106
Examples: Hierarchical Classification 107
Examples: Hierarchical Policy Configuration 107
Examples: Classification for Voice and Video 108
Examples: Average Rate Shaping Configuration 109
Examples: Queue-limit Configuration 110

Examples: Queue Buffers Configuration 111
Examples: Policing Action Configuration 111
Examples: Policer VLAN Configuration 112
Examples: Policing Units 112
Examples: Single-Rate Two-Color Policing Configuration 112
Examples: Dual-Rate Three-Color Policing Configuration 113
Examples: Table Map Marking Configuration 113
Example: Table Map Configuration to Retain CoS Markings 114
Where to Go Next 114
Additional References for QoS 115
Feature History for QoS 115

CHAPTER 3 **Configuring Weighted Random Early Detection** 117
Avoiding Network Congestion 117
Tail Drop 117
Weighted Random Early Detection 117
How WRED Works 118
WRED Weight Calculation 118
Limitations for WRED Configuration 118
Usage Guidelines for WRED 119
Configuring WRED 120
Configuring WRED based on DSCP Values 120
Configuring WRED based on Class of Service Values 121
Configuring WRED based on IP Precedence Values 122
Configure WRED under Class Default 123
WRED Configuration Example 124
Support for WRED with Hierarchical QoS 124
Verifying WRED Configuration 125
Best Practices for WRED Configuration 126
Feature History for Weighted Random Early Detection 128



CHAPTER 1

Configuring Auto-QoS

- Prerequisites for Auto-QoS, on page 1
- Restrictions for Auto-QoS, on page 1
- Information About Configuring Auto-QoS, on page 1
- How to configure Auto-QoS, on page 3
- Monitoring Auto-QoS, on page 9
- Troubleshooting Auto-QoS, on page 9
- Configuration Examples for Auto-QoS, on page 9
- Where to Go Next for Auto-QoS, on page 30
- Feature History for Auto-QoS, on page 30

Prerequisites for Auto-QoS

The prerequisites for auto-QoS are the same as the prerequisites for standard QoS.

Restrictions for Auto-QoS

The following are restrictions for auto-QoS:

- Auto-qos is not supported on SVI interfaces.
- Do not configure the **auto qos voip cisco-phone** option for IP phones that support video. This option causes DSCP markings of video packets to get overwritten, because these packets do not have Expedited Forwarding priority, which results in these packets getting classified in the class-default class.

Information About Configuring Auto-QoS

Auto-QoS Overview

You can use the auto-QoS feature to simplify the deployment of QoS features. Auto-QoS determines the network design and enables QoS configurations so that the switch can prioritize different traffic flows.

The switch employs the MQC model. This means that instead of using certain global configurations, auto-QoS applied to any interface on a switch configures several global class maps and policy maps.

Auto-QoS matches traffic and assigns each matched packet to qos-groups. This allows the output policy map to put specific qos-groups into specific queues, including into the priority queue.

QoS is needed in both directions, both on inbound and outbound. When inbound, the switch port needs to trust the DSCP in the packet (done by default). When outbound, the switch port needs to give voice packets "front of line" priority. If voice is delayed too long by waiting behind other packets in the outbound queue, the end host drops the packet because it arrives outside of the receive window for that packet.

Auto-QoS Compact Overview

When you enter an auto-QoS command, the switch displays all the generated commands as if the commands were entered from the CLI. You can use the auto-QoS compact feature to hide the auto-QoS generated commands from the running configuration. This would make it easier to comprehend the running-configuration and also help to increase efficient usage of memory.

Auto-QoS Global Configuration Templates

In general, an auto-QoS command generates a series of class maps that either match on ACLs or on DSCP and/or CoS values to differentiate traffic into application classes. An input policy is also generated, which matches the generated classes and in some cases, polices the classes to a set bandwidth. Eight egress-queue class maps are generated. The actual egress output policy assigns a queue to each one of these eight egress-queue class maps.

The auto-QoS commands only generate templates as needed. For example, the first time any new auto-QoS command is used, global configurations that define the eight queue egress service-policy are generated. From this point on, auto-QoS commands applied to other interfaces do not generate templates for egress queuing because all auto-QoS commands rely on the same eight queue models, which have already been generated from the first time a new auto-QoS command was used.

Auto-QoS Policy and Class Maps

After entering the appropriate auto-QoS command, the following actions occur:

- Specific class maps are created.
- Specific policy maps (input and output) are created.
- Policy maps are attached to the specified interface.
- Trust level for the interface is configured.

Note For all Auto-QoS policies, ensure that you specify the behaviour for untrusted traffic.

For untrusted traffic, no statistics are updated for class-default.

Effects of Auto-QoS on Running Configuration

When auto-QoS is enabled, the **auto qos** interface configuration commands and the generated global configuration are added to the running configuration.

The switch applies the auto-QoS-generated commands as if the commands were entered from the CLI. An existing user configuration can cause the application of the generated commands to fail or to be overridden by the generated commands. These actions may occur without warning. If all the generated commands are successfully applied, any user-entered configuration that was not overridden remains in the running configuration. Any user-entered configuration that was overridden can be retrieved by reloading the switch without saving the current configuration to memory. If the generated commands are not applied, the previous running configuration is restored.

Effects of Auto-Qos Compact on Running Configuration

If auto-QoS compact is enabled:

- Only the auto-QoS commands entered from the CLI are displayed in running-config.
- The generated global and interface configurations are hidden.
- When you save the configuration, only the auto-qos commands you have entered are saved (and not the hidden configuration).
- When you reload the switch, the system detects and re-executes the saved auto-QoS commands and the AutoQoS SRND4.0 compliant config-set is generated .

Note Do not make changes to the auto-QoS-generated commands when auto-QoS compact is enabled, because user-modifications are overridden when the switch reloads.

When auto-qos global compact is enabled:

- **show derived-config** command can be used to view hidden AQC derived commands.
- AQC commands will not be stored to memory. They will be regenerated every time the switch is reloaded.
- When compaction is enabled, auto-qos generated commands should not be modified .
- If the interface is configured with auto-QoS and if AQC needs to be disabled, auto-qos should be disabled at interface level first.

How to configure Auto-QoS

Configuring Auto-QoS

For optimum QoS performance, configure auto-QoS on all the devices in your network.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigE 1/0/1`	Specifies the port that is connected to a VoIP port, video device, or the uplink port that is connected to another trusted switch or router in the network interior, and enters the interface configuration mode.
Step 3	Depending on your auto-QoS configuration, use one of the following commands: • **auto qos voip** {**cisco-phone** | **cisco-softphone** | **trust**} • **auto qos video** {**cts** | **ip-camera** | **media-player**} • **auto qos classify** [**police**] • **auto qos trust** {**cos** | **dscp**} **Example:** `Device(config-if)# auto qos trust dscp`	The following commands enable auto-QoS for VoIP: • **auto qos voip cisco-phone**—If the port is connected to a Cisco IP Phone, the QoS labels of incoming packets are only trusted (conditional trust through CDP) when the telephone is detected. **Note** Do not configure the **auto qos voip cisco-phone** option for IP phones that support video. This option causes DSCP markings of video packets to get overwritten, because these packets do not have Expedited Forwarding priority, which results in these packets getting classified in the class-default class. • **auto qos voip cisco-softphone**—The port is connected to device running the Cisco SoftPhone feature. This command generates a QoS configuration for interfaces connected to PCs running the Cisco IP SoftPhone application and mark, as well as police traffic coming from such interfaces. Ports configured with this command are considered untrusted. • **auto qos voip trust**—The uplink port is connected to a trusted switch or router, and the VoIP traffic classification in the ingress packet is trusted.

	Command or Action	Purpose
		The following commands enable auto-QoS for the specified video device (system, camera, or media player): • **auto qos video cts**—A port connected to a Cisco Telepresence system. QoS labels of incoming packets are only trusted (conditional trust through CDP) when a Cisco TelePresence is detected. • **auto qos video ip-camera**—A port connected to a Cisco video surveillance camera. QoS labels of incoming packets are only trusted (conditional trust through CDP) when a Cisco camera is detected. • **auto qos video media-player**—A port connected to a CDP-capable Cisco digital media player. QoS labels of incoming packets are only trusted (conditional trust through CDP) when a digital media player is detected. The following command enables auto-QoS for classification: • **auto qos classify police**— This command generates a QoS configuration for untrusted interfaces. The configuration places a service-policy on the interface to classify traffic coming from untrusted desktops/devices and mark them accordingly. The service-policies generated do police. The following commands enable auto-QoS for trusted interfaces: • **auto qos trust cos**—Class of service. • **auto qos trust dscp**—Differentiated Services Code Point.
Step 4	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 5	**show auto qos interface** *interface-id* **Example:**	(Optional) Displays the auto-QoS command on the interface on which auto-QoS was enabled. Use the **show running-config** command to

	Command or Action	Purpose
	`Device# show auto qos interface HundredGigE 1/0/1`	display the auto-QoS configuration and user modifications.

Upgrading Auto-QoS

Before you begin

Prior to upgrading, you need to remove all auto-QoS configurations currently on the switch. This sample procedure describes that process.

After following this sample procedure, you must then reboot the switch with the new or upgraded software image and reconfigure auto-QoS.

Procedure

Step 1 **show auto qos**

Example:

```
Device# show auto qos

TwentyFiveGigE1/0/1
auto qos trust dscp

TwentyFiveGigE1/0/2
auto qos trust cos
```

In privileged EXEC mode, record all current auto QoS configurations by entering this command.

Step 2 **no auto qos trust cos**

Example:

```
Device(config-if)#no auto qos trust cos
```

In interface configuration mode, run the appropriate **no auto qos** command on each interface that has an auto QoS configuration.

Step 3 **show run | autoQoS**

Example:

```
Device# show running-config | i autoQos
```

Return to privileged EXEC mode, and record any remaining auto QoS maps class maps, policy maps, access lists, table maps, or other configurations by entering this command.

Step 4 **no policy-map** *policy-map_name*

Example:

```
Device(config)# no policy-map pmap_101
Device(config)# no class-map cmap_101
Device(config)# no ip access-list extended AutoQos-101
Device(config)# no table-map 101
Device(config)# no table-map policed-dscp
```

In global configuration mode, remove the QoS class maps, policy maps, access-lists, table maps, and any other auto QoS configurations by entering these commands:

- **no policy-map** *policy-map-name*
- **no class-map** *class-map-name*
- **no ip access-list extended** *Auto-QoS-x*
- **no table-map** *table-map-name*
- **no table-map policed-dscp**

Step 5 **show run | autoQoS**

Example:

```
Device#show running-config | i autoQos
```

Return to privileged EXEC mode, run this command again to ensure that no auto-QoS configuration or remaining parts of the auto-QoS configuration exists

Step 6 **show auto qos**

Example:

```
Device# show auto qos
```

Run this command to ensure that no auto-QoS configuration or remaining parts of the configuration exists.

Step 7 **write memory**

Example:

```
Device# write memory
```

Write the changes to the auto QoS configuration to NV memory by entering the **write memory** command.

What to do next

Reboot the switch with the new or upgraded software image.

After rebooting with the new or upgraded software image, re-configure auto-QoS for the appropriate switch interfaces as determined by running the **show auto qos** command described in step 1.

Note There is only one table-map for exceed and another table-map for violate markdown per switch or stack. If the switch already has a table-map under the exceed action, then the auto-qos policy cannot be applied.

Enabling Auto-Qos Compact

To enable auto-Qos compact, enter this command:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**auto qos global compact**
	Example:
	`Device(config)# auto qos global compact`
	`%Warning: This command will reset all autoqos generated configurations`
		Enables auto-Qos compact and generates (hidden) the global configurations for auto-QoS.
		You can then enter the auto-QoS command you want to configure in the interface configuration mode and the interface commands that the system generates are also hidden.
		To display the auto-QoS configuration that has been applied, use these the privileged EXEC commands:
		• **show derived-config**
		• **show policy-map**
		• **show access-list**
		• **show class-map**
		• **show table-map**
		• **show auto qos**
		• **show policy-map interface**
		• **show ip access-lists**

What to do next

To disable auto-QoS compact, remove auto-Qos instances from all interfaces by entering the **no** form of the corresponding auto-QoS commands and then enter the **no auto qos global compact** global configuration command.

Monitoring Auto-QoS

Table 1: Commands for Monitoring Auto-QoS

Command	Description
show auto qos [**interface** [*interface-id*]]	Displays the initial auto-QoS configuration. You can compare the **show auto qos** and the **show running-config** command output to identify the user-defined QoS settings.
show running-config	Displays information about the QoS configuration that might be affected by auto-QoS. You can compare the **show auto qos** and the **show running-config** command output to identify the user-defined QoS settings.
show derived-config	Displays the hidden **mls qos** command which get configured along with the running configs because of auto-qos template.

Troubleshooting Auto-QoS

To troubleshoot auto-QoS, use the **debug auto qos** privileged EXEC command. For more information, see the **debug auto qos** command in the command reference for this release.

To disable auto-QoS on a port, use the **no** form of the **auto qos** command interface configuration command, such as **no auto qos voip**. Only the auto-QoS-generated interface configuration commands for this port are removed. If this is the last port on which auto-QoS is enabled and you enter the **no auto qos voip** command, auto-QoS is considered disabled even though the auto-QoS-generated global configuration commands remain (to avoid disrupting traffic on other ports affected by the global configuration).

Configuration Examples for Auto-QoS

Example: auto qos trust cos

The following is an example of the **auto qos trust cos** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Cos-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)

- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigE1/0/2
Device(config-if)# auto qos trust cos
Device(config-if)# end
Device# show policy-map interface HundredGigE1/0/2

HundredGigE1/0/2

  Service-policy input: AutoQos-4.0-Trust-Dscp-Input-Policy

  Class-map: class-default (match-any)
  0 packets
  Match: any
  QoS Set
  dscp dscp table AutoQos-4.0-Trust-Dscp-Table

Service-policy output: AutoQos-4.0-Output-Policy

  queue stats for all priority classes:
  Queueing
  priority level 1

  (total drops) 0
  (bytes output) 0

Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
  0 packets
  Match: dscp cs4 (32) cs5 (40) ef (46)
  Match: cos 5
  Priority: 30% (30000000 kbps), burst bytes 750000000,

  Priority Level: 1

Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
  0 packets
  Match: dscp cs2 (16) cs3 (24) cs6 (48) cs7 (56)
  Match: cos 3
  Queueing

  queue-limit dscp 16 percent 80
  queue-limit dscp 24 percent 90
  queue-limit dscp 48 percent 100
  queue-limit dscp 56 percent 100
  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%

  queue-buffers ratio 10
```

```
Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
  0 packets
  Match: dscp af41 (34) af42 (36) af43 (38)
  Match: cos 4
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
  0 packets
  Match: dscp af21 (18) af22 (20) af23 (22)
  Match: cos 2
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  0 packets
  Match: dscp af11 (10) af12 (12) af13 (14)
  Match: cos 1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  0 packets
  Match: dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  0 packets
  Match: dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
  0 packets
  Match: any
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 25%
  queue-buffers ratio 25
```

Example: auto qos trust dscp

The following is an example of the **auto qos trust dscp** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Dscp-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigE1/0/2
Device(config-if)# auto qos trust dscp
Device(config-if)# end
Device#show policy-map interface HundredGigE1/0/2

HundredGigE1/0/2

  Service-policy input: AutoQos-4.0-Trust-Dscp-Input-Policy

    Class-map: class-default (match-any)
      0 packets
      Match: any
      QoS Set
        dscp dscp table AutoQos-4.0-Trust-Dscp-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 0

    Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
      0 packets
      Match:  dscp cs4 (32) cs5 (40) ef (46)
      Match: cos  5
      Priority: 30% (30000000 kbps), burst bytes 750000000,

      Priority Level: 1

    Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
      0 packets
```

```
  Match:  dscp cs2 (16) cs3 (24) cs6 (48) cs7 (56)
  Match: cos  3
  Queueing
  queue-limit dscp 16 percent 80
  queue-limit dscp 24 percent 90
  queue-limit dscp 48 percent 100
  queue-limit dscp 56 percent 100
  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%

  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
  0 packets
  Match:  dscp af41 (34) af42 (36) af43 (38)
  Match: cos  4
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
  0 packets
  Match:  dscp af21 (18) af22 (20) af23 (22)
  Match: cos  2
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  0 packets
  Match:  dscp af11 (10) af12 (12) af13 (14)
  Match: cos  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  0 packets
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  0 packets
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
```

```
    queue-buffers ratio 10

  Class-map: class-default (match-any)
    0 packets
    Match: any
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 25%
    queue-buffers ratio 25
```

Example: auto qos video cts

The following is an example of the **auto qos video cts** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Cos-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitEthernet1/0/2
Device(config-if)# auto qos video cts
Device(config-if)# end
Device# show policy-map interface HundredGigabitEthernet1/0/2

HundredGigabitEthernet1/0/2

  Service-policy input: AutoQos-4.0-Trust-Cos-Input-Policy

    Class-map: class-default (match-any)
      Match: any
      QoS Set
        cos cos table AutoQos-4.0-Trust-Cos-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
```

```
  Queueing
  priority level 1

  (total drops) 0
  (bytes output) 0

Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
  Match:  dscp cs4 (32) cs5 (40) ef (46)
  Match: cos  5
  Priority: 30% (300000 kbps), burst bytes 7500000,

  Priority Level: 1

Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
  Match:  dscp cs3 (24) cs6 (48) cs7 (56)
  Match: cos  3
  Queueing
  queue-limit dscp 16 percent 80
  queue-limit dscp 24 percent 90
  queue-limit dscp 48 percent 100

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%

  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
  Match:  dscp af41 (34) af42 (36) af43 (38)
  Match: cos  4
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
  Match:  dscp af21 (18) af22 (20) af23 (22)
  Match: cos  2
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  Match:  dscp af11 (10) af12 (12) af13 (14)
  Match: cos  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
```

```
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
    Match:  dscp af31 (26) af32 (28) af33 (30)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: class-default (match-any)
    Match: any
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 25%
    queue-buffers ratio 25
```

Example: auto qos video ip-camera

The following is an example of the **auto qos video ip-camera** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Dscp-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# auto qos video ip-camera
Device(config-if)# end
Device# show policy-map interface HundredGigabitE1/0/2

HundredGigabitE1/0/2

  Service-policy input: AutoQos-4.0-Trust-Dscp-Input-Policy

    Class-map: class-default (match-any)
      Match: any
```

```
    QoS Set
      dscp dscp table AutoQos-4.0-Trust-Dscp-Table

Service-policy output: AutoQos-4.0-Output-Policy

  queue stats for all priority classes:
    Queueing
    priority level 1

    (total drops) 0
    (bytes output) 0

  Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
    Match:  dscp cs4 (32) cs5 (40) ef (46)
    Match: cos  5
    Priority: 30% (300000 kbps), burst bytes 7500000,

    Priority Level: 1

  Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
    Match:  dscp cs3 (24) cs6 (48) cs7 (56)
    Match: cos  3
    Queueing
    queue-limit dscp 16 percent 80
    queue-limit dscp 24 percent 90
    queue-limit dscp 48 percent 100

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%

    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
    Match:  dscp af41 (34) af42 (36) af43 (38)
    Match: cos  4
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
    Match:  dscp af21 (18) af22 (20) af23 (22)
    Match: cos  2
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
    Match:  dscp af11 (10) af12 (12) af13 (14)
    Match: cos  1
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 4%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
```

```
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
  Match: any
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 25%
  queue-buffers ratio 25
```

Example: auto qos video media-player

The following is an example of the **auto qos video media-player** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Dscp-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# auto qos video media-player
```

```
Device(config-if)# end
Device# show policy-map interface HundredGigabitE1/0/2

HundredGigabitE1/0/2

  Service-policy input: AutoQos-4.0-Trust-Dscp-Input-Policy

    Class-map: class-default (match-any)
      Match: any
      QoS Set
        dscp dscp table AutoQos-4.0-Trust-Dscp-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 0

    Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
      Match:  dscp cs4 (32) cs5 (40) ef (46)
      Match: cos  5
      Priority: 30% (300000 kbps), burst bytes 7500000,

      Priority Level: 1

    Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
      Match:  dscp cs3 (24) cs6 (48) cs7 (56)
      Match: cos  3
      Queueing
      queue-limit dscp 16 percent 80
      queue-limit dscp 24 percent 90
      queue-limit dscp 48 percent 100

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%

      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
      Match:  dscp af41 (34) af42 (36) af43 (38)
      Match: cos  4
      Queueing

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%
      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
      Match:  dscp af21 (18) af22 (20) af23 (22)
      Match: cos  2
      Queueing

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%
      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
      Match:  dscp af11 (10) af12 (12) af13 (14)
```

```
  Match: cos  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
  Match: any
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 25%
  queue-buffers ratio 25
```

Example: auto qos voip trust

The following is an example of the **auto qos voip trust** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Cos-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)

- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitE1/0/3
Device(config-if)# auto qos voip trust
Device(config-if)# end
Device# show policy-map interface HundredGigabitE1/0/3

HundredGigabitE1/0/3

  Service-policy input: AutoQos-4.0-Trust-Cos-Input-Policy

    Class-map: class-default (match-any)
      Match: any
      QoS Set
        cos cos table AutoQos-4.0-Trust-Cos-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 0

    Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
      Match:  dscp cs4 (32) cs5 (40) ef (46)
      Match: cos  5
      Priority: 30% (300000 kbps), burst bytes 7500000,

      Priority Level: 1

    Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
      Match:  dscp cs3 (24) cs6 (48) cs7 (56)
      Match: cos  3
      Queueing
      queue-limit dscp 16 percent 80
      queue-limit dscp 24 percent 90
      queue-limit dscp 48 percent 100

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%

      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
      Match:  dscp af41 (34) af42 (36) af43 (38)
      Match: cos  4
      Queueing

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%
      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
      Match:  dscp af21 (18) af22 (20) af23 (22)
      Match: cos  2
      Queueing

      (total drops) 0
```

```
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
    Match:  dscp af11 (10) af12 (12) af13 (14)
    Match: cos  1
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 4%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
    Match:  dscp cs1 (8)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 1%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
    Match:  dscp af31 (26) af32 (28) af33 (30)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: class-default (match-any)
    Match: any
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 25%
    queue-buffers ratio 25
```

Example: auto qos voip cisco-phone

The following is an example of the **auto qos voip cisco-phone** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-CiscoPhone-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- AutoQos-4.0-Voip-Data-Class (match-any)
- AutoQos-4.0-Voip-Signal-Class (match-any)
- AutoQos-4.0-Default-Class (match-any)

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitE1/0/5
Device(config-if)# auto qos voip cisco-phone
Device(config-if)# end
Device# show policy-map interface HundredGigabitE1/0/5

 HundredGigabitE1/0/5

  Service-policy input: AutoQos-4.0-CiscoPhone-Input-Policy

    Class-map: AutoQos-4.0-Voip-Data-Class (match-any)
      Match: ip dscp ef (46)
      QoS Set
        ip dscp ef
      police:
          cir 128000 bps, bc 8000 bytes, be 8000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          set-dscp-transmit dscp table policed-dscp
        violated 0 bytes; actions:
          drop
        conformed 0000 bps, exceed 0000 bps, violate 0000 bps

    Class-map: AutoQos-4.0-Voip-Signal-Class (match-any)
      Match: ip dscp cs3 (24)
      QoS Set
        ip dscp cs3
      police:
          cir 32000 bps, bc 8000 bytes, be 8000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          set-dscp-transmit dscp table policed-dscp
        violated 0 bytes; actions:
          drop
        conformed 0000 bps, exceed 0000 bps, violate 0000 bps

    Class-map: AutoQos-4.0-Default-Class (match-any)
      Match: access-group name AutoQos-4.0-Acl-Default
      QoS Set
        dscp default
      police:
          cir 10000000 bps, bc 8000 bytes, be 8000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
```

```
        set-dscp-transmit dscp table policed-dscp
      violated 0 bytes; actions:
        drop
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: class-default (match-any)
    Match: any

Service-policy output: AutoQos-4.0-Output-Policy

  queue stats for all priority classes:
    Queueing
    priority level 1

    (total drops) 0
    (bytes output) 0

  Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
    Match:  dscp cs4 (32) cs5 (40) ef (46)
    Match: cos  5
    Priority: 30% (300000 kbps), burst bytes 7500000,

    Priority Level: 1

  Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
    Match:  dscp cs3 (24) cs6 (48) cs7 (56)
    Match: cos  3
    Queueing
    queue-limit dscp 16 percent 80
    queue-limit dscp 24 percent 90
    queue-limit dscp 48 percent 100

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%

    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
    Match:  dscp af41 (34) af42 (36) af43 (38)
    Match: cos  4
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
    Match:  dscp af21 (18) af22 (20) af23 (22)
    Match: cos  2
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
    Match:  dscp af11 (10) af12 (12) af13 (14)
    Match: cos  1
    Queueing

    (total drops) 0
```

```
    (bytes output) 0
    bandwidth remaining 4%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
    Match:  dscp cs1 (8)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 1%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
    Match:  dscp af31 (26) af32 (28) af33 (30)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: class-default (match-any)
    Match: any
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 25%
    queue-buffers ratio 25
```

Example: auto qos voip cisco-softphone

The following is an example of the **auto qos voip cisco-softphone** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-CiscoSoftPhone-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- AutoQos-4.0-Voip-Data-Class (match-any)
- AutoQos-4.0-Voip-Signal-Class (match-any)
- AutoQos-4.0-Multimedia-Conf-Class (match-any)
- AutoQos-4.0-Bulk-Data-Class (match-any)
- AutoQos-4.0-Transaction-Class (match-any)
- AutoQos-4.0-Scavanger-Class (match-any)
- AutoQos-4.0-Signaling-Class (match-any)
- AutoQos-4.0-Default-Class (match-any)

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigE1/0/20
Device(config-if)# auto qos voip cisco-softphone
Device(config-if)# end
Device# show policy-map interface HundredGigE1/0/20

 HundredGigE1/0/20

  Service-policy input: AutoQos-4.0-CiscoSoftPhone-Input-Policy

    Class-map: AutoQos-4.0-Voip-Data-Class (match-any)
      Match: ip dscp ef (46)
      QoS Set
        ip dscp ef
      police:
          cir 128000 bps, bc 8000 bytes, be 8000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          set-dscp-transmit dscp table policed-dscp
        violated 0 bytes; actions:
          drop
        conformed 0000 bps, exceed 0000 bps, violate 0000 bps

    Class-map: AutoQos-4.0-Voip-Signal-Class (match-any)
      Match: ip dscp cs3 (24)
      QoS Set
        ip dscp cs3
      police:
          cir 32000 bps, bc 8000 bytes, be 8000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          set-dscp-transmit dscp table policed-dscp
        violated 0 bytes; actions:
          drop
        conformed 0000 bps, exceed 0000 bps, violate 0000 bps

    Class-map: AutoQos-4.0-Multimedia-Conf-Class (match-any)
      Match: access-group name AutoQos-4.0-Acl-MultiEnhanced-Conf
      QoS Set
        dscp af41
      police:
          cir 5000000 bps, bc 8000 bytes, be 8000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
```

```
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Bulk-Data-Class (match-any)
  Match: access-group name AutoQos-4.0-Acl-Bulk-Data
  QoS Set
    dscp af11
  police:
      cir 10000000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Transaction-Class (match-any)
  Match: access-group name AutoQos-4.0-Acl-Transactional-Data
  QoS Set
    dscp af21
  police:
      cir 10000000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Scavanger-Class (match-any)
  Match: access-group name AutoQos-4.0-Acl-Scavanger
  QoS Set
    dscp cs1
  police:
      cir 10000000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Signaling-Class (match-any)
  Match: access-group name AutoQos-4.0-Acl-Signaling
  QoS Set
    dscp cs3
  police:
      cir 32000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Default-Class (match-any)
  Match: access-group name AutoQos-4.0-Acl-Default
  QoS Set
```

```
      dscp default
    police:
        cir 10000000 bps, bc 8000 bytes, be 8000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        set-dscp-transmit dscp table policed-dscp
      violated 0 bytes; actions:
        drop
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: class-default (match-any)
    Match: any

Service-policy output: AutoQos-4.0-Output-Policy

  queue stats for all priority classes:
    Queueing
    priority level 1

    (total drops) 0
    (bytes output) 0

  Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
    Match:  dscp cs4 (32) cs5 (40) ef (46)
    Match: cos  5
    Priority: 30% (300000 kbps), burst bytes 7500000,

    Priority Level: 1

  Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
    Match:  dscp cs3 (24) cs6 (48) cs7 (56)
    Match: cos  3
    Queueing
    queue-limit dscp 16 percent 80
    queue-limit dscp 24 percent 90
    queue-limit dscp 48 percent 100

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%

    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
    Match:  dscp af41 (34) af42 (36) af43 (38)
    Match: cos  4
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
    Match:  dscp af21 (18) af22 (20) af23 (22)
    Match: cos  2
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10
```

```
Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  Match:  dscp af11 (10) af12 (12) af13 (14)
  Match: cos  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
  Match: any
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 25%
  queue-buffers ratio 25
```

auto qos global compact

The following is an example of the **auto qos global compact** command.

```
Device# configure terminal
Device(config)# auto qos global compact
Device(config)# interface HundredGigE1/0/2
Device(config-if)# auto qos voip cisco-phone

Device# show auto qos
HundredGigE1/0/2
auto qos voip cisco-phone

Device# show running-config interface HundredGigE1/0/2
interface HundredGigE1/0/2
auto qos voip cisco-phone
end
```

Where to Go Next for Auto-QoS

Review the QoS documentation if you require any specific QoS changes to your auto-QoS configuration.

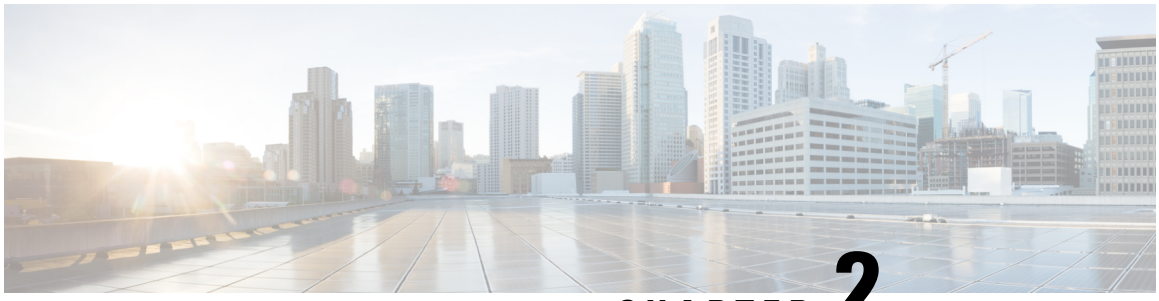
Feature History for Auto-QoS

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	Auto-QoS	Auto-QoS feature simplifies the deployment of QoS features. This feature determines the network design and enables QoS configurations so that the switch can prioritize different traffic flows.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 2

Configuring QoS

- Prerequisites for QoS, on page 31
- QoS Components, on page 32
- QoS Terminology, on page 32
- Information About QoS, on page 32
- QoS Implementation, on page 34
- QoS Wired Model, on page 38
- Classification, on page 39
- Ingress Port FIFO Parser, on page 46
- Policing, on page 48
- Marking, on page 49
- Traffic Conditioning, on page 50
- Queuing and Scheduling, on page 53
- Trust Behavior, on page 59
- Standard QoS Default Settings, on page 60
- Restrictions for QoS on Wired Targets, on page 61
- How to Configure QoS, on page 64
- Monitoring QoS, on page 103
- Configuration Examples for QoS, on page 103
- Where to Go Next, on page 114
- Additional References for QoS, on page 115
- Feature History for QoS, on page 115

Prerequisites for QoS

Before configuring standard Quality of Service (Qo)S, you must have a thorough understanding of these items:

- Standard QoS concepts.
- Classic Cisco IOS QoS.
- Modular QoS CLI (MQC).
- Understanding of QoS implementation.
- The types of applications used and the traffic patterns on your network.

- Traffic characteristics and needs of your network. For example, is the traffic on your network bursty? Do you need to reserve bandwidth for voice and video streams?
- Bandwidth requirements and speed of the network.
- Location of congestion points in the network.

QoS Components

Quality of service (QoS) consists of the following key components:

- Classification: Classification is the process of distinguishing one type of traffic from another based upon access control lists (ACLs), Differentiated Services Code Point (DSCP), Class of Service (CoS), and other factors.
- Marking and mutation: Marking is used on traffic to convey specific information to a downstream device in the network, or to carry information from one interface in a device to another. When traffic is marked, QoS operations on that traffic can be applied. This can be accomplished directly using the **set** command or through a table map, which takes input values and translates them directly to values on output.
- Shaping and policing: Shaping is the process of imposing a maximum rate of traffic, while regulating the traffic rate in such a way that downstream devices are not subjected to congestion. Shaping in the most common form is used to limit the traffic sent from a physical or logical interface. Policing is used to impose a maximum rate on a traffic class. If the rate is exceeded, then a specific action is taken as soon as the event occurs.
- Queuing: Queuing is used to prevent traffic congestion. Traffic is sent to specific queues for servicing and scheduling based upon bandwidth allocation. Traffic is then scheduled or sent out through the port.
- Bandwidth: Bandwidth allocation determines the available capacity for traffic that is subject to QoS policies.
- Trust: Trust enables traffic to pass through the device, and the Differentiated Services Code Point (DSCP), precedence, or CoS values coming in from the end points are retained in the absence of any explicit policy configuration.

QoS Terminology

The following terms are used interchangeably in this QoS configuration guide:

- Upstream (direction towards the device) is the same as ingress.
- Downstream (direction from the device) is the same as egress.

Information About QoS

By configuring the quality of service (QoS), you can provide preferential treatment to specific types of traffic at the expense of other traffic types. Without QoS, the device offers best-effort service for each packet,

regardless of the packet contents or size. The device sends the packets without any assurance of reliability, delay bounds, or throughput.

The following are specific features provided by QoS:

- Low latency
- Bandwidth guarantee
- Buffering capabilities and dropping disciplines
- Traffic policing
- Enables the changing of the attribute of the frame or packet header
- Relative services

Modular QoS CLI

With the device, QoS features are enabled through the Modular QoS command-line interface (MQC). The MQC is a command-line interface (CLI) structure that allows you to create traffic 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, while 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.

Supported QoS Features for Wired Access

The following table describes the supported QoS features for wired access.

Table 2: Supported QoS Features for Wired Access

Feature	Description
Supported targets	• Gigabit Ethernet • 10-Gigabit Ethernet • 40-Gigabit Ethernet • VLAN
Configuration sequence	QoS policy installed using the **service-policy** command.
Supported number of queues at port level	Up to eight queues supported on a port.

Feature	Description

Supported classification mechanism	• DSCP
	• IP precedence
	• CoS
	• QoS-group
	• ACL membership including:
	  • IPv4 ACLs
	  • IPv6 ACLS
	  • MAC ACLs
	• VLAN

Hierarchical QoS

The device supports hierarchical QoS (HQoS). HQoS allows you to perform:

- Hierarchical classification— Traffic classification is based upon other classes.
- Hierarchical policing—The process of having the policing configuration at multiple levels in a hierarchical policy.
- Hierarchical shaping—Shaping can also be configured at multiple levels in the hierarchy.

Note Hierarchical shaping is only supported for the port shaper, where for the parent you only have a configuration for the class default, and the only action for the class default is shaping.

QoS Implementation

Typically, networks operate on a best-effort delivery basis, which means that all traffic has equal priority and an equal chance of being delivered in a timely manner. When congestion occurs, all traffic has an equal chance of being dropped.

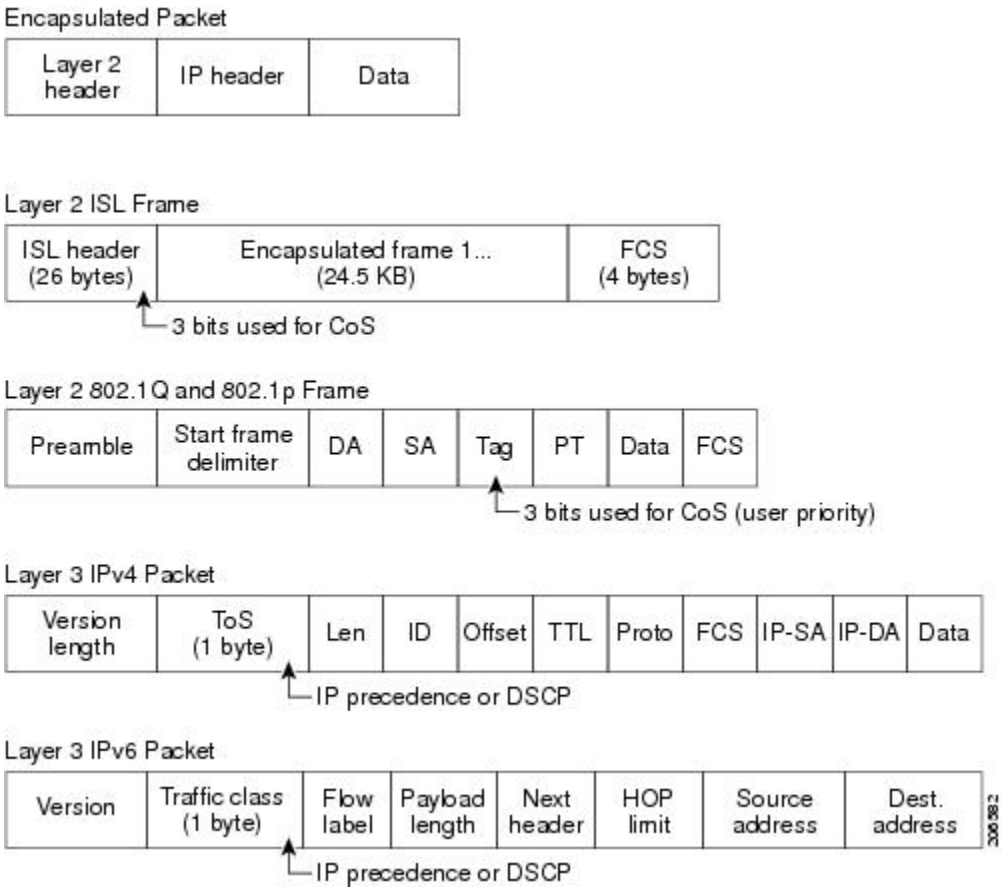
When you configure the QoS feature, you can select specific network traffic, prioritize it according to its relative importance, and use congestion-management and congestion-avoidance techniques to provide preferential treatment. Implementing QoS in your network makes network performance more predictable and bandwidth utilization more effective.

The QoS implementation is based on the Differentiated Services (Diff-Serv) architecture, a standard from the Internet Engineering Task Force (IETF). This architecture specifies that each packet is classified upon entry into the network.

The classification is carried in the IP packet header, using 6 bits from the deprecated IP type of service (ToS) field to carry the classification (*class*) information. Classification can also be carried in the Layer 2 frame.

The special bits in the Layer 2 frame or a Layer 3 packet are shown in the following figure:

Figure 1: QoS Classification Layers in Frames and Packets



Layer 2 Frame Prioritization Bits

Layer 2 Inter-Switch Link (ISL) frame headers have a 1-byte User field that carries an IEEE 802.1p class of service (CoS) value in the three least-significant bits. On ports configured as Layer 2 ISL trunks, all traffic is in ISL frames.

Layer 2 802.1Q frame headers have a 2-byte Tag Control Information field that carries the CoS value in the three most-significant bits, which are called the User Priority bits. On ports configured as Layer 2 802.1Q trunks, all traffic is in 802.1Q frames except for traffic in the native VLAN.

Other frame types cannot carry Layer 2 CoS values.

Layer 2 CoS values range from 0 for low priority to 7 for high priority.

Layer 3 Packet Prioritization Bits

Layer 3 IP packets can carry either an IP precedence value or a Differentiated Services Code Point (DSCP) value. QoS supports the use of either value because DSCP values are backward-compatible with IP precedence values.

IP precedence values range from 0 to 7. DSCP values range from 0 to 63.

End-to-End QoS Solution Using Classification

All switches and routers that access the Internet rely on the class information to provide the same forwarding treatment to packets with the same class information and different treatment to packets with different class information. The class information in the packet can be assigned by end hosts or by switches or routers along the way, based on a configured policy, detailed examination of the packet, or both. Detailed examination of the packet is expected to occur closer to the edge of the network, so that the core switches and routers are not overloaded with this task.

Switches and routers along the path can use the class information to limit the amount of resources allocated per traffic class. The behavior of an individual device when handling traffic in the Diff-Serv architecture is called per-hop behavior. If all devices along a path provide a consistent per-hop behavior, you can construct an end-to-end QoS solution.

Implementing QoS in your network can be a simple task or complex task and depends on the QoS features offered by your internetworking devices, the traffic types and patterns in your network, and the granularity of control that you need over incoming and outgoing traffic.

Packet Classification

Packet classification is the process of identifying a packet as belonging to one of several classes in a defined policy, based on certain criteria. The Modular QoS CLI (MQC) is a policy-class based language. The policy class language is used to define the following:

- Class-map template with one or several match criteria
- Policy-map template with one or several classes associated to the policy map

The policy map template is then associated to one or several interfaces on the device.

Packet classification is the process of identifying a packet as belonging to one of the classes defined in the policy map. The process of classification will exit when the packet being processed matches a specific filter in a class. This is referred to as first-match exit. If a packet matches multiple classes in a policy, irrespective of the order of classes in the policy map, it would still exit the classification process after matching the first class.

If a packet does not match any of the classes in the policy, it would be classified into the default class in the policy. Every policy map has a default class, which is a system-defined class to match packets that do not match any of the user-defined classes.

Packet classification can be categorized into the following types:

- Classification based on information that is propagated with the packet
- Classification based on information that is device specific
- Hierarchical classification

Classification Based on Information that is Propagated with a Packet

Classification that is based on information that is part of the packet and propagated either end-to-end or between hops, typically includes the following:

- Classification based on Layer 3 or 4 headers
- Classification based on Layer 2 information

Classification Based on Layer 3 or Layer 4 Header

This is the most common deployment scenario. Numerous fields in the Layer 3 and Layer 4 headers can be used for packet classification.

At the most granular level, this classification methodology can be used to match an entire flow. For this deployment type, an access control list (ACLs) can be used. ACLs can also be used to match based on various subsets of the flow (for example, source IP address only, or destination IP address only, or a combination of both).

Classification can also be done based on the precedence or DSCP values in the IP header. The IP precedence field is used to indicate the relative priority with which a particular packet needs to be handled. It is made up of three bits in the IP header's type of service (ToS) byte.

The following table shows the different IP precedence bit values and their names.

Table 3: IP Precedence Values and Names

IP Precedence Value	IP Precedence Bits	IP Precedence Names
0	000	Routine
1	001	Priority
2	010	Immediate
3	011	Flash
4	100	Flash Override
5	101	Critical
6	110	Internetwork control
7	111	Network control

Note All routing control traffic in the network uses IP precedence value 6 by default. IP precedence value 7 also is reserved for network control traffic. Therefore, the use of IP precedence values 6 and 7 is not recommended for user traffic.

The DSCP field is made up of 6 bits in the IP header and is being standardized by the Internet Engineering Task Force (IETF) Differentiated Services Working Group. The original ToS byte contained the DSCP bits has been renamed the DSCP byte. The DSCP field is part of the IP header, similar to IP precedence. The

DSCP field is a super set of the IP precedence field. Therefore, the DSCP field is used and is set in ways similar to what was described with respect to IP precedence.

Note The DSCP field definition is backward-compatible with the IP precedence values.

Some fields in Layer 2 header can also be set uing a policy.

Classification Based on Layer 2 Header

A variety of methods can be used to perform classification based on the Layer 2 header information. The most common methods include the following:

- MAC address-based classification (only for access groups)—Classification is based upon the source MAC address (for policies in the input direction) and destination MAC address (for policies in the output direction).
- Class-of-Service—Classification is based on the 3 bits in the Layer 2 header based on the IEEE 802.1p standard. This usually maps to the ToS byte in the IP header.
- VLAN ID—Classification is based on the VLAN ID of the packet.

Note Some of these fields in the Layer 2 header can also be set using a policy.

Classification Based on Information that is Device Specific

The device also provides classification mechanisms that are available where classification is not based on information in the packet header or payload.

At times you might be required to aggregate traffic coming from multiple input interfaces into a specific class in the output interface. For example, multiple customer edge routers might be going into the same access device on different interfaces. The service provider might want to police all the aggregate voice traffic going into the core to a specific rate. However, the voice traffic coming in from the different customers could have a different ToS settings. QoS group-based classification is a feature that is useful in these scenarios.

Policies configured on the input interfaces set the QoS group to a specific value, which can then be used to classify packets in the policy enabled on output interface.

The QoS group is a field in the packet data structure internal to the device. It is important to note that a QoS group is an internal label to the device and is not part of the packet header.

QoS Wired Model

To implement QoS, the device must perform the following tasks:

- Traffic classification—Distinguishes packets or flows from one another.
- Traffic marking and policing—Assigns a label to indicate the given quality of service as the packets move through the device, and then make the packets comply with the configured resource usage limits.
- Queuing and scheduling—Provides different treatment in all situations where resource contention exists.

- Shaping—Ensures that traffic sent from the device meets a specific traffic profile.

Ingress Port Activity

The following activities occur at the ingress port of a device:

- Classification: Classifying a distinct path for a packet by associating it with a QoS label. For example, the device maps the CoS or DSCP in the packet to a QoS label to distinguish one type of traffic from another. The QoS label that is generated identifies all future QoS actions to be performed on this packet.
- Policing: Policing determines whether a packet is in or out of profile by comparing the rate of the incoming traffic to the configured policer. The policer limits the bandwidth consumed by a flow of traffic. The result is passed to the marker.
- Marking: Marking evaluates the policer and configuration information for the action to be taken when a packet is out of profile and determines what to do with the packet (pass through a packet without modification, mark down the QoS label in the packet, or drop the packet).

Egress Port Activity

The following activities occur at the egress port of the device:

- Policing—Policing determines whether a packet is in or out of profile by comparing the rate of the incoming traffic to the configured policer. The policer limits the bandwidth consumed by a flow of traffic. The result is passed to the marker.
- Marking—Marking evaluates the policer and configuration information for the action to be taken when a packet is out of profile and determines what to do with the packet (pass through a packet without modification, mark down the QoS label in the packet, or drop the packet).
- Queuing—Queuing evaluates the QoS packet label and the corresponding DSCP or CoS value before selecting which of the egress queues to use. Because congestion can occur when multiple ingress ports simultaneously send data to an egress port, Weighted Tail Drop (WTD) differentiates traffic classes and subjects the packets to different thresholds based on the QoS label. If the threshold is exceeded, the packet is dropped.

Classification

Classification is the process of distinguishing one kind of traffic from another by examining the fields in the packet. Classification is enabled only if QoS is enabled on the device. By default, QoS is enabled on the device.

During classification, the device performs a lookup and assigns a QoS label to the packet. The QoS label identifies all QoS actions to be performed on the packet and from which queue the packet is sent.

Access Control Lists

You can use IP standard, IP extended, or Layer 2 MAC ACLs to define a group of packets with the same characteristics (class). You can also classify IP traffic based on IPv6 ACLs.

In the QoS context, the permit and deny actions in the access control entries (ACEs) have different meanings from security ACLs:

- If a match with a permit action is encountered (first-match principle), the specified QoS-related action is taken.
- If a match with a deny action is encountered, the ACL being processed is skipped, and the next ACL is processed.
- If no match with a permit action is encountered and all the ACEs have been examined, no QoS processing occurs on the packet, and the device offers best-effort service to the packet.
- If multiple ACLs are configured on a port, the lookup stops after the packet matches the first ACL with a permit action, and QoS processing begins.

Note When creating an access list, note that by default the end of the access list contains an implicit deny statement for everything if it did not find a match before reaching the end.

After a traffic class has been defined with the ACL, you can attach a policy to it. A policy might contain multiple classes with actions specified for each one of them. A policy might include commands to classify the class as a particular aggregate (for example, assign a DSCP) or rate-limit the class. This policy is then attached to a particular port on which it becomes effective.

You implement IP ACLs to classify IP traffic by using the **access-list** global configuration command; you implement Layer 2 MAC ACLs to classify non-IP traffic by using the **mac access-list extended** global configuration command.

Class Maps

A class map is a mechanism that you use to name a specific traffic flow (or class) and isolate it from all other traffic. The class map defines the criteria used to match against a specific traffic flow to further classify it. The criteria can include matching the access group defined by the ACL or matching a specific list of DSCP or IP precedence values or CoS values. If you have more than one type of traffic that you want to classify, you can create another class map and use a different name. After a packet is matched against the class-map criteria, you further classify it through the use of a policy map.

Note You cannot configure IPv4 and IPv6 classification criteria simultaneously in the same class-map. However, they can be configured in different class-maps in the same policy.

You create a class map by using the **class-map** global configuration command or the **class** policy-map configuration command. You should use the **class-map** command when the map is shared among mulitple policies. When you enter the **class-map** command, the device enters the class-map configuration mode.

You can create a default class by using the **class class-default** policy-map configuration command. The default class is system-defined and cannot be configured. Unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as default traffic.

Time-to-Live Classification

You can classify packets based on the ACL map. You can set Time-to_live (TTL) as a criterion in the ACL list and perform a TTL check on the incoming packet. The access control entry is used to check the IPv4 TTL to match the value on the incoming packet. The classified packet is either marked or policed based on the policy-map action. Queueing cannot be configured on this classification.

The following is an example of TTL classification:

```
policy-map TTL_MATCH
 class IPV4_TTL
  police rate 6000000000
  set dscp af23

ip access-list extended IPV4_TTL
 permit ip any any ttl eq 100
 permit tcp any any ttl ne 150

!
Device#show run class-map IPV4_TTL
class-map match-all IPV4_TTL
 match access-group name IPV4_TTL
!

Device#show policy-map interface hun1/0/47

HundredGigE1/0/47

  Service-policy output: TTL_MATCH

    Class-map: IPV4_TTL (match-all)
    553567424 packets
    Match: access-group name IPV4_TTL
    police:
    rate 6000000000 bps, burst 187500000 bytes
    conformed 22983406600 bytes; actions:
    transmit
    exceeded 32375773000 bytes; actions:
    drop
    conformed 588922000 bps, exceeded 830894000 bps
    QoS Set
    dscp af23

    Class-map: class-default (match-any)
    2184433710 packets
    Match: any
```

Layer 3 Packet Length Classification

This feature provides the capability of matching and classifying traffic on the basis of the Layer 3 packet length in the IP header. The Layer 3 packet length is the IP datagram length plus the IP header length. You can set the packet length as a matching criterion in the class policy-map, to match the value on the incoming packet. The classified packet is either marked or policed based on the policy-map action. This feature does not work on IPv6 packets.

The following is an example of Layer 3 packet length classification:

```
Service-policy output: PACKET_MATCH1

Class-map: class-default (match-any)
 16281588 packets
```

```
  Match: any

Service-policy : L3_MATCH

Class-map: PACKET_LENGTH_1 (match-any)
  9910510 packets
  Match: packet length 7582
  Match: packet length 5000
  QoS Set
  dscp cs2
  police:
 rate 3 %
 rate 1200000000 bps, burst 37500000 bytes
 conformed 10000 bytes; actions:
   transmit
 exceeded 112121 bytes; actions:
   drop
 conformed 500 bps, exceeded 3434 bps

Class-map: PACKET_LENGTH_2 (match-all)
 6371042 packets
  Match: dscp cs4 (32)
  Match: packet length 7759
  police:
 rate 12000000000 bps, burst 375000000 bytes
 conformed 44545 bytes; actions:
 transmit
 exceeded 34343 bytes; actions:
 drop
 conformed 1211 bps, exceeded 11211 bps

Class-map: class-default (match-any)
 36 packets
  Match: any
 QoS Set
   precedence 3
Device#

class-map match-any PACKET_LENGTH_1
match packet length min 7582 max 7582
match packet length min 5000 max 5000

class-map match-all PACKET_LENGTH_2
match dscp cs4
match packet length min 7759 max 7759
```

Layer 2 SRC-Miss or DST-Miss Classification

Traffic can be classified for a missing MAC address in the MAC address table, for source MAC address or destination MAC address. Policy-map with L2-Miss classification can be applied on layer 2 interfaces, in the ingress direction. Policing, marking or remarking actions can be applied with this classification. L2-Miss classification cannot be applied on layer 3 interfaces. Queueing cannot be configured on this classification.

The following is an example of L2-Miss classification:

```
Device #show run class-map DST-MISS
    class-map match-any DST-MISS
    match l2 dst-mac miss

Device #show run class-map SRC-MISS
    class-map match-all SRC-MISS
    match l2 src-mac miss
```

```
Device #show policy-map L2-MISS
Policy Map L2-MISS
    Class DST-MISS
        set dscp af22
    police cir percent 10
        conform-action transmit
        exceed-action drop
    Class SRC-MISS
        set precedence 1
    police rate percent 20
        conform-action transmit
        exceed-action drop
!
end

Device#
```

Policy Maps

A policy map specifies which traffic class to act on. Actions can include the following:

- Setting a specific DSCP or IP precedence value in the traffic class
- Setting a CoS value in the traffic class
- Setting a QoS group
- Setting a wireless LAN (WLAN) value in the traffic class
- Specifying the traffic bandwidth limitations and the action to take when the traffic is out of profile

Before a policy map can be effective, you must attach it to a port.

You create and name a policy map using the **policy-map** global configuration command. When you enter this command, the device enters the policy-map configuration mode. In this mode, you specify the actions to take on a specific traffic class by using the **class** or **set** policy-map configuration and policy-map class configuration commands.

The policy map can also be configured using the **police** and **bandwidth** policy-map class configuration commands, which define the policer, the bandwidth limitations of the traffic, and the action to take if the limits are exceeded. In addition, the policy-map can further be configured using the **priority** policy-map class configuration command, to schedule priority for the class or the queuing policy-map class configuration commands, **queue-buffers** and **queue-limit**.

To enable the policy map, you attach it to a port by using the **service-policy** interface configuration command.

Note You cannot configure both **priority** and **set** for a policy map. If both these commands are configured for a policy map, and when the policy map is applied to an interface, error messages are displayed. The following example shows this restriction:

```
Device# configure terminal
Device(config)# class-map cmap
Device(config-cmap)# exit
Device(config)# class-map classmap1
Device(config-cmap)# exit
Device(config)# policy-map pmap
Device(config-pmap)# class cmap
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# exit
Device(config-pmap)# class classmap1
Device(config-pmap-c)# set dscp 10
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/2
Device(config-if)# service-policy output pmap

Non-queuing action only is unsupported in a queuing policy!!!
%QOS-6-POLICY_INST_FAILED:
Service policy installation failed
```

Policy Map on Physical Port

You can configure a nonhierarchical policy map on a physical port that specifies which traffic class to act on. Actions can include setting a specific DSCP or IP precedence or CoS values in the traffic class, specifying the traffic bandwidth limitations for each matched traffic class (policer), and taking action when the traffic is out of profile (marking).

A policy map also has these characteristics:

- A policy map can contain multiple class statements, each with different match criteria and policers.
- A policy map can contain a predefined default traffic class explicitly placed at the end of the map.

  When you configure a default traffic class by using the **class class-default** policy-map configuration command, unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as the default traffic class (**class-default**).
- A separate policy-map class can exist for each type of traffic received through a port.

Policy Map on VLAN

The device supports a VLAN QoS feature that allows the user to perform QoS treatment at the VLAN level (classification and QoS actions) using the incoming frame's VLAN information. In VLAN-based QoS, a service policy is applied to an SVI interface. All physical interfaces belonging to a VLAN policy map then need to be programmed to refer to the VLAN-based policy maps instead of the port-based policy map.

Although the policy map is applied to the VLAN SVI, only marking or remarking actions can be performed on a per-port basis. You cannot configure the policer to take account of the sum of traffic from a number of physical ports. Each port needs to have a separate policer governing the traffic coming into that port.

QoS Profile

The device uses Ternary Content Addressable Memory (TCAM) to store classification rules. To optimize the usage of TCAM resources, use the QoS profile to turn off some of the lesser used features and turn them on when required.

With the **qos profile** {**default** | **extended**} command, you can select the required classification feature set. **default** keyword loads only the common classification features. **extended** keyword loads the complete classification feature set (but with a reduced scale) that are available for the device. By default, only the commonly used classification features are set on the device.

qos profile extended enables TCP flag along with the common classification features.

You can verify the QoS profile that is configured on the device using the **show platform software fed active qos profile** command.

Example

```
device# show platform software fed active qos profile
Using default - Common Classification Features
```

Security Group Classification

Security Group classification includes both source and destination groups, which are specified by source security group tag (SGT) and destination security group tag (DGT) respectively.

The objective of SGT QoS classification is to leverage user groups to increase policy granularity such that the policy isn't only application-aware but also provides some level of differentiated service based on the user identity (or the group of users to which they belong).

Egress QoS classification based on SGT or DGT isn't supported.

SGT Based QoS

The SGT based QoS feature provides a special treatment for a class of traffic that is based on the QoS policies and actions, for a defined user group or device. This feature enables you to assign multiple QoS policies to an application or traffic type that is initiated by different user groups. Each user group is defined by a unique SGT value and can support MQC-based QoS configuration.

The SGT based QoS feature is applicable to both the user group and the device-based QoS service levels for SGT-DGT-based packet classification. It can also potentially support defining of user groups based on contextual information for QoS policy prioritization.

Sharing DGID with SGACL

Due to resource limitations, only 4096 security group destination tags (DGTs) are supported. Classification based on DGT is achieved through a security destination tag ID known as DGID. DGID is a global resource and is shared with SGACL. DGID allocation is done on a first-come-first-serve basis. On a device, at startup, SGACL configuration is applied before QoS policy configuration. Hence DGID is first allocated for SGACL and then for QoS policy.

The **show platform software fed sw active sgacl detail** command displays the DGT to DGID mapping.

Example

```
device# show platform software fed active sgacl detail

Global Enforcement: On
*Refcnt: for the non-SGACL feature
========================= DGID Table ====================================
SGT/Refcnt     DGT     DGID   hash  test_cell monitor   permitted  denied
=========================================================================
*/1            24        1     24
 24            24        1     24       Off     Off        0           0
```

Restrictions for SGT Based QoS

The following are the limitations of the SGT based QoS feature:

- SGT based QoS isn't supported on tunnel interfaces.
- Only 4096 security destination tags and 65539 security source tags are supported.
- SGT based policy can only be attached to the input direction of an interface.

Restrictions for an Upgrade or Downgrade

- For an upgrade from an earlier release to Cisco IOS XE Release 16.12.x and later, the maximum supported DGID is 256. Reload the switch to overcome this issue.
- For a downgrade from Cisco IOS XE Release 17.1.x to IOS XE 16.12.x releases, the allocated DGID is displayed as 4096; but only 256 DGIDs are supported. Reload the switch to overcome this issue.
- An In-Service Software Upgrade (ISSU) fails if the tcp flag is set in a policy. To do an ISSU, first remove the tcp flag configuration.
- If a policy-map, which is attached to an interface, classifies traffic based on tcp flag, an ISSU upgrade fails. To do an ISSU, either detach the policy-map from the interface or remove the tcp flag classification.

Ingress Port FIFO Parser

Ingress Port FIFO (IPF) parses incoming network traffic to classify frames into different priorities levels. It derives the traffic class from different packet formats. For example, the traffic class can be derived from the Differentiated Services Code Point (DSCP) for IP packets, or, Class of Service (CoS) for dot1q tag packets. These traffic classes are further mapped to priority levels, which are used to take drop decisions, in case of congestion.

The IPF parser, can be used in the global mode and in the isolation mode (high and low priority configuration at port level). By default it is in the isolation mode. In the isolation mode, priority differentiation is made at the port level rather than the system level.

To configure the IPF parser in the global mode, use the following command:

```
configure port-ingress-fifo mode global
```

The following are the examples of **show** commands to see the traffic class to priority mappings:

```
Device# show platform hardware fed active qos ipf interface twentyFiveGigE 1/0/1 cos-map
IPF cos to traffic class map for Interface [cos : traffic-class]:
------------------------------------------------------------------------
0 : 0           1 : 1           2 : 2           3 : 3
4 : 4           5 : 5           6 : 6           7 : 7
8 : 4           9 : 4           10 : 4          11 : 4
12 : 4          13 : 4          14 : 4          15 : 4
```

```
Device# show platform hardware fed active qos ipf interface twentyFiveGigE 1/0/1 dscp-map
IPF dscp to traffic class map for Interface [dscp : traffic-class]:
---------------------------------------------------------------
0 : 0           1 : 0           2 : 0           3 : 0
4 : 0           5 : 0           6 : 0           7 : 0
8 : 1           9 : 1           10 : 1          11 : 1
12 : 1          13 : 1          14 : 1          15 : 1
16 : 2          17 : 4          18 : 4          19 : 4
20 : 4          21 : 4          22 : 4          23 : 4
24 : 3          25 : 4          26 : 4          27 : 4
28 : 4          29 : 4          30 : 4          31 : 4
32 : 4          33 : 4          34 : 4          35 : 4
36 : 4          37 : 4          38 : 4          39 : 4
40 : 4          41 : 4          42 : 4          43 : 4
44 : 4          45 : 4          46 : 5          47 : 4
48 : 6          49 : 4          50 : 4          51 : 4
52 : 4          53 : 4          54 : 4          55 : 4
56 : 7          57 : 4          58 : 4          59 : 4
60 : 4          61 : 4          62 : 4          63 : 4
```

```
Device#show platform hardware fed active qos ipf interface twentyFiveGigE 1/0/1 exp-map
IPF exp to traffic class map for Interface [exp : traffic-class]:
---------------------------------------------------------------
0 : 0           1 : 1           2 : 2           3 : 3
4 : 4           5 : 5           6 : 6           7 : 7
```

```
Device#show platform hardware qos ipf interface twentyFiveGigE 1/0/1 ipf-parse-cfg
IPF configration for Interface:
-------------------------------
Port Trust:           Enabled
Default TC:           0
Dscp based parsing:   Disabled
Exp based parsing:    Disabled
Fdcos based parsing:  Enabled
cos based parsing:    Disabled
```

```
Device#show platform hardware fed active qos ipf tc-to-pri asic 0
IPF traffic class to priority for[Asic:Core:TlaInst]::[0:0:0]
-------------------------------------------------------------
Priority                Traffic Classes
--------                ---------------
Low Pri :               0  1  4
High Pri:               2  3  5  6  7
IPF traffic class to priority for[Asic:Core:TlaInst]::[0:0:1]
-------------------------------------------------------------
Priority                Traffic Classes
--------                ---------------
Low Pri :               0  1  4
High Pri:               2  3  5  6  7
```

Statistics show command:

```
Device#show platform hardware fed active qos ipf statistics asic 0
Ipf Statistics:[Asic|Core|Tla] : [0 | 0 | 0] - Global Mode
-----------------------------------------------
Ipf misc packet drops:                          0
Ipf Drop Statistics
-------------------
```

```
low pri Frames drop:                        0
low pri mop Frames drop:                    0
high pri Frames drop:                       0
almost full Frames drop:                    0
RCP Frames drop:                            0

Ipf Statistics:[Asic|Core|Tla] : [0 | 0 | 1] - Global Mode
-----------------------------------------------
Ipf misc packet drops:                              0
Ipf Drop Statistics
-------------------
low pri Frames drop:                        0
low pri mop Frames drop:                    0
high pri Frames drop:                       0
almost full Frames drop:                    0
RCP Frames drop:                            0
```

Policing

After a packet is classified and has a DSCP-based, CoS-based, or QoS-group label assigned to it, the policing and marking process can begin.

Policing involves creating a policer that specifies the bandwidth limits for the traffic. Packets that exceed the limits are *out of profile* or *nonconforming*. Each policer decides on a packet-by-packet basis whether the packet is in or out of profile and specifies the actions on the packet. These actions, carried out by the marker, include passing through the packet without modification, dropping the packet, or modifying (marking down) the assigned DSCP or CoS value of the packet and allowing the packet to pass through.

To avoid out-of-order packets, both conform and nonconforming traffic typically exit the same queue.

Note All traffic, regardless of whether it is bridged or routed, is subjected to a policer, if one is configured. As a result, bridged packets might be dropped or might have their DSCP or CoS fields modified when they are policed and marked.

You can only configure policing on a physical port.

After you configure the policy map and policing actions, attach the policy-map to an ingress or egress port by using the **service-policy** interface configuration command.

Token-Bucket Algorithm

Policing uses a token-bucket algorithm. As each frame is received by the device, a token is added to the bucket. The bucket has a hole in it and leaks at a rate that you specify as the average traffic rate in bits per second. Each time a token is added to the bucket, the device verifies that there is enough room in the bucket. If there is not enough room, the packet is marked as nonconforming, and the specified policer action is taken (dropped or marked down).

How quickly the bucket fills is a function of the bucket depth (burst-byte), the rate at which the tokens are removed (rate-bps), and the duration of the burst above the average rate. The size of the bucket imposes an upper limit on the burst length and limits the number of frames that can be transmitted back-to-back. If the burst is short, the bucket does not overflow, and no action is taken against the traffic flow. However, if a burst is long and at a higher rate, the bucket overflows, and the policing actions are taken against the frames in that burst.

You configure the bucket depth (the maximum burst that is tolerated before the bucket overflows) by using the burst-byte option of the **police** policy-map class configuration command. You configure how fast (the average rate) that the tokens are removed from the bucket by using the rate option of the **police** policy-map class configuration command.

Marking

Marking is used to convey specific information to a downstream device in the network, or to carry information from one interface in a device to another.

Marking can be used to set certain field/bits in the packet headers, or marking can also be used to set certain fields in the packet structure that is internal to the device. Additionally, the marking feature can be used to define mapping between fields. The following marking methods are available for QoS:

- Packet header
- Device specific information
- Table maps

Packet Header Marking

Marking on fields in the packet header can be classified into two general categories:

- IPv4/v6 header bit marking
- Layer 2 header bit marking

The marking feature at the IP level is used to set the precedence or the DSCP in the IP header to a specific value to get a specific per-hop behavior at the downstream device (switch or router), or it can also be used to aggregate traffic from different input interfaces into a single class in the output interface. The functionality is currently supported on both the IPv4 and IPv6 headers.

Marking in the Layer 2 headers is typically used to influence dropping behavior in the downstream devices (switch or router). It works in tandem with the match on the Layer 2 headers. The bits in the Layer 2 header that can be set using a policy map are class of service.

Switch-Specific Information Marking

This form of marking includes marking of fields in the packet data structure that are not part of the packets header, so that the marking can be used later in the data path. This is not propagated between the switches. Marking of QoS group falls into this category. This form of marking is only supported in policies that are enabled on the input interfaces. The corresponding matching mechanism can be enabled on the output interfaces on the same switch and an appropriate QoS action can be applied.

Table Map Marking

Table map marking enables the mapping and conversion from one field to another using a conversion table. This conversion table is called a table map.

Depending upon the table map attached to an interface, CoS, DSCP, and Precedence values of the packet are rewritten. The device allows configuring both ingress table map policies and egress table map policies.

As an example, a table map can be used to map the Layer 2 CoS setting to a precedence value in Layer 3. This feature enables combining multiple **set** commands into a single table, which indicates the method to perform the mapping. This table can be referenced in multiple policies, or multiple times in the same policy.

A table map-based policy supports the following capabilities:

- Mutation—You can have a table map that maps from one DSCP value set to another DSCP value set, and this can be attached to an egress port.
- Rewrite—Packets coming in are rewritten depending upon the configured table map.
- Mapping—Table map based policies can be used instead of set policies.

The following steps are required for table map marking:

1. Define the table map—Use the **table-map** global configuration command to map the values. The table does not know of the policies or classes within which it will be used. The default command in the table map is used to indicate the value to be copied into the to field when there is no matching from field.
2. Define the policy map—You must define the policy map where the table map will be used.
3. Associate the policy to an interface.

Note A table map policy on an input port changes the trust setting of that port to the from type of qos-marking.

Note In order to trust a value other than the dscp value, use table map with default copy in the ingress direction.

Note When you map a QoS group to a DSCP value in an egress table map policy, the QoS group does not map the equivalent COS value for DSCP. Configure a separate QoS group to COS table map if you want to define the QoS group to a non-zero COS value.

Traffic Conditioning

To support QoS in a network, traffic entering the service provider network needs to be policed on the network boundary routers to ensure that the traffic rate stays within the service limit. Even if a few routers at the network boundary start sending more traffic than what the network core is provisioned to handle, the increased traffic load leads to network congestion. The degraded performance in the network makes it difficult to deliver QoS for all the network traffic.

Traffic policing functions (using the police feature) and shaping functions (using the traffic shaping feature) manage the traffic rate, but differ in how they treat traffic when tokens are exhausted. The concept of tokens comes from the token bucket scheme, a traffic metering function.

Note When running QoS tests on network traffic, you may see different results for the shaper and policing data. Network traffic data from shaping provides more accurate results.

This table compares the policing and shaping functions.

Table 4: Comparison Between Policing and Shaping Functions

Policing Function	Shaping Function
Sends conforming traffic up to the line rate and allows bursts.	Smooths traffic and sends it out at a constant rate.
When tokens are exhausted, action is taken immediately.	When tokens are exhausted, it buffers packets and sends them out later, when tokens are available. A class with shaping has a queue associated with it which will be used to buffer the packets.
Policing has multiple units of configuration – in bits per second, packets per second and cells per second.	Shaping has only one unit of configuration - in bits per second.
Policing has multiple possible actions associated with an event, marking and dropping being example of such actions.	Shaping does not have the provision to mark packets that do not meet the profile.
Works for both input and output traffic.	Implemented for output traffic only.
Transmission Control Protocol (TCP) detects the line at line speed but adapts to the configured rate when a packet drop occurs by lowering its window size.	TCP can detect that it has a lower speed line and adapt its retransmission timer accordingly. This results in less scope of retransmissions and is TCP-friendly.

Policing

The QoS policing feature is used to impose a maximum rate on a traffic class. The QoS policing feature can also be used with the priority feature to restrict priority traffic. If the rate is exceeded, then a specific action is taken as soon as the event occurs. The rate (committed information rate [CIR] and peak information rate [PIR]) and the burst parameters (conformed burst size [B_c] and extended burst size [B_e]) are all configured in bytes per second.

The following policing forms or policers are supported for QoS:

- Single-rate two-color policing
- Dual-rate three-color policing

Note Single-rate three-color policing is not supported.

Single-Rate Two-Color Policing

Single-rate two-color policer is the mode in which you configure only a CIR and a B_c.

The B_c is an optional parameter, and if it is not specified it is computed by default. In this mode, when an incoming packet has enough tokens available, the packet is considered to be conforming. If at the time of packet arrival, enough tokens are not available within the bounds of B_c, the packet is considered to have exceeded the configured rate.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 48.

Dual-Rate Three-Color Policing

With the dual rate policer, the device supports only color-blind mode. In this mode, you configure a committed information rate (CIR) and a peak information rate (PIR). As the name suggests, there are two token buckets in this case, one for the peak rate, and one for the conformed rate.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 48.

In the color-blind mode, the incoming packet is first checked against the peak rate bucket. If there are not enough tokens available, the packet is said to violate the rate. If there are enough tokens available, then the tokens in the conformed rate buckets are checked to determine if there are enough tokens available. The tokens in the peak rate bucket are decremented by the size of the packet. If it does not have enough tokens available, the packet is said to have exceeded the configured rate. If there are enough tokens available, then the packet is said to conform, and the tokens in both the buckets are decremented by the size of the packet.

The rate at which tokens are replenished depends on the packet arrival. Assume that a packet comes in at time T1 and the next one comes in at time T2. The time interval between T1 and T2 determines the number of tokens that need to be added to the token bucket. This is calculated as:

Time interval between packets (T2-T1) * CIR)/8 bytes

Shaping

Shaping is the process of imposing a maximum rate of traffic, while regulating the traffic rate in such a way that the downstream switches and routers are not subjected to congestion. Shaping in the most common form is used to limit the traffic sent from a physical or logical interface.

Shaping has a buffer associated with it that ensures that packets which do not have enough tokens are buffered as opposed to being immediately dropped. The number of buffers available to the subset of traffic being shaped is limited and is computed based on a variety of factors. The number of buffers available can also be tuned using specific QoS commands. Packets are buffered as buffers are available, beyond which they are dropped.

Class-Based Traffic Shaping

The device uses class-based traffic shaping. This shaping feature is enabled on a class in a policy that is associated to an interface. A class that has shaping configured is allocated a number of buffers to hold the packets that do not have tokens. The buffered packets are sent out from the class using FIFO. In the most

common form of usage, class-based shaping is used to impose a maximum rate for an physical interface or logical interface as a whole. The following shaping forms are supported in a class:

- Average rate shaping
- Hierarchical shaping

Shaping is implemented using a token bucket. The values of CIR, B_c and B_e determine the rate at which the packets are sent out and the rate at which the tokens are replenished.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 48.

Average Rate Shaping

You use the **shape average** policy-map class command to configure average rate shaping.

This command configures a maximum bandwidth for a particular class. The queue bandwidth is restricted to this value even though the port has more bandwidth available. The device supports configuring shape average by either a percentage or by a target bit rate value.

Hierarchical Shaping

Shaping can also be configured at multiple levels in a hierarchy. This is accomplished by creating a parent policy with shaping configured, and then attaching child policies with additional shaping configurations to the parent policy.

The port shaper uses the class default and the only action permitted in the parent is shaping. The queuing action is in the child with the port shaper. With the user configured shaping, you cannot have queuing action in the child.

Queuing and Scheduling

The device uses both queuing and scheduling to help prevent traffic congestion. The device supports the following queuing and scheduling features:

- Bandwidth
- Weighted Tail Drop
- Priority queues
- Queue buffers
- Weighted Random Early Detection

When you define a queuing policy on a port, control packets are mapped to the best priority queue with the highest threshold. Control packets queue mapping works differently in the following scenarios:

- Without a quality of service (QoS) policy—If no QoS policy is configured, control packets with DSCP values 16, 24, 48, and 56 are mapped to queue 0 with the highest threshold of threshold2.
- With an user-defined policy—An user-defined queuing policy configured on egress ports can affect the default priority queue setting on control packets.

Note Queuing policy in egress direction does not support **match access-group** classification.

Control traffic is redirected to the best queue based on the following rules:

1. If defined in a user policy, the highest- level priority queue is always chosen as the best queue.
2. In the absence of a priority queue, Cisco IOS software selects queue 0 as the best queue. When the software selects queue 0 as the best queue, you must define the highest bandwidth to this queue to get the best QoS treatment to the control plane traffic.
3. If thresholds are not configured on the best queue, Cisco IOS software assigns control packets with Differentiated Services Code Point (DSCP) values 16, 24, 48, and 56 are mapped to threshold2 and reassigns the rest of the control traffic in the best queue to threshold1.

If a policy is not configured explicitly for control traffic, the Cisco IOS software maps all unmatched control traffic to the best queue with threshold2, and the matched control traffic is mapped to the queue as configured in the policy.

Note To provide proper QoS for Layer 3 packets, you must ensure that packets are explicitly classified into appropriate queues. When the software detects DSCP values in the default queue, then it automatically reassigns this queue as the best queue.

Bandwidth

The device supports the following bandwidth configurations:

- Bandwidth
- Bandwidth percent
- Bandwidth remaining precent

Bandwidth Percent

You can use the **bandwidth percent** policy-map class command to allocate a minimum bandwidth to a particular class. The total sum cannot exceed 100 percent and in case the total sum is less than 100 percent, then the rest of the bandwidth is divided equally among all bandwidth queues.

Note A queue can oversubscribe bandwidth in case the other queues do not utilize the entire port bandwidth.

You cannot mix bandwidth types on a policy map. For example, you cannot configure bandwidth in a single policy map using both a bandwidth percent and in kilobits per second.

Bandwidth Remaining Percent

Use the **bandwidth remaining percent** policy-map class command to create a percent for sharing unused bandwidth in specified queues. Any unused bandwidth will be used by these specific queues in the percent that is specified by the configuration. Use this command when the **priority** command is also used for certain queues in the policy.

When you assign percent, the queues will be assigned certain weights which are inline with these percents.

You can specify a percent between 0 to 100. For example, you can configure a bandwidth remaining percent of 2 on one class, and another queue with a bandwidth remaining percent of 4 on another class. The bandwidth remaining percent of 4 will be scheduled twice as often as the bandwidth remaining percent of 2.

The total bandwidth percent allocation for the policy can exceed 100. For example, you can configure a queue with a bandwidth remaining percent of 50, and another queue with a bandwidth remaining precent of 100.

Weighted Tail Drop

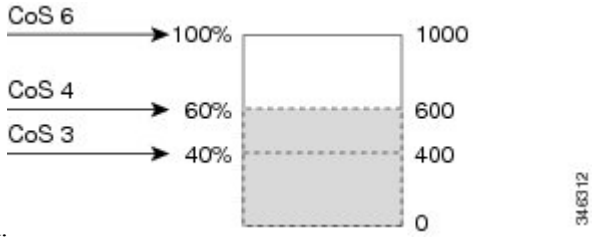
The device egress queues use an enhanced version of the tail-drop congestion-avoidance mechanism called weighted tail drop (WTD). WTD is implemented on queues to manage the queue lengths and to provide drop precedences for different traffic classifications.

As a frame is enqueued to a particular queue, WTD uses the frame's assigned QoS label to subject it to different thresholds. If the threshold is exceeded for that QoS label (the space available in the destination queue is less than the size of the frame), the device drops the frame.

Each queue has three configurable threshold values. The QoS label determines which of the three threshold values is subjected to the frame.

Figure 2: WTD and Queue Operation

The following figure shows an example of WTD operating on a queue whose size is 1000 frames. Three drop percentages are configured: 40 percent (400 frames), 60 percent (600 frames), and 100 percent (1000 frames). These percentages indicate that up to 400 frames can be queued at the 40-percent threshold, up to 600 frames at the 60-percent threshold, and up to 1000 frames at the 100-percent threshold.



In the example, CoS value 6 has a greater importance than the other CoS values, and is assigned to the 100-percent drop threshold (queue-full state). CoS values 4 is assigned to the 60-percent threshold, and CoS values 3 is assigned to the 40-percent threshold. All of these threshold values are assigned using the **queue-limit cos** command.

Assuming the queue is already filled with 600 frames, and a new frame arrives. It contains CoS value 4 and is subjected to the 60-percent threshold. If this frame is added to the queue, the threshold will be exceeded, so the device drops it.

Weighted Tail Drop Default Values

The following are the Weighted Tail Drop (WTD) default values and the rules for configuring WTD threshold values.

- If you configure less than three queue-limit percentages for WTD, then WTD default values are assigned to these thresholds.

  The following are the WTD threshold default values:

  Table 5: WTD Threshold Default Values

Threshold	Default Value Percentage
0	80
1	90
2	400

- If 3 different WTD thresholds are configured, then the queues are programmed as configured.
- If 2 WTD thresholds are configured, then the maximum value percentage will be 400.
- If a WTD single threshold is configured as x, then the maximum value percentage will be 400.
  - If the value of x is less than 90, then threshold1=90 and threshold 0= x.
  - If the value of x equals 90, then threshold1=90, threshold 0=80.
  - If the value x is greater than 90, then threshold1=x, threshold 0=80.

Priority Queues

Each port supports eight egress queues, of which two can be given a priority.

You use the **priority level** policy class-map command to configure the priority for two classes. One of the classes has to be configured with a priority queue level 1, and the other class has to be configured with a priority queue level 2. Packets on these two queues are subjected to less latency with respect to other queues.

You cannot send 100 percent line rate traffic when a priority queue is configured. There can only be 99.6 percent line rate traffic with priority queue configured, ensuring a latency of less than 20 microseconds.

Note You can configure a priority only with a level.

Only one strict priority or a priority with levels is allowed in one policy map. Multiple priorities with the same priority levels without kbps/percent are allowed in a policy map only if all of them are configured with police.

Priority Queue Policer

The switch supports configuration of policing rate on priority queue. Priority queue policer supports only a single-rate two-color policing.

Note Policing with table-maps is not supported.

Examples of Configuring Priority Queue Policer

Example1

```
Policy Map priority-1
    Class prio1
      priority level 1
     police rate percent 10
       conform-action transmit
       exceed-action drop
    Class prio2
      priority level 2
     police rate percent 5
       conform-action transmit
       exceed-action drop
    Class new
      bandwidth 20 (%)
```

Example 2

```
Policy Map priority-1
    Class prio1
      priority level 1 20 (%)
     police rate percent 10
       conform-action transmit
       exceed-action drop
    Class prio2
      priority level 2 25 (%)
     police rate percent 5
       conform-action transmit
       exceed-action drop
    Class new
      bandwidth 20 (%)
```

Queue Buffer

At boot time, when there is no policy map enabled on the wired port, there are two queues created by default. Wired ports can have a maximum of 8 queues configured using MQC-based policies. The following table shows which packets go into which one of the queues:

Table 6: DSCP, Precedence, and CoS - Queue Threshold Mapping Table

DSCP, Precedence or CoS	Queue	Threshold
Control Packets	0	2
Rest of Packets	1	2

Note You can guarantee the availability of buffers, set drop thresholds, and configure the maximum memory allocation for a queue. You use the **queue-buffers** policy-map class command to configure the queue buffers. You use the **queue-limit** policy-map class command to configure the maximum thresholds.

There are two types of buffer allocations: hard buffers, which are explicitly reserved for the queue, and soft buffers, which are available for other ports when unused by a given port.

Note By default, Queue 0 is not a priority queue. A policy-map can enable Queue 0 to be a priority queue by using the **priority level** command. If Queue 0 is assigned a priority level of 1, the soft maximum limit for this queue is automatically set to the same value as the hard maximum limit.

Queue Buffer Allocation

The buffer allocation to any queue can be tuned using the **queue-buffers ratio** policy-map class configuration command.

Dynamic Threshold and Scaling

Traditionally, reserved buffers are statically allocated for each queue. No matter whether the queue is active or not, its buffers are held up by the queue. In addition, as the number of queues increases, the portion of the reserved buffers allocated for each queue can become smaller and smaller. Eventually, a situation may occur where there are not enough reserved buffers to support a jumbo frame for all queues.

The device supports Dynamic Thresholding and Scaling (DTS), which is a feature that provides a fair and efficient allocation of buffer resources. When congestion occurs, this DTS mechanism provides an elastic buffer allocation for the incoming data based on the occupancy of the global/port resources. Conceptually, DTS scales down the queue buffer allocation gradually as the resources are used up to leave room for other queues, and vice versa. This flexible method allows the buffers to be more efficiently and fairly utilized.

As mentioned in the previous sections, there are two limits configured on a queue—a hard limit and a soft limit.

Hard limits are not part of DTS. These buffers are available only for that queue. The sum of the hard limits should be less than the globally set up hard maximum limit. The global hard limit configured for egress queuing is currently set to 12672. In the default scenario, when there are no MQC policies configured, each port takes up 176 buffers, allowing room for more hard buffers to be allocated based upon the configuration.

Soft limit buffers participate in the DTS process. Additionally, some of the soft buffer allocations can exceed the global soft limit allocation. The global soft limit allocation for egress queuing is currently set to 27024. The sum of the hard and soft limits add up to 39696, which in turn translates to 10.1 MB. Because the sum of the soft buffer allocations can exceed the global limit, it allows a specific queue to use a large number of buffers when the system is lightly loaded. The DTS process dynamically adjusts the per-queue allocation as the system becomes more heavily loaded.

Weighted Random Early Detection

Weighted random early detection (WRED) is a mechanism to avoid congestion in networks. WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion, thus avoiding large number of packet drops at once.

For more information about WRED, see Configuring Weighted Random Early Detection, on page 117

Trust Behavior

Port Security on a Trusted Boundary for Cisco IP Phones

In a typical network, you connect a Cisco IP Phone to a device port and cascade devices that generate data packets from the back of the telephone. The Cisco IP Phone guarantees the voice quality through a shared data link by marking the CoS level of the voice packets as high priority (CoS = 5) and by marking the data packets as low priority (CoS = 0). Traffic sent from the telephone to the device is typically marked with a tag that uses the 802.1Q header. The header contains the VLAN information and the class of service (CoS) 3-bit field, which is the priority of the packet.

For most Cisco IP Phone configurations, the traffic sent from the telephone to the device should be trusted to ensure that voice traffic is properly prioritized over other types of traffic in the network. By using the **trust device** interface configuration command, you configure the device port to which the telephone is connected to trust the traffic received on that port.

Note The **trust device** *device_type* command available in interface configuration mode is a stand-alone command on the device. When using this command in an AutoQoS configuration, if the connected peer device is not a corresponding device (defined as a device matching your trust policy), both CoS and DSCP values are set to "0" and any input policy will not take effect. If the connected peer device is a corresponding device, input policy will take effect.

With the trusted setting, you also can use the trusted boundary feature to prevent misuse of a high-priority queue if a user bypasses the telephone and connects the PC directly to the device. Without trusted boundary, the CoS labels generated by the PC are trusted by the device (because of the trusted CoS setting). By contrast, trusted boundary uses CDP to detect the presence of a Cisco IP Phone (such as the Cisco IP Phone 7910, 7935, 7940, and 7960) on a device port. If the telephone is not detected, the trusted boundary feature disables the trusted setting on the device port and prevents misuse of a high-priority queue. Note that the trusted boundary feature is not effective if the PC and Cisco IP Phone are connected to a hub that is connected to the device.

Trust Behavior for Wired Ports

In scenarios where the incoming packet type differs from the outgoing packet type, the trust behavior and the queuing behavior are explained in the following table. Note that the default trust mode for a port is DSCP based. The trust mode ‘falls back’ to CoS if the incoming packet is a pure Layer 2 packet. You can also change the trust setting from DSCP to CoS. This setting change is accomplished by using an MQC policy that has a class default with a 'set cos cos table default default-cos' action, where default-cos is the name of the table map created (which only performs a default copy).

For wired ports that are connected to the device (end points such as IP phones, laptops, cameras, telepresence units, or other devices), the trust device configuration is enabled on the interface. Their DSCP, precedence, or CoS values coming in from these end points are trusted by the device and therefore are retained in the absence of any explicit policy configuration.

The packets are enqueued to the appropriate queue per the default initial configuration. No priority queuing at the device is done by default. This is true for unicast and multicast packets.

Table 7: Trust and Queuing Behavior

Incoming Packet	Outgoing Packet	Trust Behavior	Queuing Behavior
Layer 3	Layer 3	Preserve DSCP/Precedence	Based on DSCP
Layer 2	Layer 2	Not applicable	Based on CoS
Tagged	Tagged	Preserve DSCP and CoS	Based on DSCP (trust DSCP takes precedence)
Layer 3	Tagged	Preserve DSCP, CoS is set to 0	Based on DSCP

Standard QoS Default Settings

Default Wired QoS Configuration

There are two queues configured by default on each wired interface on the device. All control traffic traverses and is processed through queue 0. All other traffic traverses and is processed through queue 1.

DSCP Maps

Default CoS-to-DSCP Map

When DSCP transparency mode is disabled, the DSCP values are derived from CoS as per the following table. If these values are not appropriate for your network, you need to modify them.

Note The DSCP transparency mode is disabled by default. If it is enabled (**mls qos rewrite ip dscp** configuration command), DSCP rewrite will not happen.

Table 8: Default CoS-to-DSCP Map

CoS Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

Default IP-Precedence-to-DSCP Map

You use the IP-precedence-to-DSCP map to map IP precedence values in incoming packets to a DSCP value that QoS uses internally to represent the priority of the traffic. The following table shows the default IP-precedence-to-DSCP map. If these values are not appropriate for your network, you need to modify them.

Table 9: Default IP-Precedence-to-DSCP Map

IP Precedence Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

Default DSCP-to-CoS Map

You use the DSCP-to-CoS map to generate a CoS value, which is used to select one of the four egress queues. The following table shows the default DSCP-to-CoS map. If these values are not appropriate for your network, you need to modify them.

Table 10: Default DSCP-to-CoS Map

DSCP Value	CoS Value
0–7	0
8–15	1
16–23	2
24–31	3
32–39	4
40–47	5
48–55	6
56–63	7

Restrictions for QoS on Wired Targets

A target is an entity where a policy is applied. A wired target can be either a port or VLAN.

The following are restrictions for applying QoS features on the device for the wired target:

- A maximum of 8 queuing classes are supported on the device port for the wired target.
- A maximum of 63 policers are supported per policy on the wired port for the wired target, in the ingress or egress directions.
- A maximum of 1599 policy-maps can be created.
- No more than two levels are supported in a QoS hierarchy.
- In a hierarchical policy, overlapping actions between parent and child are not allowed, except when a policy has the port shaper in the parent and queuing features in the child policy.
- A QoS policy cannot be attached to any EtherChannel interface.
- Policing in both the parent and child is not supported in a QoS hierarchy.
- Marking in both the parent and child is not supported in a QoS hierarchy.
- With shaping, there is an IPG overhead of 20Bytes for every packet that is accounted internally in the hardware. Shaping accuracy will be effected by this, specially for packets of small size.
- Class-maps with empty actions are not supported. If there are two policies with the same order of class-maps and if there are class-maps with no action in one of the policies, there may be traffic drops. As a workaround, allocate minimal bandwidth for all the classes in PRIORITY_QUEUE.
- A maximum of 256 classes are supported per policy on the wired port for the wired target.
- Based on the Cisco UADP architecture, traffic is subjected to QoS lookup and the corresponding configured actions even if this traffic is later dropped in the Egress Global Resolution block and is never transmitted out of the actual interface.
- The actions under a policer within a policy map have the following restrictions:
  - The conform action must be transmit.
  - The exceed/violate action for markdown type can only be cos2cos, prec2prec, dscp2dscp.
  - The markdown types must be the same within a policy.
- Only marking policy is supported on SVI.
- A port-level input marking policy takes precedence over an SVI policy; however, if no port policy is configured, the SVI policy takes precedence. For a port policy to take precedence, define a port-level policy; so that the SVI policy is overwritten.
- Classification counters have the following specific restrictions:
  - Classification counters count packets instead of bytes.
  - Filter-based classification counters are not supported
  - Only QoS configurations with marking or policing trigger the classification counter.
  - As long as there is policing or marking action in the policy, the class will have classification counters.
  - Classification counters are not supported on pure queuing policies under any class-map.
  - When there are multiple match statements in a class, the traffic counter is cumulative for all the match statements in the class.

- Classification counters (class-map) are not available in queuing policy with actions like bandwidth, WRED, queue-buffer, shaping, and so on. The **show policy-map interface** command output will display classification counters (class-map) only for policies having either remarking or policer action.

- The device supports a total of eight table maps for policer exceed markdown and eight table maps for policer violate markdown.
- Hierarchical policies are required for the following:
  - Port-shapers
  - Aggregate policers
  - PV policy
  - Parent shaping and child marking/policing
- In a HQoS policy with parent shaping and child policy having priority level queuing and priority level policing, the statistics for policing are not updated. Only QoS shaper statistics are updated. To view the QoS shaper statistics, use the **show policy-map interface** command in global configuration mode.
- For ports with wired targets, these are the only supported hierarchical policies:
  - Police chaining in the same policy is unsupported.
  - Hierarchical queuing is unsupported in the same policy (port shaper is the exception).
  - In a parent class, all filters must have the same type. The child filter type must match the parent filter type with the following exceptions:
    - If the parent class is configured to match IP, then the child class can be configured to match the ACL.
    - If the parent class is configured to match CoS, then the child class can be configured to match the ACL.
- The **trust device** *device_type* command available in interface configuration mode is a stand-alone command on the device. When using this command in an AutoQoS configuration, if the connected peer device is not a corresponding device (defined as a device matching your trust policy), both CoS and DSCP values are set to "0" and any input policy will not take effect. If the connected peer device is a corresponding device, input policy will take effect.

The following are restrictions for applying QoS features on the VLAN to the wired target:

- For a flat or nonhierarchical policy, only marking or a table map is supported.

The following are restrictions and considerations for applying QoS features on EtherChannel and channel member interfaces:

- QoS is not supported on an EtherChannel interface.
- QoS is supported on EtherChannel member interfaces in both ingress and egression directions. All EtherChannel members must have the same QoS policy applied. If the QoS policy is not the same, each individual policy on the different link acts independently.

- On attaching a service policy to channel members, the following warning message appears to remind the user to make sure the same policy is attached to all ports in the EtherChannel: ' Warning: add service policy will cause inconsistency with port xxx in ether channel xxx. '.
- Auto QoS is not supported on EtherChannel members.

Note On attaching a service policy to an EtherChannel, the following message appears on the console: ' Warning: add service policy will cause inconsistency with port xxx in ether channel xxx. '. This warning message should be expected. This warning message is a reminder to attach the same policy to other ports in the same EtherChannel. The same message will be seen during boot up. This message does not mean there is a discrepancy between the EtherChannel member ports.

How to Configure QoS

Configuring Class, Policy, and Maps

Creating a Traffic Class

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

Before you begin

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

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**class-map** *class-map name* {**match-any** | **match-all**}
	Example:
	`Device(config)# class-map test_1000`
	`Device(config-cmap)#`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• **match-any:** Any one of the match criteria must be met for traffic entering the traffic class to be classified as part of it.

	Command or Action	Purpose
		• **match-all:** All of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. **Note** This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match access-group** {*index number* | *name*}
	Example:
	`Device(config-cmap)# match access-group 100`
	`Device(config-cmap)#`
		The following parameters are available for this command:
		• access-group
		• cos
		• dscp
		• group-object
		• ip
		• mpls
		• non-client-nrt
		• precedence
		• protocol
		• qos-group
		• vlan
		• wlan
		(Optional) For this example, enter the access-group ID:
		• Access list index (value from 1 to 2799)
		• Named access list

Step 4	**match cos** *cos value*
	Example:
	`Device(config-cmap)# match cos 2 3 4 5`
	`Device(config-cmap)#`
		(Optional) Matches IEEE 802.1Q or ISL class of service (user) priority values.
		• Enters up to 4 CoS values separated by spaces (0 to 7).

Step 5	**match dscp** *dscp value* **Example:** `Device(config-cmap)# match dscp af11 af12`	(Optional) Matches the DSCP values in IPv4 and IPv6 packets.

	Command or Action	Purpose
	`Device(config-cmap)#`	

Step 6	**match ip** { **dscp** *dscp value* | **precedence** *precedence value* }
	Example:
	`Device(config-cmap)# match ip dscp af11 af12`
	`Device(config-cmap)#`
		(Optional) Matches IP values including the following:
		• **dscp**—Matches IP DSCP (DiffServ codepoints).
		• **precedence**—Matches IP precedence (0 to 7).
		Note Since CPU generated packets are not marked at egress, the packet will not match the configured class-map.

Step 7	**match qos-group** *qos group value*
	Example:
	`Device(config-cmap)# match qos-group 10`
	`Device(config-cmap)#`
		(Optional) Matches QoS group value (from 0 to 31).

Step 8	**match vlan** *vlan value*
	Example:
	`Device(config-cmap)# match vlan 210`
	`Device(config-cmap)#`
		(Optional) Matches a VLAN ID (from 1 to 4095).

Step 9	**end** **Example:** `Device(config-cmap)# end`	Saves the configuration changes.

What to do next

Configure the policy map.

Creating a Traffic Policy

To create a traffic policy, use the **policy-map** global configuration 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 entered after you enter the policy map configuration mode. After entering the **class** command, the device is automatically in policy map class configuration mode, which is where the QoS policies for the traffic policy are defined.

The following policy map class-actions are supported:

- bandwidth—Bandwidth configuration options.
- exit—Exits from the QoS class action configuration mode.
- no—Negates or sets default values for the command.
- police—Policer configuration options.
- priority—Strict scheduling priority configuration options for this class.
- queue-buffers—Queue buffer configuration options.
- queue-limit—Queue maximum threshold for Weighted Tail Drop (WTD) configuration options.
- service-policy—Configures the QoS service policy.
- set—Sets QoS values using the following options:
  - CoS values
  - DSCP values
  - Precedence values
  - QoS group values
- shape—Traffic-shaping configuration options.

Before you begin

You should have first created a class map.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy-map name*
	Example:
	`Device(config)# policy-map test_2000`
	`Device(config-pmap)#`
		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* | **class-default**}
	Example:
	`Device(config-pmap)# class test_1000`
	`Device(config-pmap-c)#`
		Specifies the name of the class whose policy you want to create or change.
		You can also create a system default class for unclassified packets.

	Command or Action	Purpose

Step 4	**bandwidth** { *Kb/s* | **percent** *percentage* | **remaining** {*percent* | *ratio*} }
	Example:
	`Device(config-pmap-c)# bandwidth 500`
	`Device(config-pmap-c)#`
		(Optional) Sets the bandwidth using one of the following:
		• *Kb/s*: Kilobits per second, enter a value between 100 and 100000000 for Kb/s.
		• **percent**: Enter the percentage of the total bandwidth to be used for this policy map.
		• **remaining**: Enter the percentage ratio of the remaining bandwidth.
		For a more detailed example of this command and its usage, see Configuring Bandwidth, on page 86.

Step 5	**exit**
	Example:
	`Device(config-pmap-c)# exit`
	`Device(config-pmap-c)#`
		(Optional) Exits from QoS class action configuration mode.

Step 6	**no**
	Example:
	`Device(config-pmap-c)# no`
	`Device(config-pmap-c)#`
		(Optional) Negates the command.

Step 7	**police** {*target_bit_rate* | **cir** | **rate**}
	Example:
	`Device(config-pmap-c)# police 100000`
	`Device(config-pmap-c)#`
		(Optional) Configures the policer:
		• *target_bit_rate*: Enter the bit rate per second, enter a value between 8000 and 10000000000.
		• **cir**: Committed Information Rate
		• **rate**: Specify police rate, PCR for hierarchical policies or SCR for single-level ATM 4.0 policer policies.
		For a more detailed example of this command and its usage, see Configuring Police , on page 88.

Step 8	**priority level** *level value*
	Example:
	`Device(config-pmap-c)# priority level 1`
	`Device(config-pmap-c)#`
		(Optional) Sets the strict scheduling priority for this class. Command options include:
		• **level**: Establishes a multi-level priority queue. Enter a value (1 or 2).

	Command or Action	Purpose
		For a more detailed example of this command and its usage, see Configuring Priority, on page 90.

Step 9	**queue-buffers ratio** *ratio limit*
	Example:
	`Device(config-pmap-c)# queue-buffers ratio 10`
	`Device(config-pmap-c)#`
		(Optional) Configures the queue buffer for the class. Enter the queue buffers ratio limit (0 to 100).
		For a more detailed example of this command and its usage, see Configuring Queue Buffers , on page 94.

Step 10	**queue-limit** {*packets* | **cos** | **dscp** | **percent**}
	Example:
	`Device(config-pmap-c)# queue-limit cos 7 percent 50`
	`Device(config-pmap-c)#`
		(Optional) Specifies the queue maximum threshold for the tail drop:
		• *packets*: Packets by default, enter a value between 1 to 2000000.
		• **cos**: Enter the parameters for each COS value.
		• **dscp**: Enter the parameters for each DSCP value.
		• **percent**: Enter the percentage for the threshold.
		For a more detailed example of this command and its usage, see Configuring Queue Limits, on page 97.

Step 11	**service-policy** *policy-map name*
	Example:
	`Device(config-pmap-c)# service-policy test_2000`
	`Device(config-pmap-c)#`
		(Optional) Configures the QoS service policy.

Step 12	**set** {**cos** | **dscp** | **ip** | **precedence** | **qos-group** | **wlan**}
	Example:
	`Device(config-pmap-c)# set cos 7`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **dscp**: Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**: Sets IP specific values.
		• **precedence**: Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**: Sets the QoS Group.

	Command or Action	Purpose

Step 13	**shape average** {*target _bit_rate* | **percent**}
	Example:
	`Device(config-pmap-c) #shape average percent 50`
	`Device(config-pmap-c) #`
		(Optional) Sets the traffic shaping. Command parameters include:
		• *target_bit_rate*: Target bit rate.
		• **percent**: Percentage of interface bandwidth for Committed Information Rate.
		For a more detailed example of this command and its usage, see Configuring Shaping, on page 99.

Step 14	**end**
	Example:
	`Device(config-pmap-c) #end`
	`Device(config-pmap-c) #`
		Saves the configuration changes.

What to do next

Configure the interface.

Configuring Class-Based Packet Marking

This procedure explains how to configure the following class-based packet marking features on your device:

- CoS value
- DSCP value
- IP value
- Precedence value
- QoS group value
- WLAN value

Before you begin

You should have created a class map and a policy map before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy1`
	`Device(config-pmap)#`
		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:
	`Device(config-pmap)# class class1`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change.
		Command options for policy class map configuration mode include the following:
		• **bandwidth**—Bandwidth configuration options.
		• **exit**—Exits from the QoS class action configuration mode.
		• **no**—Negates or sets default values for the command.
		• **police**—Policer configuration options.
		• **priority**—Strict scheduling priority configuration options for this class.
		• **queue-buffers**—Queue buffer configuration options.
		• **queue-limit**—Queue maximum threshold for Weighted Tail Drop (WTD) configuration options.
		• **service-policy**—Configures the QoS service policy.
		• **set**—Sets QoS values using the following options:
		  • CoS values
		  • DSCP values
		  • Precedence values
		  • QoS group values
		  • WLAN values
		• **shape**—Traffic-shaping configuration options.

	Command or Action	Purpose
		Note This procedure describes the available configurations using **set** command options. The other command options (**bandwidth**) are described in other sections of this guide. Although this task lists all of the possible **set** commands, only one **set** command is supported per class.

Step 4	**set cos** {*cos value* | **cos table** *table-map name* | **dscp table** *table-map name* | **precedence table** *table-map name* | **qos-group table** *table-map name* | **wlan user-priority table** *table-map name*}
	Example:
	`Device(config-pmap)# set cos 5`
	`Device(config-pmap)#`
		(Optional) Sets the specific IEEE 802.1Q Layer 2 CoS value of an outgoing packet. Values are from 0 to7.
		You can also set the following values using the **set cos** command:
		• **cos table**—Sets the CoS value based on a table map.
		• **dscp table**—Sets the code point value based on a table map.
		• **precedence table**—Sets the code point value based on a table map.
		• **qos-group table**—Sets the CoS value from QoS group based on a table map.
		• **wlan user-priority table**—Sets the CoS value from the WLAN user priority based on a table map.

Step 5	**set dscp** {*dscp value* | **default** | **dscp table** *table-map name* | **ef** | **precedence table** *table-map name* | **qos-group table** *table-map name* | **wlan user-priority table** *table-map name*}
	Example:
	`Device(config-pmap)# set dscp af11`
	`Device(config-pmap)#`
		(Optional) Sets the DSCP value.
		In addition to setting specific DSCP values, you can also set the following using the **set dscp** command:
		• **default**—Matches packets with default DSCP value (000000).
		• **dscp table**—Sets the packet DSCP value from DSCP based on a table map.
		• **ef**—Matches packets with EF DSCP value (101110).
		• **precedence table**—Sets the packet DSCP value from precedence based on a table map.

	Command or Action	Purpose
		• **qos-group table**—Sets the packet DSCP value from a QoS group based upon a table map. • **wlan user-priority table**—Sets the packet DSCP value based upon a WLAN user-priority based upon a table map.

Step 6	**set ip** {**dscp** | **precedence**}
	Example:
	`Device(config-pmap)# set ip dscp c3`
	`Device(config-pmap)#`
		(Optional) Sets IP specific values. These values are either IP DSCP or IP precedence values.
		You can set the following values using the **set ip dscp** command:
		• *dscp value*—Sets a specific DSCP value.
		• **default**—Matches packets with default DSCP value (000000).
		• **dscp table**—Sets the packet DSCP value from DSCP based on a table map.
		• **ef**—Matches packets with EF DSCP value (101110).
		• **precedence table**—Sets the packet DSCP value from precedence based on a table map.
		• **qos-group table**—Sets the packet DSCP value from a QoS group based upon a table map.
		• **wlan user-priority table**—Sets the packet DSCP value based upon a WLAN user-priority based upon a table map.
		You can set the following values using the **set ip precedence** command:
		• *precedence value*—Sets the precedence value (from 0 to 7) .
		• **cos table**—Sets the packet precedence value from Layer 2 CoS based on a table map.
		• **dscp table**—Sets the packet precedence from DSCP value based on a table map.
		• **precedence table**—Sets the precedence value from precedence based on a table map

	Command or Action	Purpose
		• **qos-group table**—Sets the precedence value from a QoS group based upon a table map.

Step 7	**set precedence** {*precedence value* | **cos table** *table-map name* | **dscp table** *table-map name* | **precedence table** *table-map name* | **qos-group table** *table-map name*}
	Example:
	`Device(config-pmap)# set precedence 5`
	`Device(config-pmap)#`
		(Optional) Sets precedence values in IPv4 and IPv6 packets.
		You can set the following values using the **set precedence** command:
		• *precedence value*—Sets the precedence value (from 0 to 7) .
		• **cos table**—Sets the packet precedence value from Layer 2 CoS on a table map.
		• **dscp table**—Sets the packet precedence from DSCP value on a table map.
		• **precedence table**—Sets the precedence value from precedence based on a table map.
		• **qos-group table**—Sets the precedence value from a QoS group based upon a table map.

Step 8	**set qos-group** {*qos-group value* | **dscp table** *table-map name* | **precedence table** *table-map name*}
	Example:
	`Device(config-pmap)# set qos-group 10`
	`Device(config-pmap)#`
		(Optional) Sets QoS group values. You can set the following values using this command:
		• *qos-group value*—A number from 1 to 31.
		• **dscp table**—Sets the code point value from DSCP based on a table map.
		• **precedence table**—Sets the code point value from precedence based on a table map.

Step 9	**set wlan user-priority** {*wlan user-priority value* | **cos table** *table-map name* | **dscp table** *table-map name* | **qos-group table** *table-map name* | **wlan table** *table-map name*}
	Example:
	`Device(config-pmap)# set wlan user-priority 1`
	`Device(config-pmap)#`
		(Optional) Sets the WLAN user priority value. You can set the following values using this command:
		• *wlan user-priority value*—A value between 0 to 7.
		• **cos table**—Sets the WLAN user priority value from CoS based on a table map.
		• **dscp table**—Sets the WLAN user priority value from DSCP based on a table map.

	Command or Action	Purpose
		• **qos-group table**—Sets the WLAN user priority value from QoS group based on a table map. • **wlan table**—Sets the WLAN user priority value from the WLAN user priority based on a table map.

Step 10	**end**
	Example:
	`Device(config-pmap)# end`
	`Device#`
		Saves configuration changes.

Step 11	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Attach the traffic policy to an interface using the **service-policy** command.

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).

Before you begin

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

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** *type* **Example:** `Device(config)# interface GigabitEthernet1/0/1`	Enters interface configuration mode and configures an interface. Command parameters for the interface configuration include:

	Command or Action	Purpose
	`Device(config-if)#`	• **ANI**— Autonomic-Networking virtual interface • **AccessTunnel** — Access Tunnel interface • **Auto Template**— Auto-template interface • **CEM-PG**— Circuit Emulation interface with Protection group • **FortyGigabitEthernet**— Forty Gigabit Ethernet • **GigabitEthernet**—Gigabit Ethernet IEEE 802.3z • **Internal Interface**— Internal interface • **LISP**— Locator ID Separation Protocol Virtual Interface • **Loopback**—Loopback interface • **Null**—Null interface • **PROTECTION_GROUP**—Protection-group controller • **Port-channel**—Ethernet Channel of interface • **SDH_ACR** —Virtual SDH-ACR controller • **TLS-VIF** —TLS Virtual Interface • **TenGigabitEthernet**—10-Gigabit Ethernet • **Tunnel**—Tunnel interface • **Tunnel-tp**—MPLS Transport Profile interface • **Vlan**—Catalyst VLANs • **Range**—Interface range **Note** Tunnel interface is not supported.
Step 3	**service-policy** {**input** *policy-map* | **output** *policy-map*} **Example:** `Device(config-if)# service-policy output policy_map_01`	Attaches a policy map to an input or output interface. This policy map is then used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

	Command or Action	Purpose
	`Device(config-if)#`	

Step 4	**end**
	Example:
	`Device(config-if)# end`
	`Device#`
		Saves configuration changes.

Step 5	**show policy map** **Example:** `Device# show policy map`	(Optional) Displays statistics for the policy on the specified interface.

What to do next

Proceed to attach any other traffic policy to an interface, and to specify the direction in which the policy should be applied.

Classifying, Policing, and Marking Traffic on Physical Ports by Using Policy Maps

You can configure a nonhierarchical policy map on a physical port that specifies which traffic class to act on. Actions supported are remarking and policing.

Before you begin

You should have already decided upon the classification, policing, and marking of your network traffic by policy maps prior to beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**class-map** {*class-map name* | **match-any** | **match-all**}
	Example:
	`Device(config)# class-map ipclass1`
	`Device(config-cmap)# exit`
	`Device(config)#`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• 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.

	Command or Action	Purpose
		• If you specify **match-all**, all of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. **Note** This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match access-group** { *access list index* | *access list name* }
	Example:
	`Device(config-cmap)# match access-group 1000`
	`Device(config-cmap)# exit`
	`Device(config)#`
		The following parameters are available for this command:
		• access-group
		• cos
		• dscp
		• group-object
		• ip
		• mpls
		• non-client-nrt
		• precedence
		• protocol
		• qos-group
		• vlan
		• wlan
		(Optional) For this example, enter the access-group ID:
		• Access list index (value from 1 to 2799)
		• Named access list

Step 4	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map flowit`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.

Step 5	**class** {*class-map-name* | **class-default**} **Example:**	Defines a traffic classification, and enter policy-map class configuration mode.

	Command or Action	Purpose
	`Device(config-pmap)# class ipclass1` `Device(config-pmap-c)#`	By default, no policy map class-maps are defined. If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command. A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.

Step 6	**set** {**cos** | **dscp** | **ip** | **precedence** | **qos-group** | **wlan user-priority**}
	Example:
	`Device(config-pmap-c)# set dscp 45`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**—Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **dscp**—Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**—Sets IP specific values.
		• **precedence**—Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**—Sets QoS group.
		• **wlan user-priority**—Sets WLAN user priority.
		In this example, the **set dscp** command classifies the IP traffic by setting a new DSCP value in the packet.

Step 7	**police** {*target_bit_rate* | **cir** | **rate** }
	Example:
	`Device(config-pmap-c)# police 100000 conform-action transmit exceed-action drop`
	`Device(config-pmap-c)#`
		(Optional) Configures the policer:
		• *target_bit_rate*—Specifies the bit rate per second, enter a value between 8000 and 10000000000.
		• **cir**—Committed Information Rate.
		• **rate**—Specifies the police rate PCR for hierarchical policies.
		In this example, the **police** command adds a policer to the class where any traffic beyond the 100000 set target bit rate is dropped.

	Command or Action	Purpose
Step 8	**exit** **Example:** `Device(config-pmap-c)# exit`	Returns to policy map configuration mode.
Step 9	**exit** **Example:** `Device(config-pmap)# exit`	Returns to global configuration mode.
Step 10	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigabitEthernet 1/0/2`	Specifies the port to attach to the policy map, and enters interface configuration mode. Valid interfaces include physical ports.
Step 11	**service-policy input** *policy-map-name* **Example:** `Device(config-if)# service-policy input flowit`	Specifies the policy-map name, and applies it to an ingress port. Only one policy map per ingress port is supported.
Step 12	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 13	**show policy-map** [*policy-map-name* [**class** *class-map-name*]] **Example:** `Device# show policy-map`	(Optional) Verifies your entries.
Step 14	**copy running-config startup-config** **Example:** `Device# copy-running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

If applicable to your QoS configuration, configure classification, policing, and marking of traffic on SVIs by using policy maps.

Classifying and Marking Traffic by Using Policy Maps

Before you begin

You should have already decided upon the classification, policing, and marking of your network traffic by using policy maps prior to beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**class-map** {*class-map name* | **match-any** | **match-all**} **Example:** `Device(config)# class-map class_vlan100`	Enters class map configuration mode. • Creates a class map to be used for matching packets to the class whose name you specify. • 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. • If you specify **match-all**, all of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. **Note** This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match vlan** *vlan number*
	Example:
	`Device(config-cmap)# match vlan 100`
	`Device(config-cmap)# exit`
	`Device(config)#`
		Specifies the VLAN to match to the class map.

	Command or Action	Purpose

Step 4	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map policy_vlan100`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.

Step 5	**description** *description* **Example:** `Device(config-pmap)# description vlan 100`	(Optional) Enters a description of the policy map.

Step 6	**class** {*class-map-name* | **class-default**}
	Example:
	`Device(config-pmap)# class class_vlan100`
	`Device(config-pmap-c)#`
		Defines a traffic classification, and enters the policy-map class configuration mode.
		By default, no policy map class-maps are defined.
		If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command.
		A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.

Step 7	**set** {**cos** | **dscp** | **ip** | **precedence** | **qos-group** | **wlan user-priority**}
	Example:
	`Device(config-pmap-c)# set dscp af23`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**—Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **dscp**—Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**—Sets IP specific values.
		• **precedence**—Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**—Sets QoS group.
		• **wlan user-priority**—Sets WLAN user-priority.

	Command or Action	Purpose
		In this example, the **set dscp** command classifies the IP traffic by matching the packets with a DSCP value of AF23 (010010).
Step 8	**exit** **Example:** `Device(config-pmap-c)# exit`	Returns to policy map configuration mode.
Step 9	**exit** **Example:** `Device(config-pmap)# exit`	Returns to global configuration mode.
Step 10	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/3`	Specifies the port to attach to the policy map, and enters interface configuration mode. Valid interfaces include physical ports.
Step 11	**service-policy input** *policy-map-name* **Example:** `Device(config-if)# service-policy input policy_vlan100`	Specifies the policy-map name, and applies it to an ingress port. Only one policy map per ingress port is supported.
Step 12	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 13	**show policy-map** [*policy-map-name* [**class** *class-map-name*]] **Example:** `Device# show policy-map`	(Optional) Verifies your entries.
Step 14	**copy running-config startup-config** **Example:** `Device# copy-running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Table Maps

Table maps are a form of marking, and also enable the mapping and conversion of one field to another using a table. For example, a table map can be used to map and convert a Layer 2 CoS setting to a precedence value in Layer 3.

Note

- A table map can be referenced in multiple policies or multiple times in the same policy.
- A table map configured for a custom output policy under the default class-map, takes affect for all DSCP traffic regardless of which class map the traffic is classified for. The workaround is to remove the table map and configure the **set dscp** command under the default class to change the DSCP marking for classified traffic. If there is any non-queuing action (policer or marking) on a user-defined class, then the packet retains its value or remarks in the user-defined class itself.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**table-map** *name* {**default** {*default value* | **copy** | **ignore**} | **exit** | **map** {**from** *from value* **to** *to value* } | **no**}
	Example:
	`Device(config)# table-map table01`
	`Device(config-tablemap)#`
		Creates a table map and enters the table map configuration mode. In table map configuration mode, you can perform the following tasks:
		• **default**: Configures the table map default value, or sets the default behavior for a value not found in the table map to copy or ignore.
		• **exit**: Exits from the table map configuration mode.
		• **map**: Maps a *from* to a *to* value in the table map.
		• **no**: Negates or sets the default values of the command.

Step 3	**map from** *value* **to** *value*
	Example:
	`Device(config-tablemap)# map from 0 to 2`
	`Device(config-tablemap)# map from 1 to`
		In this step, packets with DSCP values 0 are marked to the CoS value 2, DSCP value 1 to the CoS value 4, DSCP value 24 to the CoS value 3, DSCP value 40 to the CoS value 6 and all others to the CoS value 0.

	Command or Action	Purpose
	4 Device(config-tablemap)# **map from 24 to 3** Device(config-tablemap)# **map from 40 to 6** Device(config-tablemap)# **default 0** Device(config-tablemap)#	**Note** The mapping from CoS values to DSCP values in this example is configured by using the **set** policy map class configuration command as described in a later step in this procedure.
Step 4	**exit** **Example:** Device(config-tablemap)# **exit** Device(config)#	Returns to global configuration mode.
Step 5	**exit** **Example:** Device(config) **exit** Device#	Returns to privileged EXEC mode.
Step 6	**show table-map** **Example:** Device# **show table-map** Table Map table01 from 0 to 2 from 1 to 4 from 24 to 3 from 40 to 6 default 0	Displays the table map configuration.
Step 7	**configure terminal** **Example:** Device# **configure terminal** Device(config)#	Enters global configuration mode.
Step 8	**policy-map** **Example:** Device(config)# **policy-map table-policy** Device(config-pmap)#	Configures the policy map for the table map.
Step 9	**class class-default** **Example:** Device(config-pmap)# **class**	Matches the class to the system default.

	Command or Action	Purpose
	class-default Device(config-pmap-c)#	
Step 10	**set cos dscp table** *table map name* **Example:** Device(config-pmap-c)# **set cos dscp table table01** Device(config-pmap-c)#	If this policy is applied on input port, that port will have trust DSCP enabled on that port and marking will take place depending upon the specified table map.
Step 11	**end** **Example:** Device(config-pmap-c)# **end** Device#	Returns to privileged EXEC mode.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring QoS Features and Functionality

Configuring Bandwidth

This procedure explains how to configure bandwidth on your device.

Before you begin

You should have created a class map for bandwidth before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** Device# **configure terminal**	Enters global configuration mode.
Step 2	**policy-map** *policy name* **Example:** Device(config)# **policy-map policy_bandwidth01**	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.

	Command or Action	Purpose
	`Device(config-pmap)#`	

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_bandwidth01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**bandwidth** {*Kb/s* | **percent** *percentage* | **remaining** {**ratio** *ratio* }}
	Example:
	`Device(config-pmap-c)# bandwidth 200000`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map. The parameters include:
		• *Kb/s*: Configures a specific value in kilobits per second (from 100 to 100000000).
		• **percent**: Allocates minimum bandwidth to a particular class based on a percentage. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is equally divided along all bandwidth queues.
		• **remaining**: Allocates minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize entire port bandwidth. The total sum cannot exceed 100 percent. It is preferred to use this command when the **priority** command is used for certain queues in the policy. You can also assign ratios rather than percentages to each queue; the queues will be assigned certain weights which are inline with these ratios. Ratios can range from 0 to 100. Total bandwidth ratio allocation for the policy in this case can exceed 100.

	Command or Action	Purpose
		Note You cannot mix bandwidth types on a policy map. For example, you cannot configure bandwidth in a single policy map using both a bandwidth percent and in kilobits per second.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating the policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Police

This procedure explains how to configure policing on your device.

Before you begin

You should have created a class map for policing before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal**	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_police01`
	`Device(config-pmap)#`
		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:**	Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options

	Command or Action	Purpose
	`Device(config-pmap)# class class_police01` `Device(config-pmap-c)#`	for policy class map configuration mode include the following: • *word*—Class map name. • **class-default**—System default class matching any otherwise unclassified packets.

Step 4	**police** {*target_bit_rate* [*burst bytes* | **bc** | **conform-action** | **pir**] | **cir** {*target_bit_rate* | **percent** *percentage*} | **rate** {*target_bit_rate* | **percent** *percentage*} **conform-action transmit exceed-action** {**drop** [**violate action**] | **set-cos-transmit** | **set-dscp-transmit** | **set-prec-transmit** | **transmit** [**violate action**] }}
	Example:
	`Device(config-pmap-c)# police 8000 conform-action transmit exceed-action drop`
	`Device(config-pmap-c)#`
		The following **police** subcommand options are available:
		• *target_bit_rate*—Bits per second (from 8000 to 10000000000).
		• *burst bytes*—Enter a value from 1000 to 512000000.
		• **bc**—Conform burst.
		• **conform-action**—Action taken when rate is less than conform burst.
		• **pir**—Peak Information Rate.
		• **cir**—Committed Information Rate.
		• *target_bit_rate*—Target bit rate (8000 to10000000000).
		• **percent**—Percentage of interface bandwidth for CIR.
		• **rate**—Specifies the police rate, PCR for hierarchical policies, or SCR for single-level ATM 4.0 policer policies.
		• *target_bit_rate*—Target Bit Rate (8000 to 10000000000).
		• **percent**—Percentage of interface bandwidth for rate.
		The following **police conform-action transmit exceed-action** subcommand options are available:
		• **drop**—Drops the packet.
		• **set-cos-transmit**—Sets the CoS value and sends it.
		• **set-dscp-transmit**—Sets the DSCP value and sends it.

	Command or Action	Purpose
		• **set-prec-transmit**—Rewrites the packet precedence and sends it. • **transmit**—Transmits the packet. **Note** Policer-based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies. **Note** The **show policy-map** command output does not display counters for conformed bytes and exceeded bytes

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Priority

This procedure explains how to configure priority on your device.

Note The device supports giving priority to specified queues. There are two priority levels available (1 and 2). Queues supporting voice and video should be assigned a priority level of 1.

Before you begin

You should have created a class map for priority before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_priority01`
	`Device(config-pmap)#`
		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:
	`Device(config-pmap)# class class_priority01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*—Class map name.
		• **class-default**—System default class matching any otherwise unclassified packets.

Step 4	**priority level***level_value*
	Example:
	`Device(config-pmap-c)# priority level 1`
	`Device(config-pmap-c)#`
		(Optional) The **priority** command assigns a strict scheduling priority for the class.
		The command options include:
		• **level** *level_value*—Specifies the multilevel (1-2) priority queue.
		Note Priority level 1 is more important than priority level 2. Priority level 1 reserves bandwidth that is processed first for QoS, so its latency is very low. Both priority level 1 and 2 reserve bandwidth.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring SGT based QoS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**class-map** *class-map-name* {**match-any** | **match-all** } **Example:** `Device(config)# class-map c1`	Specifies the class-map and enters class-map configuration mode.
Step 3	**match security-group source tag** *sgt-number* **Example:** `Device(config-cmap)# match security-group source tag 1000`	Configures the value for security-group source security tag.
Step 4	**match security-group destination tag** *dgt-number* **Example:** `Device(config-cmap)# match security-group destination tag 2000`	Configures the value for security-group destination security tag.

Step 5	**exit**
	Example:
	`Device(config-cmap)# exit`
	`Device#`
		Exits route-map configuration mode and returns to global configuration mode.

Step 6	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map pin`
	`Device(config-pmap)#`
		Specifies the policy-map and enters policy-map configuration mode.
		policy-map-name is the name of the child policy map. The name can be a maximum of 40 alphanumeric characters.

Step 7	**class** *class-name* **Example:** `Device(config-pmap)# class c1`	Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command

	Command or Action	Purpose
	`Device(config-pmap-c)#`	options for policy class map configuration mode include the following: • *word*—Class map name. • **class-default**—System default class matching any otherwise unclassified packets.
Step 8	**set dscp** *dscp-value* **Example:** `Device(config-pmap-c)# set dscp af11`	Configures the Differentiated Services CodePoint (DSCP) value.

Step 9	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes. Exits class-map configuration mode and enters global configuration mode.

Step 10	**interface** *interface-num* **Example:** `Device(config)# interface GigabitEthernet1/0/24`	Specifies the interface and enters the interface configuration mode.
Step 11	**service-policy** { **input** | **output** } *policy-map-name* **Example:** `Device(config-if)#service-policy input pin`	Assigns policy-map to the ingress of the interface.

Step 12	**end**
	Example:
	`Device(config-if)# end`
	`Device#`
		Saves configuration changes. Exits interface configuration mode and enters global configuration mode.

Configuration Example for SGT based QoS Classification

The following is an sample configuration for SGT based QoS on an interface:

```
ip access-list role-based sgt_acl
 10 permit ip
cts role-based sgt-map 24.0.0.0/8 sgt 24
cts role-based enforcement
cts role-based permissions from 24 to 24 sgt_acl

class-map match-all c1
 match protocol attribute business-relevance business-relevant
```

```
 match protocol attribute traffic-class ops-admin-mgmt
 match security-group destination tag 24
 match security-group source tag 24

policy-map pin
 class c1
  set dscp af11
 class class-default
  set dscp af12

interface GigabitEthernet1/0/24
 no switchport
 ip address 24.1.1.2 255.255.255.0
 service-policy input pin
 ip nbar protocol-discovery
```

Configuring Queues and Shaping

Configuring Egress Queue Characteristics

Depending on the complexity of your network and your QoS solution, you may need to perform all of the procedures in this section. You need to make decisions about these characteristics:

- Which packets are mapped by DSCP, CoS, or QoS group value to each queue and threshold ID?
- What drop percentage thresholds apply to the queues, and how much reserved and maximum memory is needed for the traffic type?
- How much of the fixed buffer space is allocated to the queues?
- Does the bandwidth of the port need to be rate limited?
- How often should the egress queues be serviced and which technique (shaped, shared, or both) should be used?

Note You can only configure the egress queues on the device.

Configuring Queue Buffers

The device allows you to allocate buffers to queues. If there is no allocation made to buffers, then they are divided equally for all queues. You can use the queue-buffer ratio to divide it in a particular ratio. Since by default DTS (Dynamic Threshold and Scaling) is active on all queues, these are soft buffers.

Note Queue-buffer ratio cannot be configured with a queue-limit.

Before you begin

The following are prerequisites for this procedure:

- You should have created a class map for the queue buffer before beginning this procedure.
- You must have configured either bandwidth, shape, or priority on the policy map prior to configuring the queue buffers.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_queuebuffer01`
	`Device(config-pmap)#`
		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:
	`Device(config-pmap)# class class_queuebuffer01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*—Class map name.
		• **class-default**—System default class matching any otherwise unclassified packets.

Step 4	**bandwidth** {*Kb/s* | **percent** *percentage* | **remaining** {**ratio** *ratio value* }}
	Example:
	`Device(config-pmap-c)# bandwidth percent 80`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map. The command parameters include:
		• *Kb/s*—Use this command to configure a specific value. The range is 20000 to 100000000.
		• **percent**—Allocates a minimum bandwidth to a particular class using a percentage. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is equally divided along all bandwidth queues.
		• **remaining**—Allocates a minimum bandwidth to a particular class. The queue

	Command or Action	Purpose
		can oversubscribe bandwidth in case other queues do not utilize entire port bandwidth. The total sum cannot exceed 100 percent. It is preferred to use this command when the **priority** command is used for certain queues in the policy. You can also assign ratios rather than a percentage to each queue; the queues will be assigned certain weights that are inline with these ratios. Ratios can range from 0 to 100. Total bandwidth ratio allocation for the policy in this case can exceed 100. **Note** You cannot mix bandwidth types on a policy map.

Step 5	**queue-buffers** {**ratio** *ratio value*}
	Example:
	`Device(config-pmap-c)# queue-buffers ratio 10`
	`Device(config-pmap-c)#`
		Configures the relative buffer size for the queue.
		Note The sum of all configured buffers in a policy must be less than or equal to 100 percent. Unallocated buffers are are evenly distributed to all the remaining queues. Ensure sufficient buffers are allocated to all queues including the priority queues.
		Note Protocol Data Units(PDUs) for network control protocols such as spanning-tree and LACP utilize the priority queue or queue 0 (when a priority queue is not configured). Ensure sufficient buffers are allocated to these queues for the protocols to function.

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 7	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Queue Limits

You use queue limits to configure Weighted Tail Drop (WTD). WTD ensures the configuration of more than one threshold per queue. Each class of service is dropped at a different threshold value to provide for QoS differentiation. With the device, each queue has 3 explicit programmable threshold classes—0, 1, 2. Therefore, the enqueue/drop decision of each packet per queue is determined by the packet's threshold class assignment, which is determined by the DSCP, CoS, or QoS group field of the frame header.

WTD also uses a soft limit, and therefore you are allowed to configure the queue limit to up to 400 percent (maximum four times the reserved buffer from common pool). This soft limit prevents overrunning the common pool without impacting other features.

Note You can only configure queue limits on the device egress queues on wired ports.

Before you begin

The following are prerequisites for this procedure:

- You should have created a class map for the queue limits before beginning this procedure.
- You must have configured either bandwidth, shape, or priority on the policy map prior to configuring the queue limits.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_queuelimit01`
	`Device(config-pmap)#`
		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:** `Device(config-pmap)# class class_queuelimit01`	Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:

	Command or Action	Purpose
	`Device(config-pmap-c)#`	• *word*—Class map name. • **class-default**—System default class matching any otherwise unclassified packets.

Step 4	**bandwidth** {*Kb/s* | **percent** *percentage* | **remaining** {**ratio** *ratio value* }}
	Example:
	`Device(config-pmap-c)# bandwidth 500000`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map. The parameters include:
		• *Kb/s*—Use this command to configure a specific value. The range is 20000 to 100000000.
		• **percent**—Allocates a minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is equally divided along all bandwidth queues.
		• **remaining**—Allocates a minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize entire port bandwidth. The total sum cannot exceed 100 percent. It is preferred to use this command when the **priority** command is used for certain queues in the policy. You can also assign ratios rather than a percentage to each queue; the queues will be assigned certain weights that are inline with these ratios. Ratios can range from 0 to 100. Total bandwidth ratio allocation for the policy in this case can exceed 100.
		Note You cannot mix bandwidth types on a policy map.

Step 5	**queue-limit** {*packets* **packets** | **cos** {*cos value* { *maximum threshold value* | **percent** *percentage* } | **values** {*cos value* | **percent** *percentage* } } | **dscp** {*dscp value* {*maximum threshold value* | **percent** *percentage*} | *match packet* {*maximum threshold value* | **percent** *percentage*} | **default** {*maximum threshold value* | **percent** *percentage*} | **ef** {*maximum threshold value* | **percent** *percentage*} | **dscp values** *dscp value*} | **percent** *percentage* }}	Sets the queue limit threshold percentage values. With every queue, there are three thresholds (0,1,2), and there are default values for each of these thresholds. Use this command to change the default or any other queue limit threshold setting. For example, if DSCP 3, 4, and 5 packets are being sent into a specific queue in a configuration, then you can use this command to set the threshold percentages for these three

	Command or Action	Purpose

	Example:
	`Device(config-pmap-c)# queue-limit dscp 3 percent 20`
	`Device(config-pmap-c)# queue-limit dscp 4 percent 30`
	`Device(config-pmap-c)# queue-limit dscp 5 percent 40`
		DSCP values. For additional information about queue limit threshold values, see Weighted Tail Drop, on page 55.
		Note The device does not support absolute queue-limit percentages. The device only supports DSCP or CoS queue-limit percentages.

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 7	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Proceed to configure any additional policy maps for QoS for your network. After creating your policy maps, proceed to attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Shaping

You use the **shape** command to configure shaping (maximum bandwidth) for a particular class. The queue's bandwidth is restricted to this value even though the port has additional bandwidth left. You can configure shaping as an average percent, as well as a shape average value in bits per second.

Before you begin

You should have created a class map for shaping before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**policy-map** *policy name* **Example:**	Enters policy map configuration mode.

	Command or Action	Purpose
	`Device(config)# policy-map policy_shaping01` `Device(config-pmap)#`	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:
	`Device(config-pmap)# class class_shaping01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*—Class map name.
		• **class-default**—System default class matching any otherwise unclassified packets.

Step 4	**shape average** {*target bit rate* | **percent** *percentage*}
	Example:
	`Device(config-pmap-c)# shape average percent 50`
	`Device(config-pmap-c)#`
		Configures the average shape rate. You can configure the average shape rate by target bit rates (bits per second) or by percentage of interface bandwidth for the Committed Information Rate (CIR).

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Sharped Profile Queuing

This procedure explains how to configure sharped profile queuing on your switch:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_shaping01`
	`Device(config-pmap)#`
		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.
		policy-map-name is the name of the child policy map. The name can be a maximum of 40 alphanumeric characters.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_shaping01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*—Class map name.
		• **class-default**—System default class matching any otherwise unclassified packets.

Step 4	**bandwidth** {*Kb/s* | **percent** *percentage* | **remaining** {**ratio** *ratio value*}}
	Example:
	`Device(config-pmap-c)# bandwidth 200000`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map. The parameters include:
		• *Kb/s*—Configures a specific value in kilobits per second (from 100 to 100000000).
		• **percent**-—Allocates minimum bandwidth to a particular class based on a percentage. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is equally divided along all bandwidth queues.
		• **remaining**— Allocates minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize entire port bandwidth. The total sum cannot exceed 100 percent. It is preferred to use this command when

	Command or Action	Purpose
		the **priority** command is used for certain queues in the policy. You can also assign ratios rather than percentages to each queue; the queues will be assigned certain weights which are inline with these ratios. Ratios can range from 1 to 65536. Total bandwidth ratio allocation for the policy in this case can exceed 100. **Note** You cannot mix bandwidth types on a policy map.

Step 5	**shape average** {*target bit rate* | **percent** *percentage*}
	Example:
	`Device(config-pmap-c)# shape average percent 50`
	`Device(config-pmap-c)#`
		Configures the average shape rate. You can configure the average shape rate by target bit rates (bits per second) or by percentage of interface bandwidth for the Committed Information Rate (CIR).

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Sharped Profile Queuing Configuration

The following is the example for sharped queuing:

```
Policy Map test
    Class test1
      bandwidth 20 (%)
      Average Rate Traffic Shaping
      cir 40%
    Class test3
      Average Rate Traffic Shaping
      cir 50%
    Class test2
      Average Rate Traffic Shaping
      cir 50%
    Class test4
      bandwidth 20 (%)
    Class test5
      Average Rate Traffic Shaping
      cir 70%
    Class test6
      Average Rate Traffic Shaping
      cir 60%
```

Monitoring QoS

The following commands can be used to monitor QoS on the device:

Table 11: Monitoring QoS

Command	Description
show class-map [*class_map_name*]	Displays a list of all class maps configured.
show policy-map [*policy_map_name*]	Displays a list of all policy maps configured. Command parameters include: • policy map name • interface • session
show policy-map session [**input** | **output** | **uid** *UUID*]	Displays the session QoS policy. Command parameters include: • input—Input policy • output—Output policy • uid—Policy based on SSS unique identification.
show table-map	Displays all the table maps and their configurations.

Configuration Examples for QoS

Examples: TCP Protocol Classification

TCP packets can be classified based on port numbers. The configuration for TCP protocol is as follows:

```
Device#show ip acce tcp
Extended IP access list tcp
    10 permit tcp any any eq 80
Device #
Device #show run class-map tcp

Current configuration : 63 bytes
!
class-map match-all tcp
 match access-group name tcp
!
end
```

```
Device #
Device #show run policy-map tcp

Current configuration : 56 bytes
!
policy-map tcp
 class tcp
  police 1000000000
!
end
Device #

Device #show run int tw 1/0/1

Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
 service-policy output tcp
end

Device #
```

Examples: UDP Protocol Classification

UDP packets can be classified based on port numbers. The configuration example for UDP protocol is as follows:

```
Device#show ip acce udp
Extended IP access list udp
    10 permit udp any any eq ntp
Device #

Device #show run class-map udp
Building configuration...

Current configuration : 63 bytes
!
class-map match-all udp
 match access-group name udp
!
end

Device #
Device #show run policy-map udp
Building configuration...

Current configuration : 56 bytes
!
policy-map udp
 class udp
  police 1000000000
!
end
Device #
Device #show run int tw 1/0/1

Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
```

```
 service-policy output udp
end

Device #
```

Examples: RTP Protocol Classification

RTP packets can be classified based on port numbers. The configuration example for RTP protocol is as follows:

```
Device# show ip access-list rtp
Extended IP access list rtp
    10 permit udp any any eq 554
    11 permit tcp any any eq 554
Device #

Device #show run class-map rtp

Current configuration : 63 bytes
!
class-map match-all rtp
 match access-group name rtp
!
end

Device #
Device #show run policy-map rtp

Current configuration : 56 bytes
!
policy-map rtp
 class rtp
  police 1000000000
!
end

Device #
Device #show run int tw 1/0/1

Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
 service-policy output rtp
end

Device #
```

Examples: Classification by Access Control Lists

This example shows how to classify packets for QoS by using access control lists (ACLs):

```
Device# configure terminal
Device(config)# access-list 101 permit ip host 12.4.1.1 host 15.2.1.1
Device(config)# class-map acl-101
Device(config-cmap)# description match on access-list 101
Device(config-cmap)# match access-group 101
Device(config-cmap)#
```

After creating a class map by using an ACL, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Class of Service Layer 2 Classification

This example shows how to classify packets for QoS using a class of service Layer 2 classification:

```
Device# configure terminal
Device(config)# class-map cos
Device(config-cmap)# match cos ?
  <0-7>  Enter up to 4 class-of-service values separated by white-spaces
Device(config-cmap)# match cos 3 4 5
Device(config-cmap)#
```

After creating a class map by using a CoS Layer 2 classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Class of Service DSCP Classification

This example shows how to classify packets for QoS using a class of service DSCP classification:

```
Device# configure terminal
Device(config)# class-map dscp
Device(config-cmap)# match dscp af21 af22 af23
Device(config-cmap)#
```

After creating a class map by using a DSCP classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: VLAN ID Layer 2 Classification

This example shows how to classify for QoS using a VLAN ID Layer 2 classification:

```
Device# configure terminal
Device(config)# class-map vlan-120
Device(config-cmap)# match vlan ?
  <1-4095>  VLAN id
Device(config-cmap)# match vlan 120
Device(config-cmap)#
```

After creating a class map by using a VLAN Layer 2 classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Classification by DSCP or Precedence Values

This example shows how to classify packets by using DSCP or precedence values:

```
Device# configure terminal
Device(config)# class-map prec2
Device(config-cmap)# description matching precedence 2 packets
```

```
Device(config-cmap)# match ip precedence 2
Device(config-cmap)# exit
Device(config)# class-map ef
Device(config-cmap)# description EF traffic
Device(config-cmap)# match ip dscp ef
Device(config-cmap)#
```

After creating a class map by using a DSCP or precedence values, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Hierarchical Classification

The following is an example of a hierarchical classification, where a class named parent is created, which matches another class named child. The class named child matches based on the IP precedence being set to 2.

```
Device# configure terminal
Device(config)# class-map child
Device(config-cmap)# match ip precedence 2
Device(config-cmap)# exit
Device(config)# class-map parent
Device(config-cmap)# match class child
Device(config-cmap)#
```

After creating the parent class map, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Hierarchical Policy Configuration

The following is an example of a configuration using hierarchical polices:

```
Device# configure terminal
Device(config)# class-map c1
Device(config-cmap)# match dscp 30
Device(config-cmap)# exit

Device(config)# class-map c2
Device(config-cmap)# match precedence 4
Device(config-cmap)# exit

Device(config)# class-map c3
Device(config-cmap)# exit

Device(config)# policy-map child
Device(config-pmap)# class c1
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# police rate percent 20 conform-action transmit exceed action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)# exit

Device(config-pmap)# class c2
Device(config-pmap-c)# bandwidth 20000
Device(config-pmap-c)# exit
Device(config-pmap)# class class-default
Device(config-pmap-c)# bandwidth 20000
Device(config-pmap-c)# exit
```

```
Device(config-pmap)# exit

Device(config)# policy-map parent
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average 1000000
Device(config-pmap-c)# service-policy child
Device(config-pmap-c)# end
```

The following example shows a hierarchical policy using table maps:

```
Device(config)# table-map dscp2dscp
Device(config-tablemap)# default copy
Device(config)# policy-map ssid_child_policy
Device(config-pmap)# class voice
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# police 15000000
Device(config-pmap)# class video
Device(config-pmap-c)# priority level 2
Device(config-pmap-c)# police 10000000
Device(config)# policy-map ssid_policy
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average 30000000
Device(config-pmap-c)# queue-buffer ratio 0
Device(config-pmap-c)# set dscp dscp table dscp2dscp
Device(config-pmap-c)# service-policy ssid_child_policy
```

Examples: Classification for Voice and Video

This example describes how to classify packet streams for voice and video using device specific information.

In this example, voice and video are coming in from end-point A into GigabitEthernet1/0/1on the device and have precedence values of 5 and 6, respectively. Additionally, voice and video are also coming from end-point B into FortyGigabitEthernet1/0/2 on the device with DSCP values of EF and AF11, respectively.

Assume that all the packets from the both the interfaces are sent on the uplink interface, and there is a requirement to police voice to 100 Mbps and video to 150 Mbps.

To classify per the above requirements, a class to match voice packets coming in on GigabitEthernet1/0/1 is created, named voice-interface-1, which matches precedence 5. Similarly another class for voice is created, named voice-interface-2, which will match voice packets in GigabitEthernet1/0/2. These classes are associated to two separate policies named input-interface-1, which is attached to GigabitEthernet1/0/1, and input-interface-2, which is attached toGigabitEthernet1/0/2. The action for this class is to mark the qos-group to 10. To match packets with QoS-group 10 on the output interface, a class named voice is created which matches on QoS-group 10. This is then associated to another policy named output-interface, which is associated to the uplink interface. Video is handled in the same way, but matches on QoS-group 20.

The following example shows how classify using the above device specific information:

```
Device(config)#
Device(config)# class-map voice-interface-1
Device(config-cmap)# match ip precedence 5
Device(config-cmap)# exit

Device(config)# class-map video-interface-1
Device(config-cmap)# match ip precedence 6
Device(config-cmap)# exit
```

```
Device(config)# class-map voice-interface-2
Device(config-cmap)# match ip dscp ef
Device(config-cmap)# exit

Device(config)# class-map video-interface-2
Device(config-cmap)# match ip dscp af11
Device(config-cmap)# exit

Device(config)# policy-map input-interface-1
Device(config-pmap)# class voice-interface-1
Device(config-pmap-c)# set qos-group 10
Device(config-pmap-c)# exit

Device(config-pmap)# class video-interface-1
Device(config-pmap-c)# set qos-group 20

Device(config-pmap-c)# policy-map input-interface-2
Device(config-pmap)# class voice-interface-2
Device(config-pmap-c)# set qos-group 10
Device(config-pmap-c)# class video-interface-2
Device(config-pmap-c)# set qos-group 20
Device(config-pmap-c)# exit
Device(config-pmap)# exit

Device(config)# class-map voice
Device(config-cmap)# match qos-group 10
Device(config-cmap)# exit

Device(config)# class-map video
Device(config-cmap)# match qos-group 20
Device(config)# policy-map output-interface
Device(config-pmap)# class voice
Device(config-pmap-c)# police 256000 conform-action transmit exceed-action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)# exit

Device(config-pmap)# class video
Device(config-pmap-c)# police 1024000 conform-action transmit exceed-action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)# exit
```

Examples: Average Rate Shaping Configuration

The following example shows how to configure average rate shaping:

```
Device# configure terminal
Device(config)# class-map prec1
Device(config-cmap)# description matching precedence 1 packets
Device(config-cmap)# match ip precedence 1
Device(config-cmap)# end

Device# configure terminal
Device(config)# class-map prec2
Device(config-cmap)# description matching precedence 2 packets
Device(config-cmap)# match ip precedence 2
Device(config-cmap)# exit

Device(config)# policy-map shaper
Device(config-pmap)# class prec1
```

```
Device(config-pmap-c)# shape average 512000
Device(config-pmap-c)# exit

Device(config-pmap)# policy-map shaper
Device(config-pmap)# class prec2
Device(config-pmap-c)# shape average 512000
Device(config-pmap-c)# exit

Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average 1024000
```

After configuring the class maps, policy map, and shape averages for your configuration, proceed to then apply the policy map to the interface for QoS.

Examples: Queue-limit Configuration

The following example shows how to configure a queue-limit policy based upon DSCP values and percentages:

```
Device# configure terminal
Device#(config)# policy-map port-queue
Device#(config-pmap)# class dscp-1-2-3
Device#(config-pmap-c)# bandwidth percent 20
Device#(config-pmap-c)# queue-limit dscp 1 percent 80
Device#(config-pmap-c)# queue-limit dscp 2 percent 90
Device#(config-pmap-c)# queue-limit dscp 3 percent 100
Device#(config-pmap-c)# exit

Device#(config-pmap)# class dscp-4-5-6
Device#(config-pmap-c)# bandwidth percent 20
Device#(config-pmap-c)# queue-limit dscp 4 percent 20
Device#(config-pmap-c)# queue-limit dscp 5 percent 30
Device#(config-pmap-c)# queue-limit dscp 6 percent 20
Device#(config-pmap-c)# exit

Device#(config-pmap)# class dscp-7-8-9
Device#(config-pmap-c)# bandwidth percent 20
Device#(config-pmap-c)# queue-limit dscp 7 percent 20
Device#(config-pmap-c)# queue-limit dscp 8 percent 30
Device#(config-pmap-c)# queue-limit dscp 9 percent 20
Device#(config-pmap-c)# exit

Device#(config-pmap)# class dscp-10-11-12
Device#(config-pmap-c)# bandwidth percent 20
Device#(config-pmap-c)# queue-limit dscp 10 percent 20
Device#(config-pmap-c)# queue-limit dscp 11 percent 30
Device#(config-pmap-c)# queue-limit dscp 12 percent 20
Device#(config-pmap-c)# exit

Device#(config-pmap)# class dscp-13-14-15
Device#(config-pmap-c)# bandwidth percent 10
Device#(config-pmap-c)# queue-limit dscp 13 percent 20
Device#(config-pmap-c)# queue-limit dscp 14 percent 30
Device#(config-pmap-c)# queue-limit dscp 15 percent 20
Device#(config-pmap-c)# end
Device#
```

After finishing with the above policy map queue-limit configuration, you can then proceed to apply the policy map to an interface for QoS.

Examples: Queue Buffers Configuration

The following example shows how configure a queue buffer policy and then apply it to an interface for QoS:

```
Device# configure terminal
Device(config)# policy-map policy1001
Device(config-pmap)# class class1001
Device(config-pmap-c)# bandwidth remaining ratio 10
Device(config-pmap-c)# queue-buffer ratio ?
  <0-100>  Queue-buffers ratio limit
Device(config-pmap-c)# queue-buffer ratio 20
Device(config-pmap-c)# end

Device# configure terminal
Device(config)# interface HundredGigabitE1/0/3
Device(config-if)# service-policy output policy1001
Device(config-if)# end
```

Examples: Policing Action Configuration

The following example displays the various policing actions that can be associated to the policer. These actions are accomplished using the conforming, exceeding, or violating packet configurations. You have the flexibility to drop, mark and transmit, or transmit packets that have exceeded or violated a traffic profile.

For example, a common deployment scenario is one where the enterprise customer polices traffic exiting the network towards the service provider and marks the conforming, exceeding and violating packets with different DSCP values. The service provider could then choose to drop the packets marked with the exceeded and violated DSCP values under cases of congestion, but may choose to transmit them when bandwidth is available.

Note The Layer 2 fields can be marked to include the CoS fields, and the Layer 3 fields can be marked to include the precedence and the DSCP fields.

One useful feature is the ability to associate multiple actions with an event. For example, you could set the precedence bit and the CoS for all conforming packets. A submode for an action configuration could then be provided by the policing feature.

This is an example of a policing action configuration:

```
Device# configure terminal
Device(config)# policy-map police
Device(config-pmap)# class class-default
Device(config-pmap-c)# police cir 1000000 pir 2000000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-dscp-transmit dscp table exceed-markdown-table
Device(config-pmap-c-police)# violate-action set-dscp-transmit dscp table
violate-markdown-table
Device(config-pmap-c-police)# end
```

In this example, the exceed-markdown-table and violate-mark-down-table are table maps.

Note Policer-based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Examples: Policer VLAN Configuration

The following example displays a VLAN policer configuration. At the end of this configuration, the VLAN policy map is applied to an interface for QoS.

```
Device# configure terminal
Device(config)# class-map vlan100
Device(config-cmap)# match vlan 100
Device(config-cmap)# exit
Device(config)# policy-map vlan100
Device(config-pmap)# policy-map class vlan100
Device(config-pmap-c)# police 100000 bc conform-action transmit exceed-action drop
Device(config-pmap-c-police)# end
Device# configure terminal
Device(config)# interface HundredGigabitE1/0/5
Device(config-if)#  service-policy input vlan100
```

Examples: Policing Units

The policing unit is the basis on which the token bucket works. CIR and PIR are specified in bits per second. The burst parameters are specified in bytes. This is the default mode; it is the unit that is assumed when no units are specified. The CIR and PIR can also be configured in percent, in which case the burst parameters have to be configured in milliseconds.

The following is an example of a policer configuration in bits per second. In this configuration, a dual-rate three-color policer is configured where the units of measurement is bits. The burst and peak burst are all specified in bits.

```
Device(config)# policy-map bps-policer
Device(config-pmap)# class class-default
Device(config-pmap-c)# police rate 100000 peak-rate 1000000
conform-action transmit exceed-action set-dscp-transmit dscp table
DSCP_EXCE violate-action drop
```

Examples: Single-Rate Two-Color Policing Configuration

The following example shows how to configure a single-rate two-color policer:

```
Device(config)# class-map match-any prec1
Device(config-cmap)# match ip precedence 1
Device(config-cmap)# exit
Device(config)# policy-map policer
Device(config-pmap)# class prec1
Device(config-pmap-c)# police cir 256000 conform-action transmit exceed-action drop
Device(config-pmap-c-police)# exit
```

```
Device(config-pmap-c)#
```

Examples: Dual-Rate Three-Color Policing Configuration

The following example shows how to configure a dual-rate three-color policer:

```
Device# configure terminal
Device(config)# policy-Map dual-rate-3color-policer
Device(config-pmap)# class class-default
Device(config-pmap-c)# police cir 64000 bc 2000 pir 128000 be 2000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-dscp-transmit dscp table exceed-markdown-table
Device(config-pmap-c-police)# violate-action set-dscp-transmit dscp table
violate-markdown-table
Device(config-pmap-c-police)# exit
Device(config-pmap-c)#
```

In this example, the exceed-markdown-table and violate-mark-down-table are table maps.

Note Policer based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Examples: Table Map Marking Configuration

The following steps and examples show how to use table map marking for your QoS configuration:

1. Define the table map.

   Define the table-map using the **table-map** command and indicate the mapping of the values. This table does not know of the policies or classes within which it will be used. The default command in the table map indicates the value to be copied into the ‘to’ field when there is no matching ‘from’ field. In the example, a table map named table-map1 is created. The mapping defined is to convert the value from 0 to 1 and from 2 to 3, while setting the default value to 4.

   ```
   Device(config)# table-map table-map1
   Device(config-tablemap)# map from 0 to 1
   Device(config-tablemap)# map from 2 to 3
   Device(config-tablemap)# default 4
   Device(config-tablemap)# exit
   ```

2. Define the policy map where the table map will be used.

   In the example, the incoming CoS is mapped to the DSCP based on the mapping specified in the table table-map1. For this example, if the incoming packet has a DSCP of 0, the CoS in the packet is set 1. If no table map name is specified the command assumes a default behavior where the value is copied as is from the ‘from’ field (DSCP in this case) to the ‘to’ field (CoS in this case). Note however, that while the CoS is a 3-bit field, the DSCP is a 6-bit field, which implies that the CoS is copied to the first three bits in the DSCP.

```
Device(config)# policy map policy1
Device(config-pmap)# class class-default
Device(config-pmap-c)# set cos dscp table table-map1
Device(config-pmap-c)# exit
```

3. Associate the policy to an interface.

```
Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# service-policy output policy1
Device(config-if)# exit
```

Example: Table Map Configuration to Retain CoS Markings

The following example shows how to use table maps to retain CoS markings on an interface for your QoS configuration.

The cos-trust-policy policy (configured in the example) is enabled in the ingress direction to retain the CoS marking coming into the interface. If the policy is not enabled, only the DSCP is trusted by default. If a pure Layer 2 packet arrives at the interface, then the CoS value will be rewritten to 0 when there is no such policy in the ingress port for CoS.

```
Device# configure terminal
Device(config)# table-map cos2cos
Device(config-tablemap)# default copy
Device(config-tablemap)# exit


Device(config)# policy map cos-trust-policy
Device(config-pmap)# class class-default
Device(config-pmap-c)# set cos cos table cos2cos
Device(config-pmap-c)# exit

Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# service-policy input cos-trust-policy
Device(config-if)# exit
```

Where to Go Next

Review the auto-QoS documentation to see if you can use these automated capabilities for your QoS configuration.

Additional References for QoS

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	*Command Reference (Catalyst 9400 Series Switches)* *Cisco IOS Quality of Service Solutions Command Reference*

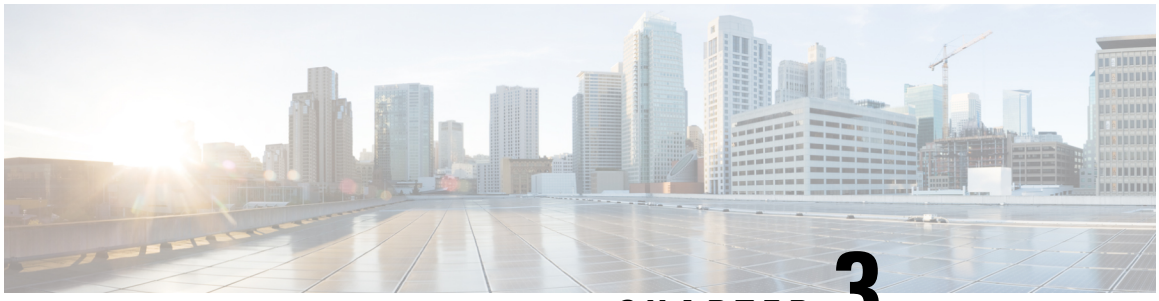
Feature History for QoS

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	QoS Functionality	QoS provides preferential treatment to specific types of traffic at the expense of other traffic types. Without QoS, the device offers best-effort service for each packet, regardless of the packet contents or size. **Note** This release does not support converged access.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 3

Configuring Weighted Random Early Detection

- Avoiding Network Congestion, on page 117
- Tail Drop, on page 117
- Weighted Random Early Detection, on page 117
- Limitations for WRED Configuration, on page 118
- Usage Guidelines for WRED, on page 119
- Configuring WRED, on page 120
- WRED Configuration Example, on page 124
- Support for WRED with Hierarchical QoS, on page 124
- Verifying WRED Configuration, on page 125
- Best Practices for WRED Configuration, on page 126
- Feature History for Weighted Random Early Detection, on page 128

Avoiding Network Congestion

Heterogeneous networks include different protocols used by applications, giving rise to the need to prioritize traffic in order to satisfy time-critical applications while still addressing the needs of less time-dependent applications, such as file transfer. If your network is designed to support different traffic types that share a single data path between devices in a network, implementing congestion avoidance mechanisms ensures fair treatment across the various traffic types and avoids congestion at common network bottlenecks. Congestion avoidance mechanism is achieved through packet dropping.

Random Early Detection (RED) is a commonly used congestion avoidance mechanism in a network.

Tail Drop

Tail drop treats all traffic equally and does not differentiate within a class of service. When the output queue is full and tail drop is in effect, packets are dropped until the congestion is eliminated and the queue is no longer full.

Weighted Random Early Detection

The RED mechanism takes advantage of the congestion control mechanism of TCP. Packets are randomly dropped prior to periods of high congestion. Assuming the packet source uses TCP, it decreases its transmission

rate until all the packets reach their destination, indicating that the congestion is cleared. You can use RED as a way to cause TCP to slow down transmission of packets. TCP not only pauses, but also restarts quickly and adapts its transmission rate to the rate that the network can support.

WRED is the Cisco implementation of RED. It combines the capabilities of RED algorithm with IP Precedence or Differentiated Services Code Point (DSCP) or Class of Service (COS) values.

How WRED Works

WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion. WRED drops some packets early rather than waiting until the queue is full. Thus it avoids dropping large number of packets at once and minimizes the chances of TCP global synchronization.

Approximate Fair Drop (AFD) is an Active Queue Management (AQM) algorithm that determines the packet drop probability. The probability of dropping packets depends upon the arrival rate calculation of a flow at ingress and the current queue length.

AFD based WRED is implemented on wired network ports.

AFD based WRED emulates the preferential dropping behavior of WRED. This preferential dropping behavior is achieved by changing the weights of AFD sub-classes based on their corresponding WRED drop thresholds. Within a physical queue, traffic with larger weight incurs less drop probability than that of smaller weight.

- Each WRED enabled queue has high and low thresholds.
- A sub-class of higher priority has a larger AFD weight.
- The sub-classes are sorted in ascending order, based on lowest of WRED minThreshold.

WRED Weight Calculation

AFD weight is calculated using low and high threshold values; AFD is an adjsuted index of the average of WRED high and WRED low threshold values.

When a packet arrives at an interface, the following events occur:

1. The drop probability is calculated. The drop probability increases as the AFD weight decreases. That means, if the average of low and high threshold values is less, the drop probability is more.
2. WRED consideres the priority of packet flows and the threshold values before deciding to drop the packet. The CoS, DSCP or IP Precedence values are mapped to the specified thresholds. Once these thresholds are exceeded, packets with the configured values that are mapped to these thresholds are eligible to be dropped. Other packets with CoS, DSCP or IP Precedence values assigned to the higher thresholds are en-queued. This process keeps the higher priority flows intact and minimises the latency in packet transmission.
3. If packets are not dropped using WRED, they are tail-dropped.

Limitations for WRED Configuration

- Weighted Tail Drop (WTD) is enabled by default on all the queues.

- WRED can be enabled / disabled per queue. When WRED is disabled, WTD is adapted on the target queue. Policy-map with WRED profile is configured only on physical ports as output policy.
- WRED is supported only in network port queues and is not supported on internal CPU queues and stack queues.
- Each WRED physical queue can support three subqueues, with unique WRED threshold pair configuration.
- Ensure that you configure bandwidth or shape in the policy-map along with WRED.
- Specify all the WRED thresholds only in percentage mode.
- Map the WRED threshold pairs by mapping class-map filter with corresponding match filters.

  We recommend the class-map with match “any” filter.
- WRED for priority traffic is not supported.
- WRED and queue limit are not supported for the same policy.
- Wired ports support a maximum of eight physical queues, of which you can configure WRED only on four physical queues, each with three threshold pairs. The remaining queues are configured with WTD. Policies with more than four WRED queues are rejected.

Usage Guidelines for WRED

To configure AFD based WRED feature, specify the policy map and add the class. Use the **random-detect** command to specify the method (using the dscp-based / cos-based / precedence-based arguments) that you want WRED to use to calculate the drop probability.

Note You can modify the policy on the fly. The AFD weights are automatically recalculated.

WRED can be configured for any kind of traffic like IPv4/IPv6, Multicast, and so on. WRED is supported on all 8 queueing classes.

Consider the following points when you are configuring WRED with **random-detect** command:

- With dscp-based argument, WRED uses the DSCP value to calculate drop probability.
- With cos-based argument, WRED uses the COS value to calculate drop probability.
- By default, WRED uses the IP Precedence value to calculate drop probability. **precedence-based** argument is the default and it is not displayed in the CLI.

  Note **show run policy-map** *policy-map* command does not display “precedence” though precedence is configured with **random-detect** command.
- The dscp-based and precedence-based arguments are mutually exclusive.
- Each of the eight physical queues can be configured with different WRED profiles.

Configuring WRED

Configuring WRED based on DSCP Values

Use the following steps to configure WRED profile based on DSCP values in packet mode:

Procedure

	Command or Action	Purpose
Step 1	**class-map** *match-criteria class-name* **Example:** `device(config)# class-map match-any CS`	Configures match criteria for the class map. Recommended match-criteria is match-any
Step 2	**match** *class-map-name* **Example:** `device(config-cmap)#match dscp CS1`	Match a class-map.
Step 3	**policy-map** *name* **Example:** `device(config)#policy-map PWRED`	Specifies the name of the WRED profile policy to be created.
Step 4	**class** *class-name* **Example:** `device(config-pmap)#class CS`	Specifies the name of the Class to be associated with the policy.
Step 5	Use either **bandwidth** {*kbps*| **remaining** *percentage* | **percent** *percentage*} or **shape** { **average** | **peak** }*cir* **Example:** `device(config-pmap-c)#bandwidth percent 10`	Specify either the bandwidth allocated for a class belonging to a policy map or the traffic shaping.
Step 6	**random-detect** *dscp-based* **Example:** `device(config-pmap-c)#random-detect dscp-based`	Configures WRED to use the DSCP value when it calculates the drop probability for the packet.
Step 7	**random-detect dscp** *dscp-value* **percent** *minThreshold maxThreshold* **Example:** `device(config-pmap-c)#random-detect dscp cs1 percent 10 20`	Specifies the minimum and maximum thresholds, in percentage.
Step 8	**interface** *interface-name* **Example:**	Enters the interface configuration mode.

	Command or Action	Purpose
	`device(config)#interface HundredGigE1/0/2`	
Step 9	**service-policy output** *policy-map* **Example:** `device(config-if)#service-policy output pwred`	Attaches the policy map to an output interface.

Configuring WRED based on Class of Service Values

Use the following steps to configure WRED profile based on Class of Service (COS) values in packet mode:

Procedure

	Command or Action	Purpose
Step 1	**class-map** *match-criteria class-name* **Example:** `device(config)# class-map match-any CS`	Configures match criteria for the class map. Recommended match-criteria is match-any
Step 2	**match** *class-map-name* **Example:** `device(config-cmap)#match cos 3`	Match a class-map.
Step 3	**policy-map** *name* **Example:** `device(config)#policy-map PWRED`	Specifies the name of the WRED profile policy to be created.
Step 4	**class** *class-name* **Example:** `device(config-pmap)#class CS`	Specifies the name of the Class to be associated with the policy.
Step 5	**bandwidth** {*kbps*| **remaining** *percentage* | **percent** *percentage*} **Example:** `device(config-pmap-c)#bandwidth percent 10`	Specifies the bandwidth allocated for a class belonging to a policy map.
Step 6	**random-detect** *cos-based* **Example:** `device(config-pmap-c)#random-detect cos-based`	Configures WRED to use the CoS value when it calculates the drop probability for the packet.
Step 7	**random-detect cos** *cos-value* **percent** *minThreshold maxThreshold* **Example:**	Specifies the minimum and maximum thresholds, in percentage.

	Command or Action	Purpose
	`device(config-pmap-c)#random-detect cos 3 percent 10 20`	
Step 8	**interface** *interface-name* **Example:** `device(config)# interface HundredGigE1/0/2`	Enters the interface configuration mode.
Step 9	**service-policy output** *policy-map* **Example:** `device(config-if)#service-policy output pwred`	Attaches the policy map to an output interface.

Configuring WRED based on IP Precedence Values

Use the following steps to configure WRED profile based on IP precedence values in packet mode:

Procedure

	Command or Action	Purpose
Step 1	**class-map** *match-criteria class-name* **Example:** `device(config)# class-map match-any CS`	Configures match criteria for the class map. Recommended match-criteria is match-any
Step 2	**match** *class-map-name* **Example:** `device(config-cmap)#match precedence 3`	Match a class-map.
Step 3	**policy-map** *name* **Example:** `device(config)#policy-map pwred`	Specifies the name of the WRED profile policy to be created.
Step 4	**class** *class-name* **Example:** `device(config-pmap)#class CS`	Specifies the name of the Class to be associated with the policy.
Step 5	**bandwidth** {*kbps*| **remaining** *percentage* | **percent** *percentage*} **Example:** `device(config-pmap-c)#bandwidth percent 10`	Specifies the bandwidth allocated for a class belonging to a policy map.
Step 6	**random-detect** *precedence-based* **Example:** `device(config-pmap-c)#random-detect precedence-based`	Configures WRED to use the IP precedence value when it calculates the drop probability for the packet.

	Command or Action	Purpose
Step 7	**random-detect precedence** *precedence-value* **percent** *minThreshold maxThreshold* **Example:** `device(config-pmap-c)#random-detect precedence 3 percent 10 20`	Specifies the minimum and maximum thresholds, in percentage.
Step 8	**interface** *interface-name* **Example:** `device(config)#interface HundredGigE1/0/2`	Enters the interface configuration mode.
Step 9	**service-policy output** *policy-map* **Example:** `device(config-if)#service-policy output pwred`	Attaches the policy map to an output interface.

Configure WRED under Class Default

You can configure WRED under a system-defined policy map, using the class-default class. A class-default traffic class is predefined, with match-any parameter.

Procedure

	Command or Action	Purpose
Step 1	**policy-map** *name* **Example:** `device(config)#policy-map pwred`	Specifies the name of the WRED profile policy to be created.
Step 2	**class class-default** **Example:** `device(config-pmap)#class class-default`	Creates a default class.
Step 3	**random-detect** *dscp-based* **Example:** `device(config-pmap-c)#random-detect dscp-based`	Configures WRED to use the DSCP value when it calculates the drop probability for the packet.
Step 4	**random-detect dscp** *dscp-value* **percent** *minThreshold maxThreshold* **Example:** `device(config-pmap-c)#random-detect dscp cs1 percent 10 20`	Specifies the minimum and maximum thresholds, in percentage. **Note** **random-detect range** CLI is not supported on Cisco IOS XE Release 16.6.1
Step 5	**interface** *interface-name* **Example:**	Enters the interface configuration mode.

	Command or Action	Purpose
	`device(config)#interface GigabitEthernet1/0/8`	
Step 6	**service-policy output** *policy-map* **Example:** `device(config-if)#service-policy output pwred`	Attaches the policy map to an output interface.

Example

WRED Configuration Example

The following example enables WRED to use DSCP profile for class CS. It configures three sub-classes cs1, cs2, and cs3 with their WRED minimum and maximum thresholds and finally applies the policy to Hundred Gigabit Ethernet interface 8:

```
Device(config)# class-map match-any CS
Device(config-cmap)# match dscp cs1
Device(config-cmap)# match dscp cs2
Device(config-cmap)# match dscp cs3
Device(config-cmap)# policy-map PWRED
Device(config-pmap)# class CS
Device(config-pmap-c)# bandwidth percent 10
Device(config-pmap-c)# random-detect dscp-based
Device(config-pmap-c)# random-detect dscp cs1 percent 10 20
Device(config-pmap-c)# random-detect dscp cs2 percent 20 30
Device(config-pmap-c)# random-detect dscp cs3 percent 34 44
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/8
Device(config-if)# service-policy output PWRED
```

Support for WRED with Hierarchical QoS

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

For HQoS, WRED is allowed only on the child policy and not on the parent policy. You can have the shaping configured on the parent policy and WRED on the child.

The following example configures the parent policy **pwred-parent** with traffic shaped on the basis of 10 percent of the bandwidth, that applies to its child, **pwred-child** configured for DSCP-based WRED.

```
policy-map PWRED-CHILD
 class CWRED
    bandwidth percent 10
    random-detect dscp-based
    random-detect dscp 1 percent 10 20
    random-detect dscp 10 percent 20 30
```

```
policy-map PWRED-PARENT
  class class-default
  shape average percent 10
  service-policy PWRED-CHILD
```

The following show command verifies the HQoS WRED configuration:

```
device# show policy-map PWRED-PARENT
  policy Map PWRED-PARENT
    class class-default
      average Rate Traffic Shaping
      cir 30%
      service-policy PWRED-CHILD
policy-map PWRED-CHILD
 class CWRED
  bandwidth percent 10
  random-detect dscp-based
  random-detect dscp 1 percent 10 20
  random-detect dscp 10 percent 20 30
policy-map PWRED-PARENT
 class class-default
   shape average percent 30
   service-policy PWRED-CHILD
```

Verifying WRED Configuration

Use the following show commands to verify WRED Configuration:

Procedure

Step 1 **show policy-map** *policy-map-name*

Displays the WRED and threshold labels.

Example:

```
Device# show policy-map PWRED
  Policy Map PWRED
    Class CS
      bandwidth 10 (%)
       percent-based wred

      dscp    min-threshold    max-threshold
      ------------------------------------
      cs1 (8)      10               20
      cs2 (16)     20               30
      cs3 (24)     34               44
      default (0)   -
```

Step 2 **show policy-map interface** *interface-name*

Displays WRED AFD Weights, WRED Enq(in Packets and Bytes), WRED Drops(in Packets and Bytes), Configured DSCP labels against the Threshold pairs.

Note Use this command only after you initiate the traffic. **show policy-map interface** is updated with WRED configuration only after a traffic is sent.

Example:

```
Device#show policy-map interface HundredGigE 1/0/2
 HundredGigE1/0/2

  Service-policy output: PWRED

    Class-map: CS (match-any)
      0 packets
      Match:  dscp cs1 (8)
      Match:  dscp cs2 (16)
      Match:  dscp cs3 (24)
      Queueing

      (total drops) 27374016
      (bytes output) 33459200081
      bandwidth 10% (1000000 kbps)

AFD WRED STATS BEGIN
Virtual Class   min/max        Transmit                    Random drop                 AFD
 Weight


   0          10 / 20         (Byte)33459183360           27374016                    12
                              (Pkts)522799759                427716
              dscp : 8


   1          20 / 30           Byte)0                      0                         20
                                 (Pkts)0                    0
              dscp : 16


   2          34 / 44         (Byte)16721                   0                         31
                                (Pkts)59                    0
              dscp : 24

     Total Drops(Bytes)   : 27374016

     Total Drops(Packets) : 427716
AFD WRED STATS END


    Class-map: class-default (match-any)
      0 packets
      Match: any


      (total drops) 0
      (bytes output) 192
```

Best Practices for WRED Configuration

- **Support for three WRED configuration pairs**

  Each WRED Physical Queue (AFD Queue) can support three WRED configuation pairs, with unique WRED threshold pair configuration.

```
Policy-map P1
    Class CS
        Random-detect dscp-based
```

```
        Random-detect dscp CS1 percent 10 20      // WRED pair 1
        Random-detect dscp CS2  percent 20 30      // WRED pair 2
        Random-detect dscp CS3 percent 30 40      // WRED pair 3
Class-map match-any CS
        match cs1
        match cs2
        match cs3
```

- **Appending WRED configuration pairs**

  You can add overlapping threshold pairs into the WRED configuration pairs.

```
Policy-map P1
    Class CS
       Random-detect dscp-based
       Random-detect dscp CS1 percent 10 20       // WRED pair 1
       Random-detect dscp CS2 percent 20 30       // WRED pair 2
       Random-detect dscp CS3 percent 30 40       // WRED pair 3
       Random-detect dscp CS4 percent  30 40     ==> belongs to WRED pair 3
       Random-detect dscp CS5 percent 20 30     ==> belongs to WRED pair 2
Class-map match-any CS
       match cs1
       match cs2
       match cs3
       match cs4  >>
       match cs5  >>
```

- **Default WRED pairs**

  If less than three WRED pairs are configured, any class-map filter participating WRED gets assigned to the third default WRED pair with maximum threshold (100, 100).

```
Policy-map P1
    Class CS
        Random-detect dscp-based
        Random-detect dscp CS1 percent 10 20   // WRED pair 1
        Random-detect dscp CS2 percent 20 30   // WRED pair 2
Class-map match-any CS
        match CS1
        match CS2
        match CS3
        match CS4
```

  In this case, classes CS3 and CS4 are mapped to WRED pair 3 with threshold (100, 100).

- **Rejection of Mismatched Configuration**

  If you configure random-detect without matching filters in a class-map, the policy installation is rejected.

```
Class-map match-any CS
       match CS1
       match CS2
       match CS5
Policy-map P1
    Class CS
        Shape average percent 10
        Random-detect dscp-based
        Random-detect dscp CS1 percent 10 20   // WRED pair 1
        Random-detect dscp CS2 percent 20 30   // WRED pair 2
       Random-detect dscp CS3 percent 30 40   // WRED pair 3  ===> Mismatched sub-class.
```

  When this policy is applied to the interface on the egress side, the policy fails during installation as the class-map values are incorrect:

```
device(config)# int Fo1/0/5
device(config-if)# service-policy output P1
device(config-if)#
*Feb 20 17:33:16.964: %IOSXE-5-PLATFORM: Switch 1 R0/0: fed: WRED POLICY INSTALL
FAILURE.Invalid WRED filter mark: 24 in class-map: CS
*Feb 20 17:33:16.965: %FED_QOS_ERRMSG-3-LABEL_2_QUEUE_MAPPING_HW_ERROR: Switch 1 R0/0:
 fed: Failed to detach queue-map for FortyGigabitEthernet1/0/5: code 2
```

Feature History for Weighted Random Early Detection

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	Weighted Random Early Detection mechanism	WRED is a mechanism to avoid congestion in networks. WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion, thus avoiding large number of packet drops at once. You can configure WRED to act based on any of the following values: • Differentiated Service Code Point • IP Precedence • Class of Service

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.