Configuring QoS on the FlexWAN and Enhanced FlexWAN Modules

This chapter describes how to configure Quality of Service (QoS) on the Cisco 7600 FlexWAN and Enhanced FlexWAN modules.

Note Cisco IOS Release 12.2SRA and later releases do not support the FlexWAN module or Supervisor Engine 2. These releases support the Enhanced FlexWAN module and the Sup720 and Sup32. In addition, note that Cisco IOS Release 12.2SRB introduced support for the Route Switch Processor 720 (RSP720).

This chapter contains the following sections:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

Understanding QoS on FlexWAN and Enhanced FlexWAN

Each FlexWAN and Enhanced FlexWAN module has both ingress and egress QoS capability.

Ingress QoS Capability

Because the FlexWAN and Enhanced FlexWAN modules provide for both policing and marking in the FlexWAN module itself, packets that are received through a FlexWAN port bypass the QoS functionality of the policy feature card (PFC). As a result, the QoS functionality of the PFC is not required. Policing and marking on the FlexWAN is configured by means of the Modular QoS command-line interface.

QoS features are performed by the CPUs on the FlexWAN and Enhanced FlexWAN modules; enabling more complex QoS features may affect port adapter performance.

The FlexWAN and Enhanced FlexWAN modules support the following QoS implementations on ingress ports:

- Classification
- Policing
- Marking
- CBWFQ
- Low latency queuing (LLQ)
- WRED
- Traffic shaping
- Percentage-based policing

Egress QoS Capability

The outbound QoS capabilities of the FlexWAN module include policing, marking, Class-Based Weighted Fair Queuing (CBWFQ), Weighted Random Early Detection (WRED), and traffic shaping. All features are configured using the Modular QoS command-line interface. As the features are implemented in Cisco IOS software, there are few limits on the complexity of the QoS configuration. However, as the complexity increases, the probability of a noticeable performance impact also increases.

The FlexWAN and Enhanced FlexWAN modules support the following QoS implementations on egress ports:

- Classification
- Policing
- Marking
- CBWFQ
- Low latency queuing (LLQ)
- WRED

- Traffic shaping
- Link fragmentation and interleaving (LFI)

Note Distributed Class-Based Weighted Fair Queueing (CBWFQ), Low Latency Queueing (LLQ), and Distributed Weighted Random Early Detection (dWRED) are also supported when present in a child policy that has shaping in the parent.

Note You configure FlexWAN and Enhanced FlexWAN QoS using a subset of the Modular QoS command-line interface (MQC). For information on MQC, refer to the *Modular Quality of Service Command-Line Interface Overview* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120xe/120xe5/mqc/
mcli.htm

Additional QoS Features and Resources

- For information about configuring Policy Feature Card QoS on the Cisco 7600 series router, see:
  - *Cisco 7600 Series Cisco IOS Software Configuration Guide*
    http://www.cisco.com/univercd/cc/td/doc/product/core/cis7600/software/122sx/swcg/index.htm
  - *Cisco 7600 Series Cisco IOS Command Reference*
    http://www.cisco.com/univercd/cc/td/doc/product/core/cis7600/software/122sx/cmdref/index.htm
- For information about configuring Policy Feature Card QoS on the Catalyst 6000 series switch running Cisco IOS software on the supervisor engine and on the MSFC, see:
  - *Catalyst 6500 Series Cisco IOS Software Configuration Guide*
    http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/122sx/swcg/index.htm
  - *Catalyst 6500 Series Cisco IOS Command Reference*
    http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/122sx/cmdref/index.htm
- For general information on how to configure Cisco IOS QoS, see:
  - *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2*
    http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/index.htm
  - *Cisco IOS Quality of Service Solutions Command Reference, Release 12.2*
    http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_r/index.htm

Classification

Classification entails using a traffic descriptor to categorize a packet within a specific group to define that packet and make it accessible for QoS handling on the network. Using packet classification, you can partition network traffic into multiple priority levels or classes of service.

For more information, see the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt1/qcfclass.htm#1000872

Configuring Classification

This section contains information for configuring classification for the FlexWAN and Enhanced FlexWAN QoS features.

The FlexWAN and Enhanced FlexWAN modules can classify traffic based on the following criteria:

- Differentiated Services Code Point (DSCP) (6-bits, defined in TOS byte—up to 64 values can be specified)
- IP Precedence (3-bits defined in the TOS byte in the IP header—up to 8 different precedence values can be specified)

Note Use the **match precedence** command for IPv6 classification.

- VLAN ID (Ethernet packets with specific VLAN IDs or VLAN IDs in a range)
- VLAN ID inner (VLAN ID in the 802.1Q header of bridged Ethernet packets)
- FR DLCI (Frame Relay DLCI value, which is 10 bits)
- ATM CLP bit (Cell Loss Priority bit in one or more of the ATM cells)
- Protocol (various protocols supported by the NBAR feature)
- MAC-layer address
- Frame Relay DE bit
- CoS bits (3-bit class of service field in the 802.1Q header)
- CoS bits inner (CoS bits in the 802.1Q header of bridged Ethernet packets)
- BGP index
- Packet length (match packets of a given size within a range)
- Access control list (ACL)
- EXP (3-bits used for classification in MPLS environment)

Use the **class-map** commands in global configuration mode to specify the name of the class map and define the class matching criteria.

Restrictions and Usage Guidelines

The classification restrictions and usage guidelines are as follows:

- **Traffic types** —The classification information in this section is for IP traffic. For information about configuring classification for MPLS, EoMPLS, and AToM traffic, refer to the "Configuring Multiprotocol Label Switching on FlexWAN and Enhanced FlexWAN Modules" section on page 3-1.
- **Traffic classes** —You can configure up to 255 discrete traffic classes in a single policy map, one class for each IP DSCP value. In addition to the traffic classes you specify, the class-default class is predefined when you create the policy map. It is the class to which traffic is directed if that traffic does not satisfy the match criteria of other classes defined in the policy map.

Configuration Tasks

To configure classification, perform the following steps in global configuration mode on the MSFC:

	Command	Purpose
Step 1	`Router(config)# class-map [match-all | match-any] class-name`	Creates a class map to be used for matching packets to a class.
Step 2	`Router(config-cmap)# match [ip dscp ip-dscp-value | ip precedence ip-precedence-value | mpls experimental mpls-exp-value]`	Specifies a specific IP DSCP, IP precedence, or MPLS EXP value as a match criterion.

This example shows how to configure a class map named ipp5, and enter a match statement for IP precedence 5:

```
Router(config)# class-map ipp5
Router(config-cmap)# match ip precedence 5
Router(config-cmap)#
```

This example shows how to display class-map information for a specific class map using the **show class-map** command:

```
Router# show class-map ipp5
Class Map match-all ipp5 (id 1)
Match ip precedence 5
```

Distributed Network-Based Application Recognition

Distributed Network-Based Application Recognition (DNBAR) is a classification engine used with the FlexWAN and Enhanced FlexWAN modules that recognizes a wide variety of applications, including web-based and other difficult-to-classify protocols that utilize dynamic TCP/UDP port assignments. When an application is recognized and classified by DNBAR, a network can invoke services for that specific application. DNBAR ensures that network bandwidth is used efficiently by working with QoS features to provide the following features:

- Guaranteed bandwidth
- Traffic shaping
- Traffic policing
- Packet marking

For more information, see *Distributed Network-Based Application Recognition* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121limit/121e/121e6/dnbar.htm

Policing

Policing rate limits a particular flow or group of flows using a token bucket policer. Packets not conforming to the user-specified rate and burst parameters are considered to be out-of-profile and are subject to either being dropped or marked down.

For more information, see the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt4/qcfpolsh.htm

The FlexWAN and Enhanced FlexWAN modules also support percentage-based policing.

Additionally, you can mark packets by setting the following:

- ATM Cell Loss Priority (CLP) bit
- Frame Relay Discard Eligibility (DE) bit
- IP precedence value
- IP differentiated services code point (DSCP) value
- MPLS experimental (EXP) value
- CoS bits (3-bit class of service field in the 802.1Q header)

Configuring Policing

Use the **police** command to enable policing on a class of traffic. The **police** command imposes a maximum rate on a particular class of traffic and provides the following actions if the rate is exceeded:

- Drop
- Transmit
- Transmit with re-marking

Note The PFC normally implements policing between ports in the router, however with FlexWAN and Enhanced FlexWAN modules, this feature is disabled for WAN ports. The FlexWAN and Enhanced FlexWAN modules handle policing because WAN packets generally have smaller headers than LAN packets, and the use of the PFC for policing in this manner would result in policing inaccuracies.

Restrictions and Usage Guidelines

The classification restrictions and usage guidelines are as follows:

- There are two forms of policers: single rate and dual rate. Single rate uses the committed information rate (CIR) to police traffic, while dual-rate uses a combination of CIR and peak information rate (PIR) to police traffic.
- With both forms of policers, there are three associated states of traffic: conform, exceed, and violate.
  - Traffic falls into the conform state if the bits-per-second rate has not been exceeded.
  - Traffic falls into the exceed state when the bits-per-second rate has been exceeded.
  - Traffic falls into the violate state when the bits-per-second rate is greater than the maximum-burst-bytes rate.

  For each of the above states, policing is accomplished through one of these actions: drop and transmit with re-marking or drop and transmit without re-marking.

For additional information, see *Traffic Policing* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t5/dtpoli.htm

Configuration Tasks

To configure policing, perform the following steps in global configuration mode on the MSFC:

	Command	Purpose
Step 1	`Router(config)# class-map [match-all | match-any] class-name`	Creates a class map to be used for matching packets to a class.
Step 2	`Router(config-cmap)# match [ip dscp ip-dscp-value | ip precedence ip-precedence-value | mpls experimental mpls-exp-value]`	Specifies a specific IP DSCP, IP precedence, or MPLS EXP value as a match criterion.
Step 3	`Router(config)# policy-map policy_name`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 4	`Router(config-pmap)# class class-name`	Defines the classes you want the service policy to contain.
Step 5	`Router(config-pmap-c)# police bps burst-normal burst-max conform-action action exceed-action action violate-action action`	Specifies the maximum bandwidth usage by a traffic class.

This example shows a traffic policing configuration with the average rate at 8000 bits per second, the normal burst size at 2000 bytes, and the excess burst size at 4000 bytes:

```
Router(config)# class-map acgroup2
Router(config-cmap)# match access-group 2
Router(config-cmap)# exit
Router(config)# policy-map police
Router(config-pmap)# class acgroup2
Router(config-pmap-c)# police 8000 2000 4000 conform-action transmit exceed-action
set-qos-transmit 4 violate-action drop
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface fastethernet 0/0
Router(config-if)# service-policy input police
```

Use the following commands to verify policing:

Command	Purpose
`Router# show policy-map`	Displays all configured policy maps.
`Router# show policy-map policy-map-name`	Displays the user-specified policy map.
`Router# show policy-map interface`	Displays statistics and configurations of all input and output policies that are attached to an interface.

This example shows how to display policing statistics using the **show policy-map interface** command in the EXEC mode.

```
Router# show policy-map interface
POS6/1/0
service-policy output: x
class-map: a (match-all)
0 packets, 0 bytes
5 minute rate 0 bps
match: ip precedence 0
police:
1000000 bps, 10000 limit, 10000 extended limit
```

```
conformed 0 packets, 0 bytes; action: transmit
exceeded 0 packets, 0 bytes; action: drop
        conformed 0 bps, exceed 0 bps, violate 0 bps
```

Marking

Associating a packet with an IP Precedence or IP DSCP marking allows users to classify traffic based on IP Precedence and IP DSCP value, depending on which value is marked. These markings can be used to identify traffic within the network, and other interfaces can match traffic based on the IP Precedence or DSCP markings.

For more information, see the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt1/qcfclass.htm
#998197

Configuring Marking

Marking sets various attributes of packets belonging to a particular class. You can mark IP and MPLS packets with the FlexWAN and Enhanced FlexWAN modules.

In Cisco IOS Release 12.2SR and later releases, you can also mark bridged Ethernet packets with the Enhanced FlexWAN module. See "Configuring QoS on Bridged Interfaces" section on page 28 for more information.

Note The FlexWAN and Enhanced FlexWAN modules always trust ToS. To change the ingress ToS, use marking.

Restrictions and Usage Guidelines

The marking restrictions and usage guidelines are as follows:

- You typically perform marking with the **set** command. The FlexWAN and Enhanced FlexWAN modules support the following forms of the **set** command:
  - Differentiated Services Code Point (DSCP) (6-bits, defined in the TOS byte—up to 64 values can be specified)
  - IP Precedence (3-bits defined in the TOS byte in the IP header—up to 8 different precedence values can be specified)
  - CoS bits (3-bit class of service field in the 802.1Q header)

Note The Enhanced FlexWAN module (beginning in Cisco IOS Release 12.2SR) also supports the marking of inner COS bits (for bridged Ethernet packets).

  - EXP (3-bits used for classification in MPLS environment)

Configuration Tasks

To configure marking, perform the following steps in global configuration mode on the MSFC:

	Command	Purpose
Step 1	`Router(config)# class-map [match-all | match-any] class-name`	Creates a class map to be used for matching packets to a class.
Step 2	`Router(config-cmap)# match [ip dscp ip-dscp-value | ip precedence ip-precedence-value | mpls experimental mpls-exp-value]`	Specifies a specific IP DSCP, IP precedence, or MPLS EXP value as a match criterion.
Step 3	`Router(config)# policy-map policy-name`	Specifies the name of the service policy to configure.
Step 4	`Router(config-pmap)# class class-name`	Specifies the name of a predefined class, which was defined with the **class-map** command, included in the service policy.
Step 5	`Router(config-pmap-c)# set ip precedence ip-precedence-value`	Specifies the IP precedence of packets within a traffic class. The *ip-precedence-value* is in the range 0 to 7.

This examples shows the creation of a service policy called policy1. This service policy is associated to a previously defined classification policy through the use of the **class** command. This example assumes that a classification policy called class1 was previously configured.

```
Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# set ip precedence 1
```

Use the following commands to verify marking:

Command	Purpose
`Router# show policy-map`	Displays all configured policy maps.
`Router# show policy-map policy-map-name`	Displays the user-specified policy map.
`Router# show policy-map interface`	Displays statistics and configurations of all input and output policies that are attached to an interface.

Congestion Management

Congestion management features allow you to control congestion by determining the order in which packets are sent out an interface based on priorities assigned to those packets. Congestion management entails the creation of queues, assignment of packets to those queues based on the classification of the packets, and scheduling of the packets in a queue for transmission. The congestion management QoS feature offers four types of queueing protocols, each of which allows you to specify creation of a different number of queues, affording greater or lesser degrees of differentiation of traffic, and to specify the order in which that traffic is sent.

For more information, see the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt2/qcfconmg.htm#1000872

The following sections describe congestion management on the FlexWAN and Enhanced FlexWAN modules:

Configuring Class-Based Weighted Fair Queuing

This section contains information for configuring Class-Based Weighted Fair Queueing (CBWFQ). CBWFQ provides guaranteed bandwidth rate to a non-priority class. Under congestion conditions, the class receives the guaranteed bandwidth. To configure CBWFQ, use one of the three forms of the **bandwidth** command:

- **bandwidth <x kbps>**—Minimum bandwidth guarantee of x kbps
- **bandwidth percent <x%>**—Minimum bandwidth guarantee of x% of link bandwidth
- **bandwidth remaining percent <x%>**—Minimum bandwidth guarantee of x% of remaining bandwidth in the link or the percentage bandwidth sharing of unused bandwidth among classes with bandwidth and bandwidth remaining configured

Note Low Latency Queueing (LLQ) provides guaranteed bandwidth for the priority classes. The sum of all bandwidth on a link guaranteed by CBWFQ for non-priority classes and LLQ for priority classes cannot exceed 99% of the total available link bandwidth. For more information on LLQ, see the "Configuring Low Latency Queuing" section on page 14.

The FlexWAN and Enhanced FlexWAN modules also support distributed CBWFQ. For more information, see *Distributed Class-Based Weighted Fair Queueing and Distributed Weighted Random Early Detection* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t5/dtcbwred.htm

Restrictions and Usage Guidelines

- **FlexWAN and Enhanced FlexWAN modules support**—Supports all forms of the **bandwidth** command except the **bandwidth remaining** form.
- **Physical interface**—Configuring CBWFQ on a physical interface is only possible if the interface is in the default queuing mode. Serial interfaces at E1 (2.048 Mbps) and below use WFQ by default; other interfaces use FIFO by default. Enabling CBWFQ on a physical interface overrides the default interface queuing method. Enabling CBWFQ on an ATM PVC does not override the default queuing method.
- If you configure a class in a policy map to use WRED for packet drop instead of tail drop, you must ensure that WRED is not configured on the interface to which you intend to attach that service policy.

- Traffic shaping and policing are not currently supported with CBWFQ.
- CBWFQ is supported on variable bit rate (VBR) and available bit rate (ABR) ATM connections. It is not supported on unspecified bit rate (UBR) connections.
- **Minimum bandwidth rate**—On the FlexWAN and Enhanced FlexWAN modules, the minimum CBWFQ rate is the greater of (a) 1 Kbps or (b) 1% of the link rate or the hierarchical shape rate.
- **Bandwidth allocation**—When a link is not experiencing congested conditions, the unused (or excess) bandwidth is shared among all classes. The excess bandwidth available to a class is in proportion to its guaranteed bandwidth specified by the **priority** or **bandwidth** commands. For example, if one class is guaranteed 20% of the link and a second class is guaranteed 10% of the link, then the first class receives twice as much excess bandwidth as the second class.
- **Using class-default**—The default queuing for class-default class is weighted fair queuing; at least 1% of the link bandwidth is always reserved for the default queuing. More bandwidth can be reserved using the **bandwidth** command.

Configuring a Service Policy in the Policy Map

To configure CBWFQ, use the Modular QoS command-line interface. Define the class of traffic with the **class-map** command, create a policy map that contains the **bandwidth** command, and apply the policy to the appropriate interface with the **service-policy** command.

To configure the class of traffic, see the "Configuring Classification" section on page 4. To configure a policy with CBWFQ and to assign the policy to an interface, perform the following tasks in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# class-map [match-all | match-any] class-name`	Creates a class map to be used for matching packets to a class.
Step 2	`Router(config-cmap)# match [ip dscp ip-dscp-value | ip precedence ip-precedence-value | mpls experimental mpls-exp-value]`	Specifies a specific IP DSCP, IP precedence, or MPLS EXP value as a match criterion.
Step 3	`Router(config)# policy-map policy-map`	Specifies the name of the service policy to be created or modified.
Step 4	`Router(config-pmap)# class class-name`	Specifies the name of the traffic class to be associated with the service policy.
Step 5	`Router(config-pmap-c)# bandwidth bandwidth-kbps | percent % of available bandwidth`	Specifies the percentage of available bandwidth in kilobits per second to be assigned to packets that meet the match criteria of the associated traffic class.
Step 6	`Router(config)# interface interface-name`	Specifies the interface to which the policy map will be applied.
Step 7	`Router(config-if)# service-policy [output policy-name]`	Attaches the specified policy map to the interface.

This example shows a service policy called policy1 that specifies the amount of bandwidth to allocate for traffic classes 1 and 2:

```
Router(config)# class-map class1
Router(config-cmap)# match ip dscp 30
Router(config-cmap)# exit
```

```
Router(config)# class-map class2
Router(config-cmap)# match ip dscp 10
Router(config-cmap)# exit

Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth 30000
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# class class2
Router(config-pmap-c)# bandwidth 20000
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)#

Router(config)# interface pos 2/0/0
Router(config-if)# service-policy output policy1
Router(config-if)# exit
```

Use the following commands to verify CBWFQ:

Command	Purpose
`Router# show policy-map policy-map`	Displays the configuration of all classes that make up the specified policy map.
`Router# show policy-map policy-map class class-name`	Displays the configuration of the specified class of the specified policy map.
`Router# show policy-map interface interface-name`	Displays the configuration of all classes configured for all policy maps on the specified interface.
`Router# show queue interface-type interface-number`	Displays queueing configuration and statistics for a particular interface.

Note The counters displayed after issuing the **show policy-map interface** command are updated only if congestion is present on the interface.

This example shows the information displayed when you enter the **show policy-map interface** command:

```
Router1-PE# show policy-map interface

POS6/2/0
service-policy output:

queue stats for all priority classes:
    queue size 0, queue limit 32655
    packets output 0, packet drops 0
    tail/random drops 0, no buffer drops 0, other drops 0

class-map:dscp0 (match-all)
    0 packets, 0 bytes
    30 second offered rate 0 bps, drop rate 0 bps
    match:ip dscp 0
    queue size 0, queue limit 610
    packets output 0, packet drops 0
    tail/random drops 0, no buffer drops 0, other drops 0
    shape:cir 2440000,  Bc 9760,  Be 9760
    (shape parameter is rounded to 2439000 due to granularity)
    output bytes 0, shape rate 0 bps
```

```
class-map:dscp1 (match-any)
  0 packets, 0 bytes
  30 second offered rate 0 bps, drop rate 0 bps
  match:ip dscp 1
    0 packets, 0 bytes
    30 second rate 0 bps
  queue size 0, queue limit 100000
  packets output 0, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
  bandwidth:kbps 400000, weight 64
  (bandwidth parameter is rounded to 397592 kbps due to granularity)

class-map:dscp2 (match-all)
  0 packets, 0 bytes
  30 second offered rate 0 bps, drop rate 0 bps
  match:ip dscp 2
  Priority:21% (130620 kbps), burst bytes 3265500, b/w exceed drops:0
  (Priority parameter is rounded to 129278 kbps due to granularity)

class-map:class-default (match-any)
  0 packets, 0 bytes
  30 second offered rate 0 bps, drop rate 0 bps
  match:any
    0 packets, 0 bytes
    30 second rate 0 bps
  queue size 0, queue limit 11422
  packets output 0, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
```

Configuring CBWFQ for MLPPP Links

Multilink Point-to-Point Protocol (MLPPP) allows multiple T1/E1 links to be bundled together to offer bandwidth greater than multiple T1s/E1s but less than a T3/E3. MLPPP with QoS supports CBWFQ, enabling MLPPP to carry voice and data on the same MLPPP bundle.

For information on configuring CBWFQ on MLPPP links, see the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/index.htm

Note The **ppp multilink interleave** and **ppp multilink fragment-delay** commands are required on a multilink interface only when Link Fragmentation and Interleaving (LFI) is also required.

Flow-Based Weighted Fair Queuing (WFQ)

Flow-based weighted fair queuing (WFQ) tries to ensure that reserved flows receive enough bandwidth and bounds latency to meet their minimum needs in the event of congestion. With standard WFQ, packets are queued by flow. Packets with the same source IP address, destination IP address, source Transmission Control Protocol (TCP) or UDP port, destination TCP or UDP port belong to the same flow.

For configuration information, see the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at:
http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt2/qcfwfq.htm#1000917

Configuring Low Latency Queuing

LLQ lets you specify low-latency behavior for a traffic class. LLQ allows delay-sensitive data to be given preferential treatment. You can give one or more classes priority status. You configure LLQ with the **priority** command.

The **priority** command configures guaranteed bandwidth to a priority class under worst-case congestion scenarios. It does not limit the bandwidth to the configured rate if the priority class is over subscribed. Typically, you would use the **priority** command in conjunction with an admission control mechanism that controls the amount of load offered to the priority class to avoid starvation of other classes.

Note CBWFQ and LLQ both provide guaranteed bandwidth for their respective classes. The sum of all bandwidth on a link guaranteed by CBWFQ for non-priority classes and LLQ for priority classes cannot exceed 99% of the total available link bandwidth. For more information on CBWFQ, see the "Configuring Class-Based Weighted Fair Queuing" section on page 10.

Restrictions and Usage Guidelines

The LLQ restrictions and usage guidelines are as follows:

- *FlexWAN and Enhanced FlexWAN modules priority command support*—All forms of the **priority** command are supported.
- *Bandwidth granularity*—On the FlexWAN and Enhanced FlexWAN modules, there is no restriction on the granularity of the priority rate.
- *Minimum priority rate*—The **priority** command has a minimum rate of 1 Kbps or 1% of the bandwidth.
- *Voice over IP (VoIP)*—LLQ supports Voice over IP (VoIP) on serial links and ATM permanent virtual circuits (PVCs). It does not support VoIP over Frame Relay links.
- *Priority matching*—If you use access control lists to configure matching port numbers, this feature provides priority matching for all port numbers, both odd and even numbers. Because voice typically exists on even port numbers, and control packets are generated on odd port numbers, control packets are also given priority when using this feature.

  On very slow links, giving priority to both voice and control packets may produce degraded voice quality. Therefore, if you are only assigning priority based on port numbers, you should use the **ip rtp priority** command instead of the priority command. (The **ip rtp priority** command provides priority only for even port numbers.)
- *Priority command restrictions*—The **random-detect** command, **queue-limit** command, and **bandwidth policy-map class configuration** command cannot be used while the **priority** command is configured.

  The **priority** command can be configured in multiple classes, but it should only be used for voice-like, constant bit rate (CBR) traffic.
- *Bandwidth allocation*—When a link is not under congested conditions, the unused (or excess) bandwidth is shared among all classes. The excess bandwidth available to a class is in proportion to its guaranteed bandwidth specified by the **priority** or **bandwidth** commands. For example, if one class is guaranteed 20% of the link and a second class is guaranteed 10% of the link, then the first class receives twice as much excess bandwidth as the second class.
- *Bandwidth command restriction*—You cannot configure the **bandwidth** command for a priority class.

Configuration Tasks

To configure LLQ, use the Modular QoS command-line interface. Define the class of traffic with the **class-map** command, create a policy map that contains the **priority** command, and apply the policy to the appropriate interface with the **service-policy** command.

To configure the class of traffic, see the "Configuring Classification" section on page 4. To configure a policy with LLQ and to assign the policy to an interface, perform the following tasks in global configuration mode:

	Command	Purpose
Step 1	Router(config)# **policy-map** *policy-name*	Specifies the name of the policy map to configure.
Step 2	`Router(config-pmap)# class class-map-name`	Specifies the name of a predefined class included in the service policy.
Step 3	`Router(config-pmap-c)# priority bandwidth-kbps | percent % of available bandwidth`	Gives priority to a class of traffic belonging to the policy map.
Step 4	`Router(config)# interface interface-name`	Specifies the interface to which the policy map will be applied.
Step 5	`Router(config-if)# service-policy [output policy-name]`	Attaches the specified policy map to the interface.

This example shows how to configure a priority queue reserved for traffic with an IP DSCP value of 40:

```
Router(config)# class-map gold-data
Router(config-cmap)# match-any ip dscp 40
Router(config-cmap)# exit
Router(config)# class-map match bar
Router(config-cmap)# match-any ip dscp 8
Router(config-cmap)# exit
Router(config)#
```

In the example, a priority queue for the class gold-data is reserved with a guaranteed allowed bandwidth of 50 Mbps, and a bandwidth of 20 Mbps is configured for the class bar. The **service-policy** command attaches the policy map to interface pos 4/1.

```
Router(config)# policy-map policy1
Router(config-pmap)# class gold-data
Router(config-pmap-c)# priority 50000
Router(config-pmap)# class bar
Router(config-pmap-c)# bandwidth 20000
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface pos 4/1/1
Router(config-if)# service-policy output policy1
Router(config-if)# exit
```

Use the following command to verify LLQ:

Command	Purpose
Router# ***show queue*** *interface-type interface-number*	Displays queueing configuration and statistics for a particular interface.

Configurable LLQ Burst Size

Configurable LLQ Burst Size extends the functionality available with LLQ. This feature allows customers to specify the Committed Burst (Bc) size in LLQ and, therefore, configure the network to accommodate temporary bursts of traffic.

For information on configuring LLQ burst size, see the *Configuring Burst Size in Low Latency Queueing* at:

http://www.cisco.com/en/US/products/sw/iosswrel/ps1834/products_feature_guide09186a0080080232.html

Configuring LLQ for MLPPP Links

MLPPP allows multiple T1/E1 links to be bundled together to offer bandwidth greater than multiple T1s/E1s but less than a T3/E3. MLPPP with QoS supports LLQ, enabling MLPPP to carry voice and data on the same MLPPP bundle.

For information on configuring LLQ on MLPPP links, see Configuring Distributed Low Latency Queueing and Other QoS Features in a Traffic Policy, page 20.

Note The **ppp multilink interleave** and **ppp multilink fragment-delay** commands are required on a multilink interface only when LFI is also required.

Distribution of Remaining Bandwidth

You can use MQC to specify how the *remaining bandwidth* is distributed among the output queues on a Cisco 7600 router interface or subinterface. Remaining bandwidth is the available bandwidth left on an interface or subinterface after all guaranteed traffic is accounted for. The amount of remaining bandwidth available for use is determined by the excess information rate (EIR) configured for the queue.

In MQC, the **bandwidth remaining percent** command allows you to configure the remaining bandwidth for output queues.

The following example shows how to use the **bandwidth remaining percent** command to distribute percentages of remaining bandwidth to various traffic classes in a policy map.

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# policy-map myPolicy
Router(config-pmap)# class class-default
Router(config-pmap-c)# bandwidth remaining percent 20
Router(config-pmap-c)# class prec1
Router(config-pmap-c)# bandwidth remaining percent 30
Router(config-pmap-c)# class prec2
Router(config-pmap-c)# bandwidth remaining percent 10
Router(config-pmap-c)# bandwidth percent 50
Router(config-pmap-c)# ^Z
Router#
20:44:36: %SYS-5-CONFIG_I: Configured from console by console
Router# show policy-map myPolicy
  Policy Map myPolicy
    Class prec1
         bandwidth remaining percent 30
    Class prec2
```

```
     bandwidth percent 50
     bandwidth remaining percent 10
Class class-default
     bandwidth remaining percent 20
```

Command Reference

To specify how the remaining bandwidth is distributed among the output queues on a Cisco 7600 series router interface or subinterface, use the MQC **bandwidth remaining percent** command in policy-map class configuration mode. To remove the percentage of remaining bandwidth specified for a traffic class, use the **no** form of this command.

- **bandwidth remaining percent** *percentage*
- **no bandwidth remaining percent** *percentage*

Syntax Description

percentage	Specifies a percentage value for the amount of guaranteed bandwidth, based on a relative percent of available bandwidth, to be assigned to the class. The percentage can be a number between 1 and 99.

Defaults

This command has no default behavior or values.

Command Modes

Modular QoS policy-map class configuration

Command History

12.2(18)SXE	This command was introduced on the Cisco 7600 series router.

Usage Guidelines

Use the MQC bandwidth remaining percent command to specify how the remaining bandwidth is distributed among the output queues on a Cisco 7600 series router interface or subinterface. Remaining bandwidth is the available bandwidth left on an interface or subinterface after all guaranteed traffic is accounted for.

The bandwidth remaining percent command allows you to configure the remaining bandwidth for output queues. The percentage parameter specified with the bandwidth remaining percent command is translated into an internal excess information rate (EIR) value between 0 and 255. The aggregate of all user-configured EIR bandwidth percentages cannot exceed 100 percent.

If the aggregate of all remaining bandwidth is less than 100 percent, the remainder is evenly split among user queues (including the default queue) that do not have a remaining bandwidth percentage configured. The minimum EIR value of each output queue is 1.

The EIR parameter for the network control queue is fixed at 128 and is not configurable.

If you have not configured a committed information rate (CIR) value for the default queue and it is the only user queue, the default queue receives half of the remaining bandwidth percentage of the network control queue.

Congestion Avoidance

Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottlenecks. Congestion avoidance is achieved through packet dropping. Among the more commonly used congestion avoidance mechanisms is Random Early Detection (RED), which is optimum for high-speed transit networks. Cisco IOS QoS includes an implementation of RED that, when configured, controls when the router drops packets. If you do not configure Weighted Random Early Detection (WRED), the router uses the cruder default packet drop mechanism called *tail drop*.

For more information, see the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt3/qcfconav.htm

The following sections describe congestion avoidance on the FlexWAN and Enhanced FlexWAN modules:

- Configuring Weighted Random Early Detection, page 18
- Distributed WRED, page 20
- DiffServ-Compliant WRED, page 20

Configuring Weighted Random Early Detection

Weighted Random Early Detection (WRED) is a congestion avoidance mechanism that takes advantage of the congestion control mechanism of TCP. By selectively dropping packets based on IP precedence prior to periods of high congestion, WRED tells the packet source to decrease its transmission rate. Edge routers assign IP Precedence to packets as they enter the network. WRED is useful on any output interface where you expect to have congestion. However, WRED is usually used in the core routers of a network rather than at the edge. WRED uses these precedences to determine how it treats different types of traffic.

When a packet arrives, the average queue size is calculated and one of the following events occurs:

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

Restrictions and Usage Guidelines

The WRED restrictions and usage guidelines are as follows:

- The **random-detect** command is used to enable WRED on a class of traffic. The FlexWAN and Enhanced FlexWAN modules support all three forms of the command:
  - **random-detect precedence-based**—Dropping based on IP Precedence bits

  - **random-detect dscp-based**—Dropping based on DSCP bits

- The parameters that can be tuned on a per-precedence, DSCP, or discard class basis include minimum and maximum thresholds, mark probability, and weight. When the average queue size is between the minimum and maximum thresholds, the packets are subjected to random discard. The mark probability controls the probability of dropping the packet when the queue size reaches the maximum threshold, and the weight determines the time constant needed to compute the average queue size.

For more details on the queue calculations and how WRED works, refer to the section "About WRED" in the chapter "Congestion Avoidance Overview" in the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at the following URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/index.htm

Configuration Tasks

To configure WRED, use the Modular QoS command-line interface.

1. Define the class of traffic with the **class-map** command.
2. Create a policy map that contains the **random-detect** command.
3. Apply the policy to the appropriate interface with the **service-policy** command.

To configure the class of traffic, see the "Configuring Classification" section on page 4.

To configure a policy with WRED and to assign the policy to an interface, perform the following steps in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# class-map [match-all | match-any] class-name`	Creates a class map to be used for matching packets to a class.
Step 2	`Router(config-cmap)# match [ip dscp ip-dscp-value | ip precedence ip-precedence-value | mpls experimental mpls-exp-value]`	Specifies a specific IP DSCP, IP precedence, or MPLS EXP value as a match criterion.
Step 3	`Router(config)# policy-map child-policy-name`	Specifies the name of the child policy map to configure.
Step 4	`Router(config-pmap)# class class-name`	Specifies the name of a predefined class included in the service policy.
Step 5	`Router(config-pmap-c)# bandwidth bandwidth-kbps | percent % of available bandwidth`	Specifies the percentage of available bandwidth in kilobits per second to be assigned to packets that meet the match criteria of the associated traffic class.
Step 6	`Router(config)# random-detect [dscp-based | prec-based]`	Enables WRED or distributed WRED (DWRED).

Configuration Example

This example shows how to configure WRED:

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# policy-map wred_test
Router(config-pmap)# class class-default
Router(config-pmap-c)# random-detect
```

```
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface pos 7/1/0
Router(config-if)# service-policy output wred_test
Router(config-if)# end
```

This example shows how to verify the configuration:

```
Router# show policy-map interface pos 7/1/0
 POS7/1/0

  Service-policy output: wred_test

    Class-map: class-default (match-any)
      16634097 packets, 8217243918 bytes
      30 second offered rate 482198000 bps, drop rate 0 bps
      Match: any
      queue size 0, queue limit 128
      packets output 16634097, packet drops 0
      tail/random drops 0, no buffer drops 0, other drops 0
      Random-detect:
        Exp-weight-constant: 3 (1/8)
        Mean queue depth: 0
        Class Random       Tail   Minimum   Maximum     Mark       Output
                drop       drop threshold threshold  probability  packets
        0     104806          0        32        64     1/10       3026812
        1     104569          0        36        64     1/10       3027050
        2     104732          0        40        64     1/10       3026884
        3     104169          0        44        64     1/10       3027449
        4     103047          0        48        64     1/10       3028569
        5     103156          0        52        64     1/10       3028460
        6          0          0        56        64     1/10             0
        7          0          0        60        64     1/10             0
```

Distributed WRED

Distributed WRED (dWRED) is useful on any output interface where you expect to have congestion. However, dWRED is usually used in the core routers of a network, rather than the edge. Edge routers assign IP precedences to packets as they enter the network. WRED uses these precedences to determine how to treat different types of traffic. For more information, see *Distributed Class-Based Weighted Fair Queueing and Distributed Weighted Random Early Detection* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t5/dtcbwred.htm

DiffServ-Compliant WRED

DiffServ Compliant WRED extends the functionality of WRED to enable support for DiffServ and Assured Forwarding (AF) Per Hop Behavior (PHB). This feature enables customers to implement AF PHB by coloring packets according to DSCP values and then assigning preferential drop probabilities to those packets.

This feature can be used with IP packets only. It is not intended for use with Multiprotocol Label Switching (MPLS)-encapsulated packets. For configuration information, see *Configuring Weighted Random Early Detection* at:

http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800c5d42.html#1002253

Traffic Shaping

A shaper typically delays excess traffic using a buffer, or queueing mechanism, to hold packets and shape the flow when the data rate of the source is higher than expected.

For more information, see the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt4/qcfpolsh.htm

The FlexWAN and Enhanced FlexWAN modules also support Distributed Traffic Shaping (DTS). For more information, see *Configuring Distributed Traffic Shaping* at: http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800bd8f1.html.

The following sections describe shaping on the FlexWAN and Enhanced FlexWAN modules:

- Configuring Traffic Shaping, page 21
- Configuring Hierarchical Traffic Shaping, page 22
- Configuring Queue Limit, page 24

Configuring Traffic Shaping

Traffic shaping allows us to impose a maximum rate for a particular class of traffic. The rationale for shaping is to smooth out bursty traffic into the network. It is accomplished by buffering. You perform traffic shaping with the **shape** command. These are the basic forms of the command:

- **shape average**—Bursty traffic is shaped on average to the maximum specified rate.
- **shape peak**—Bursty traffic is shaped to the maximum specified rate.

Restrictions and Usage Guidelines

The traffic shaping restrictions and usage guidelines are as follows:

- The minimum shaping rate is fixed by the command-line interface as 8,000 bps and is not dependent on the link rate.
- There is no minimum granularity on a shaper. The command-line interface allows a granularity up to 1 bps.
- A shaper value greater than link rate is allowed.

Configuration Tasks

Use MQC to configure traffic shaping. Create a class-map using the **class-map** command and a policy map using the **policy-map** command. Attach the class to the policy using the **class** command and then use the **shape** command to configure shaping for that class.

	Command	Purpose
Step 1	`Router(config)# class-map [match-all | match-any] class-name`	Creates a class map to be used for matching packets to a class.
Step 2	`Router(config-cmap)# match [ip dscp ip-dscp-value | ip precedence ip-precedence-value | mpls experimental mpls-exp-value]`	Specifies a specific IP DSCP, IP precedence, or MPLS EXP value as a match criterion.
Step 3	`Router(config)# policy-map policy-name`	Specifies the name of the policy map to configure.
Step 4	`Router(config-pmap)# class class-name`	Specifies the name of a predefined class included in the service policy.
Step 5	`Router(config-pmap-c)# shape [average | peak] mean-rate [[burst-size] [excess-burst-size]]`	Specifies the new values for the traffic shaping feature.

This example shows traffic shaping on a main interface; traffic leaving interface pos1/0/0 is shaped at the rate of 10 Mbps:

```
Router(config)# class-map class-interface-all
Router(config-cmap)# match any
Router(config-cmap)# exit
Router(config)# policy-map dts-interface-all-action
Router(config-pmap)# class class-interface-all
Router(config-pmap-c)# shape average 10000000
Router(config-pmap-c)# exit
Router(config)# interface pos1/0/0
Router(config-if)# service-policy output dts-interface-all-action
```

Use the following commands to verify traffic shaping:

Command	Purpose
`Router# show interface [interface-name] shape`	Displays detail status of the traffic shaping.
`Router# show policy policy-name`	Displays the configuration of all classes composing the specified traffic policy.
`Router# show policy policy-name class class-name`	Displays the configuration of the specified class of the specified traffic policy.

Configuring Hierarchical Traffic Shaping

Hierarchical traffic shaping allows multiple classes of traffic to be shaped at a single rate. A hierarchical traffic shaping policy consists of a child policy that identifies one or more classes of traffic, and a parent policy that shapes the output of the traffic classes into a single shape rate. You can apply a nested policy to an interface or subinterface.

Restrictions and Usage Guidelines

The hierarchical traffic shaping restrictions and usage guidelines are as follows:

- Hierarchical traffic shaping is supported on both input and output policies.
- Hierarchical traffic shaping is supported on the interfaces, subinterfaces, and PVCs for the FlexWAN and Enhanced FlexWAN modules.

Configuration Tasks

To configure hierarchical traffic shaping, use the Modular QoS command-line interface to define the parent and child policies. For the child policy, define the classes of traffic with the **class-map** command and create a policy with the **policy-map** command. For the parent policy, create a policy with the **policy-map** command, apply the child policy to the default class with the **service-policy** command, and apply the parent policy to the appropriate interface with the **service-policy** command.

To configure the classes of traffic, see the "Configuring Classification" section on page 4. To configure a child and parent policies for hierarchical traffic shaping, perform the following tasks in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# policy-map child-policy-name`	Specifies the name of the child policy map to configure.
Step 2	`Router(config-pmap)# class class-name`	Specifies the name of a predefined class included in the service policy.
Step 3	`Router(config-pmap-c)# priority bandwidth-kbps | percent % of available bandwidth`[1]	Gives priority to a class of traffic belonging to the policy map.
Step 4	`Router(config-pmap)# class class-name`	Specifies the name of the traffic class to be associated with the service policy.
Step 5	`Router(config-pmap-c)# bandwidth bandwidth-kbps | percent % of available bandwidth`[2]	Specifies the percentage of available bandwidth in kilobits per second to be assigned to packets that meet the match criteria of the associated traffic class.
Step 6	`Router(config)# policy-map parent-policy-name`	Specifies the name of the parent policy map to configure.
Step 7	`Router(config-pmap)# class class-name`	Specifies the name of a predefined class included in the service policy.
Step 8	`Router(config-pmap-c)# shape average cir`	Shapes traffic to the indicated bit rate for the specified class.
Step 9	`Router(config-pmap-c)# service-policy child-policy-name`	Links the parent and child policy and class.
Step 10	`Router(config)# interface interface-name`	Specifies the interface to which the policy map will be applied.
Step 11	`Router(config-if)# service-policy [output parent-policy-name]`	Attaches the specified nested parent and child policies to the interface.

1. Only the parameters shown are supported.
2. Only the parameters shown are supported.

This example shows a nested traffic policy configuration where traffic matching the class called "voice" will be guaranteed 3,200 Kbps, or 10% of the parent_policy's shape average:

```
Router(config)# class-map match-all voice
Router(config-cmap)# match ip dscp 5
Router(config-cmap)# exit

Router(config)# policy-map child_policy
Router(config-pmap)# class voice
```

```
Router(config-pmap-c)# priority percent 10
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# policy-map parent_policy
Router(config-pmap)# class class-default
Router(config-pmap-c)# shape average 32000000
Router(config-pmap-c)# service-policy child_policy
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# interface Serial6/1:1.1 point-to-point
Router(config-subif)# service-policy output parent_policy
```

Configuring Queue Limit

For the class-based traffic shaping and CBWFQ features, you can specify or modify the maximum number of packets the queue can hold for a class policy configured in a policy map using the **queue-limit** command from policy-map class configuration mode.

Restrictions and Usage Guidelines

The queue limit restrictions and usage guidelines are as follows:

- **FlexWAN and Enhanced FlexWAN queue limit values**—The default queue limit value is calculated as a function of line card type, buffers available, and the allocated CIR for that class. It is not chosen based on the bandwidth assigned to the traffic class or the amount of buffer memory. You can change the default queue limit to any value.

Configuration Tasks

To configure the queue limit, use the Modular QoS command-line interface. Define the class of traffic with the **class-map** command, create a policy-map that contains the **queue-limit** command, and apply the policy to the appropriate interface with the **service-policy** command.

To configure the class of traffic, see the "Configuring Classification" section on page 4. To configure a policy with the **queue-limit** command and assign the policy to an interface, perform the following tasks in global configuration mode:

	Command	Purpose
Step 1	Router(config)# **policy-map** *policy-name*	Specifies the name of the policy map to configure.
Step 2	`Router(config-pmap)# class` *class-name*	Specifies the name of a predefined class included in the service policy.
Step 3	`Router(config-if)# queue-limit` ***number-of-packets***	Specifies the maximum number of packets the queue can hold for a class policy configured in a policy map. The *number-of-packets* is 18, 25, 42, 128, 256, 512, 1024, 2000, 4000, 8000, 16000, or 32000, and specifies the maximum number of packets that the queue for this class can accumulate.

	Command	Purpose
Step 4	`Router(config)# interface interface-name`	Specifies the interface to which the policy map will be applied.
Step 5	`Router(config-if)# service-policy [input | output policy-name]`	Attaches the specified policy map to the interface.

Layer 2 QoS Applications

This section describes the QoS features for Layer 2 operations. It contains the following sections:

For information about how to configure QoS for interfaces that support bridging, see the "Configuring QoS on Bridged Interfaces" section on page 28.

Configuring QoS for ATM VC Access Trunk Emulation

Ingress and Egress QoS

The ingress QoS features are as follows:

- Aggregated ingress policy based on the bridged VLAN
- Ingress policy per dot1q VLAN per VC
- Ingress policy based on CoS value of the dot1q header

Note ATM VC access trunk emulation works in Trust CoS mode. CoS values of the dot1q header are set as the CoS values for the bridged VLAN.

Egress QoS features based on dot1q ID or CoS value of the dot1q header:

- Hierarchical traffic shaping
- Priority queuing
- Bandwidth commands

Configuration of QoS is shown in the following example:

```
Router(config)# interface atm3/0/0.100 point-to-point
Router(config-if)# pvc 1/21
Router(config-if-atm-vc)# bridge-domain 200 dot1q 100
Router(config-if-atm-vc)# bridge-domain 300 dot1q 101
Router(config-if-atm-vc)# service-policy in atm-policy-in
Router(config-if-atm-vc)# service-policy out atm-policy-out
```

As a result of this configuration of the VC, the **match vlan** or **match cos** parameters shown in the following example are the values from the dot1q header for 100 and 101.

```
Router(config)# class-map match-all match-cos-2
Router(config-cmap)# match cos inner 2
Router(config)# class-map match-all match-cos-3
Router(config-cmap)# match cos inner 3
```

```
Router(config)# class-map match-all match-vlan-101
Router(config-cmap)# match vlan inner 101
Router(config)# class-map match-all match-vlan-100
Router(config-cmap)# match vlan inner 100
!
Router(config)# policy-map atm-policy-out
Router(config-pmap)# class match-vlan-100
Router(config-pmap)# shape average 64000 256 256
Router(config-pmap)# class match-cos-2
Router(config-pmap)# priority percent 70
!
Router(config)# policy-map atm-policy-in
Router(config-pmap)# class match-cos-3
Router(config-pmap)# police 128000 4000 4000 conform-action transmit exceed-action drop
!
```

Verifying the Configuration

Use the following **show** commands to verify the configurations or counters:

```
Router# show class map
 Class Map match-any class-default (id 0)
   Match any

 Class Map match-all match-cos-2 (id 1)
   Match cos inner 2

 Class Map match-all match-cos-3 (id 2)
   Match cos inner 3

 Class Map match-all match-vlan-101 (id 4)
   Match vlan inner 101

 Class Map match-all match-vlan-100 (id 5)
   Match vlan inner 100

Router# show policy-map
 Policy Map atm-policy-out
   Class match-vlan-100
     shape average 64000 256 256
   Class match-cos-2
     priority percent 70
   Class class-default

   Policy Map atm-policy-in
     Class match-cos-3
        police 128000 4000 conform-action transmit exceed-action drop

Router# show policy-map interface
 ATM3/0/0.100: VC 1/21

  Service-policy input: atm-policy-in

    Class-map: match-cos-3 (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: cos inner 3
      Queueing
      queue size 0, queue limit 5
      packets input 0, packet drops 0
      tail/random drops 0, no buffer drops 0, other drops 0, flow drops 0
      police:
        128000 bps, 4470 limit
```

```
      conformed 0 packets, o bytes; action: transmit
      exceeded 0 packets, 0 bytes; action: drop
      conformed 0 bps, exceed 0 bps

  Class-map: class-default (match-any)
    51 packets, 6408 bytes
    30 second offered rate 0 bps, drop rate 0 bps
    Match: any

    queue size 0, queue limit 1772
    packets input, packet drops 0
    tail/random drops 0, no buffer drops 0, other drops 0, flow drops 0

Service-policy output: atm-policy-out

  queue stats for all priority classes:

    queue limit 8272 (packets)
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts queued/bytes queued) 0/0

  Class-=map match-vlan-100 (match-all)
    921 packets, 77748 bytes
    30 second offered rate 0 bps, drop rate 0 bps
    Match: vlan inner 100
    Queueing
    queue limit 5 (packets)
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts queued/bytes queued) 921/77748 shape (average) cir 64000 bc 256 be 256
    target shape rate 64000

  Class-map: match-cos-2 (match-all)
    0 packets, 0 bytes
    30 second offered rate 0 bps, drop rate 0 bps
    Match: cos inner 2
    Priority: 70 (%) (104832 kbps), burst 2620800 (bytes)

  Class-map: class-default (match-any)
    870 packets, 71340 bytes
    30 second offered rate 0 bps, drop rate 0 bps
    Match: any

    queue limit 1772 (packets)
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts queued/bytes queued) 870/71340
```

Configuring ATM Cell Loss Priority Setting

The ATM Cell Loss Priority (CLP) Setting feature allows users to control the CLP bit setting on routers with the PA-A3 and PA-A6 port adapters.

When creating a class map, the following commands are supported:

- match ip precedence
- match ip dscp
- match protocol
- match any

Configure the CLP setting feature as described at the following URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s/120s7/atmclp.htm

Configuring QoS on Bridged Interfaces

This section describes the QoS features available for interfaces that support bridging and provides information about how to configure QoS on bridged interfaces. It contains the following sections:

QoS Over Bridging

In Cisco IOS Release 12.2SR and later releases, the Enhanced FlexWAN module supports the QoS over bridging feature, which provides QoS services to bridged packets carrying Ethernet frames. The QoS over bridging feature is available on interfaces that are configured for one of the following encapsulation types:

- ATM RFC 1483 (point-to-point bridging, multipoint bridging, and multi-VLAN in single VC)
- Frame Relay RFC 1490 (point-to-point bridging and multipoint bridging)
- PPP/BCP (trunk and single-VLAN mode)
- BRE (bridged routing encapsulation) over ATM PVCs
- RBE (routed bridged encapsulation) over ATM PVCs
- Half-bridging over ATM PVCs

The QoS over bridging feature allows you to classify bridged Ethernet packets into different traffic classes and perform QoS based on packet classification. This way you can apply different QoS features to different classes of traffic.

QoS over Bridging Features

You can apply the following QoS features to bridged Ethernet packets:

- Classification (ingress)
  - match VLAN ID (vlan-inner)
  - match L2 802.1q CoS bits (cos-inner)
  - match FR DLCI
  - match FR DE
  - match ATM CLP
  - match IP DSCP
  - match IP precedence
- Classification (egress)
  - match VLAN ID (vlan-inner)
  - match L2 802.1q CoS bits (cos-inner)

  - match FR DLCI
  - match IP DSCP
  - match IP precedence
- Marking (ingress)
  - set L2 802.1q CoS bits (cos-inner)
- Marking (egress)
  - set FR DE
  - set ATM CLP
  - set L2 802.1q CoS bits (cos-inner)
- LLQ, CBWFQ, and WRED (egress)
- Shaping
- Policing (ingress and egress)
- Hierarchical QoS (HQoS)

Note The terms vlan-inner and cos-inner refer to the VLAN ID and CoS bits in the Ethernet header of a bridged packet. The terms do not refer to the inner dot1q (802.1q) tag of an 802.1q Q-in-Q packet.

For a list of QoS over bridging classification and marking statements, see Table 4 and Table 5.

Fast-Classification and Standard Matching Algorithms

Depending on the type of **match** statements included in a class map, the QoS over bridging feature uses one of the following types of matching algorithms to determine whether a packet matches any of the filters in the class map:

- Standard matching algorithm—class maps for matching ATM or Frame Relay packets (ATMCLP, FR-DE, FR-DLCI). This algorithm compares each packet to **match** statements sequentially, one-at-a-time, until a match is found. QoS policies for that traffic class are then applied to the packet.
- Fast-classification algorithm—class maps for matching packets based on VLAN inner, COS inner, IP DSCP, or IP precedence.

  Information about traffic classes is stored in fast-classification tables, which are arrays that are indexed on the filter criteria (that is, the **match** statements). These arrays contain the QoS policies to apply to packets that match the filter criteria. The fast-classification matching algorithm performs a single lookup in the fast-classification table. If a match is found, the QoS policies for that traffic class are applied to the packet. The fast-classification process is used in both ingress and egress directions.

Because the standard matching algorithm compares packets against **match** statements sequentially, overlapping match criteria are allowed. For example, the same ATM CLP, FR-DE, or FR DLCI value can be used in the **match** statements for multiple traffic classes. However, because the fast-classification algorithm performs a single table lookup, overlapping values are not allowed. If overlapping values are used, the system reverts to the standard matching algorithm instead of the fast-classification algorithm.

Restrictions and Usage Guidelines

The QoS over bridging feature has the following restrictions and usage guidelines:

- Supported on the Enhanced FlexWAN module, in Cisco IOS Release 12.2SR and later releases.
- Supported on the Supervisor Engine 720, Supervisor Engine 32 (in Release 12.2(18)SXF and later), and Route Switch Processor 720 (in Release 12.2SRB and later).
- The system performs a single (fast-classification table) lookup on **vlan inner**, **cos inner**, **IP DSCP**, and **IP precedence** values. This fast-classification lookup does not support overlapping values for these keywords. This means that you cannot use the same value for a particular keyword in more than one **match** statement in a single class map. If overlapping values are used, the system reverts to the standard matching algorithm, in which each **match** statement in the class map is examined sequentially, one-at-a-time, until a match is found.
- Tagged and untagged packets are supported. Note however, that untagged packets in the ingress direction do not support the **match cos inner** option. This is because an untagged packet does not contain an 802.1q header with the CoS bits. In the egress direction, **match cos inner** is supported on untagged packets as long as the class map does not contain overlapping values.
- All QoS restrictions and usage guidelines apply (see the "Understanding QoS on FlexWAN and Enhanced FlexWAN" section on page 2 for details).
- For ATM (RFC 1483) and Frame Relay (RFC 1490) interfaces configured for multipoint bridging, MPB restrictions and usage guidelines apply (see the "Configuring Multipoint Bridging" section on page 2-74 for details).
- Due to restrictions with the TCAM, the following match statements cannot be used in a policy map that contains IPv4 or IPv6 match ACL statements:
  - match input vlan
  - match vlan
  - match cos
  - match vlan inner
  - match cos inner
- Only ATM PVCs (and not SVCs) support the QoS over bridging feature.
- Weighted random early detection (WRED) is not supported on bridged ATM VCs.
- Marking either the IP DSCP or IP precedence field in bridged frames is not supported. (CSCsd55169 has been opened for IP precedence marking.)

Configuring QoS on Bridged Interfaces

To configure a policy map that provides QoS features for bridged packets carrying Ethernet frames, follow these steps. See the "QoS over Bridging Configuration Examples" section on page 36 for configuration examples.

Note These instructions highlight the matching and marking options available for QoS on bridged packets. Standard QoS options can also be used in policy maps for bridged packets.

	Command	Purpose
Step 1	`Router(config)# class-map [match-all | match-any] class-name`	Creates a class map that assigns packets to a QoS traffic class. The **match-all** option requires that a packet meet all **match** statements to be assigned to this traffic class. The **match-any** option allows a match on any statement.
Step 2	`Router(config-cmap)# match {vlan inner vlan-value | cos inner cos-value | match ip dscp dscp-value | match ip precedence prec-value | match fr-de | match fr-dlci dlci-value | match atm clp }`	Specifies a value to match (see Table 4 for details). Specify one or more **match** statements to define the characteristics of traffic to assign to this traffic class. Enter the **exit** command when you are done.
Step 3	`Router(config)# policy-map policy-name`	Creates a policy map that defines the QoS actions to apply to the traffic classes in this policy map.
Step 4	`Router(config-pmap)# class class-name`	Specifies the name of a traffic class to perform QoS on. The traffic class must already exist (Step 1).
Step 5	Specify one or more QoS actions (such as marking, policing, and queuing) to perform on packets in this traffic class, and enter the **exit** command when you are done. Repeat Steps 4 through 7 for other traffic classes that are part of this policy. When you are done, enter the **exit** command twice to return to configuration mode. **Note** Steps 6 and 7 list the policy map commands with QoS over bridging options. See Table 5 for details about these commands. See Table 6 for a list of additional QoS over bridging features that you can include in the policy map.	
Step 6	`Router(config-pmap-c)# set {cos-inner cos-value | set fr-de | set atm clp}`	Marks the CoS bits, Frame Relay discard eligibility (FR-DE) bit, or ATM cell loss priority (CLP) bit of packets in this traffic class (see Table 5 for details). You can also use the **set-cos-inner-transmit** option in a **police** statement.
Step 7	`Router(config-pmap-c)# police cir cir-bps [bc conform-burst-size-bytes] [pir pir-bps] [be peak-burst-size-bytes] [conform-action action] [exceed-action action] [violate-action action]`	Configures traffic policing, which sets the traffic rate for a traffic class and specifies whether to drop, transmit, or transmit with re-marking packets that conform, exceed, or violate the specified traffic rate. This command syntax is for two-rate policing, with committed and peak information rates (CIR, PIR). In addition to the standard options for **action**, the following option is available for bridged packets: • **set-cos-inner-transmit** *cos-value*—Sets the CoS bits in the Ethernet header.
Step 8	`Router(config)# interface interface-name`	Specifies the interface to attach the policy map to. The interface must be configured to support bridging.
Step 9	`Router(config-if)# service-policy {input | output} policy-name`	Attaches the specified policy map (*policy-name*) to the interface, and specifies whether to apply the QoS policy to ingress or egress packets on the interface.

QoS over Bridging Class-Map and Policy-Map Statements

Table 4 and Table 5 list the class-map and policy-map statements for QoS over bridging features. The tables list the types of interfaces that support each statement; the terms ingress and egress refer to whether the statement applies to incoming (ingress) or outgoing (egress) packets on the interface. See the "QoS over Bridging Configuration Examples" section on page 36 for configuration examples.

Table 4 QoS over Bridging Match Statements (Class Map)

Match Statements	Purpose
match vlan inner *vlan-value*	Matches packets whose VLAN ID (in the Ethernet header) is the same as *vlan-value*.
	Interfaces (ingress and egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk, tunnel)
match cos inner *cos-value*	Matches packets whose CoS bits are the same as *cos-value* (valid values are 0 through 7, as per IEEE Standard 802.1Q). The CoS bits are part of the Ethernet header. **Note** To use this option in the ingress direction, the interface must be configured for tagged packets (since untagged packets do not contain an 802.1q header). In the egress direction, untagged packets are supported as long as the policy map does not contain any overlapping match criteria. (This means that you cannot use the same **match** value for multiple traffic classes in a single policy map.)
	Interfaces (ingress and egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk)
match ip dscp *dscp-value*	Matches packets whose IP differentiated services code point (DSCP) is the same as *dscp-value* (valid values are 0 through 63).
	Ingress: ATM, Frame Relay, PPP/BCP (single-VLAN, trunk) Egress: BRE Ingress and egress: RBE, Half-bridging
match ip precedence *prec-value*	Matches packets whose IP precedence value is the same as *prec-value* (valid values are 0 through 7). (Ingress and egress)
	Ingress: ATM, Frame Relay, PPP/BCP (single-VLAN, trunk) Egress: BRE Ingress and egress: RBE, Half-bridging
match fr-de	Matches Frame Relay packets whose discard eligibility bit (FR-DE bit) is set.
	Interfaces (ingress): Frame Relay
match fr-dlci *dlci-value*	Matches Frame Relay packets whose data-link connection identifier (DLCI) is the same as *dlci-value*.
	Interfaces (ingress and egress): Frame Relay
match atm clp	Matches ATM packets whose cell loss priority (CLP) bit is set.
	Interfaces (ingress): ATM, BRE, RBE, Half-bridging

Table 5 lists the marking statements available for QoS on bridged packets.

Table 5 QoS over Bridging Marking Statements (Policy Map)

Marking Statements	Purpose
set cos-inner *cos-value*	Sets the CoS bits to *cos-value* (valid values are 0 through 7, as per IEEE Standard 802.1Q). The CoS bits are part of the Ethernet header. **Note** To use this option in the egress direction, the interface must be configured for tagged packets (since untagged packets do not contain an 802.1q header). To use this option in the ingress direction, the packet's egress port (which can be any port on the router) must be configured for 802.1q tagging; otherwise, the CoS bits are not valid.
	Interfaces (ingress and egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk)
set-cos-inner-transmit *cos-value*	(Include this in the *action* portion of the **police** command.) Sets the CoS bits of an egress packet in response to a police action (where *cos-value* is 0 through 7, as per IEEE 802.1Q). **Note** To use this option in the egress direction, the interface must be configured for tagged packets (since untagged packets do not contain an 802.1q header). To use this option in the ingress direction, the packet's egress port (which can be any port on the router) must be configured for 802.1q tagging; otherwise, the CoS bits are not valid.
	Interfaces (ingress and egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk)
set fr-de	Sets the discard eligibility (FR-DE) bit of egress Frame Relay packets to 1.
	Interfaces (egress): Frame Relay
set atm clp	Sets the cell loss priority (CLP) bit of egress ATM packets.
	Interfaces (egress): ATM, BRE, RBE, Half-bridging

Table 6 lists the other QoS features that are available for bridged packets. Examples of most of these features are provided in the "QoS over Bridging Configuration Examples" section on page 36.

Table 6 Additional QoS Features for Bridged Packets (Policy Map)

QoS Feature	Purpose
Traffic shaping (**shape** command)	Allows you to control the speed of traffic entering or leaving an interface. Shaping can be used to match traffic flow to interface speed and to ensure that service agreements are adhered to. It can also help control traffic bottlenecks. See the "Traffic Shaping" section on page 21 for more information.
	Interfaces (ingress and egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk), BRE, RBE, Half-bridging
Traffic policing (**police** command)	Allows you to control the speed of traffic entering or leaving an interface, and to specify whether to drop or transmit packets that conform, exceed, or violate the specified traffic rate. In addition, policing allows you to change the precedence of packets before transmitting them. See the "Configuring Policing" section on page 6 for more information.
	Interfaces (ingress and egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk), BRE, RBE, Half-bridging
Low latency queuing (**priority** command)	LLQ provides strict priority queuing on ATM VCs and serial interfaces. See the "Configuring Low Latency Queuing" section on page 14 for more information.
	Interfaces (egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk), BRE, RBE, Half-bridging
Class-based weighted fair queuing (**bandwidth** command)	CBWFQ allows you to allocate the bandwidth on an interface among several classes of traffic. Traffic is then assigned bandwidth based on the traffic class it belongs to. See the "Configuring Class-Based Weighted Fair Queuing" section on page 10 for more information.
	Interfaces (egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk), BRE, RBE, Half-bridging
Hierarchical QoS (HQoS)	Allows you to configure a policy map (parent) that contains other policy maps (children).
	Interfaces (ingress and egress): ATM, Frame Relay, PPP/BCP (single-VLAN, trunk), BRE, RBE, Half-bridging

Sample Interface Configurations

Following are several sample configurations for the types of interfaces that support the QoS over bridging feature:

ATM (RFC 1483) Interface:

```
interface atm3/0/0
pvc 10/100
bridge-domain 100 dot1q
```

Routing Bridged Encapsulation (RBE) Interface:

```
interface atm3/0/0.2 point
ip address 20.0.0.2 255.255.0.0
atm route-bridge ip
pvc 10/100
```

Bridged Routing Encapsulation (BRE) Interface:

```
interface atm3/0/0.1 point-to-point
pvc 0/40
bre-connect 40
```

Half-bridging Interface:

```
interface atm3/0/0.2 point
ip address 20.0.0.2 255.255.0.0
pvc 10/100
encapsulation aal5snap bridge
```

Frame Relay (RFC 1490) Interface:

```
interface POS1/0/0
encapsulation frame-relay ietf
frame-relay interface-dlci 100
bridge-domain 100
```

BCP/PPP (RFC 3518) Interface (Single-VLAN Mode):

```
interface POS1/0/0
encapsulation ppp
bridge-domain 200 dot1q
```

BCP/PPP (RFC 3518) Interface (Trunk Mode):

```
interface POS5/0/0
switchport
switchport trunk allowed vlan all
switchport mode trunk
switchport nonegotiate
no ip address
encapsulation ppp
clock source internal
```

BCP/PPP (RFC 3518) Interface (Tunnel Mode):

```
interface POS5/0/0
no ip address
encapsulation ppp
bridge-domain 100 dot1q-tunnel
clock source internal
```

QoS over Bridging Configuration Examples

The following sections provide several examples of how to configure QoS on interfaces that support bridging. The following types of configuration examples are included:

- Traffic Classification—Examples of how to map traffic into different traffic classes based on characteristics of the traffic. Different QoS features can then be applied to different traffic classes.
- Shaping Traffic—Examples of how to control the flow of traffic on an interface.
- Marking Traffic—Examples of how to set the precedence of packets based on their conformance to QoS policies. Also included is an example of a policy for traffic policing.
- Hierarchical QoS—An example of how to create a policy map that contains other policy maps.

Traffic Classification

The following several examples show how to filter (classify) traffic based on the VLAN ID or CoS bits in the Ethernet header, the Frame Relay DLCI (FR-DLCI) or discard eligibility (FR-DE) bit, and IP precedence or IP DSCP value.

The following class maps classify traffic based on the VLAN ID in the Ethernet (inner) header:

```
Router(config)# class-map match-all vlan100
Router(config-cmap)# match vlan inner 100
Router(config-cmap)# exit
Router(config)# class-map match-all vlan101
Router(config-cmap)# match vlan inner 101
Router(config-cmap)# exit
```

The following class map creates a traffic class that matches all packets whose CoS bits are equal to 7:

```
Router(config)# class-map match-all cos7
Router(config-cmap)# match cos inner 7
Router(config-cmap)# exit
```

The following class maps match Frame Relay traffic with a DLCI of 100 or 101:

```
Router(config)# class-map match-all dlci100
Router(config-cmap)# match fr-dlci 100
Router(config-cmap)# exit
Router(config)# class-map match-all dlci101
Router(config-cmap)# match fr-dlci 101
Router(config-cmap)# exit
```

The following class maps classify traffic based on the IP precedence or IP DSCP value:

```
Router(config)# class-map match-any prec2
Router(config-cmap)# match ip prec 2
Router(config-cmap)# exit

Router(config)# class-map match-any dscp10
Router(config-cmap)# match ip dscp 10
Router(config-cmap)# exit
```

The following class maps classify traffic based on whether the packets are marked for discard during congestion. The first example matches Frame Relay packets whose discard eligibility (FR-DE) bit is set, and the second matches ATM packets whose cell loss priority (ATM-CLP) bit is set:

```
Router(config)# class-map match-all FRdiscard
Router(config-cmap)# match fr-de
Router(config-cmap)# exit

Router(config)# class-map match-all ATMdiscard
Router(config-cmap)# match atm clp
```

```
Router(config-cmap)# exit
```

Shaping Traffic

The next two examples show how to classify and shape traffic based on VLAN ID and Frame Relay DLCI.

This example creates class maps to filter VLAN 100 and VLAN 101 traffic into separate traffic classes and a policy map that shapes the traffic on the VLANs. The policy map is then attached to an ATM PVC.

```
Router(config)# class-map match-all vlan100
Router(config-cmap)# match vlan inner 100
Router(config-cmap)# exit
Router(config)# class-map match-all vlan101
Router(config-cmap)# match vlan inner 101
Router(config-cmap)# exit

Router(config)# policy-map match-vlan
Router(config-pmap)# class vlan100
Router(config-pmap-c)# shape average 1000000 8000 8000
Router(config-pmap)# class vlan101
Router(config-pmap-c)# shape average 1000000 8000 8000
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# interface ATM3/0/0
Router(config-if)# pvc 10/100
Router(config-if)# bridge-domain 100 dot1q
Router(config-if)# service-policy output match-vlan
Router(config-if)# service-policy input match-vlan
Router(config-if)# exit
```

The following commands create class maps and a policy map to classify and shape Frame Relay traffic on POS interface 1/0/0. This example is similar to the one above.

```
Router(config)# class-map match-all DLCI00
Router(config-cmap)# match fr-dlci 100
Router(config-cmap)# exit

Router(config)# policy-map DLCI
Router(config-pmap)# class DLCI100
Router(config-pmap-c)# shape average 1000000 8000 8000
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# interface POS1/0/0
Router(config-if)# encapsulation frame-relay ietf
Router(config-if)# frame-relay interface-dlci 100
Router(config-if)# bridge-domain 100
Router(config-if)# service policy output DLCI
Router(config-if)# exit
```

Marking Traffic

The following example matches Frame Relay traffic with a DLCI of 100 and sets the discard eligibility bit (FR-DE) of those packets:

```
Router(config)# class-map match-all DLCI100
Router(config-cmap)# match fr-dlci 100
Router(config-cmap)# exit
Router(config)# policy-map FR-DE
Router(config-pmap)# class-map DLCI100
Router(config-pmap-c)# set fr-de
Router(config-pmap-c)# exit
Router(config-pmap)# exit
```

The following examples set the ATM CLP bit of packets that have a CoS or IP precedence value of 2. The class maps match traffic based on the value of the CoS or IP precedence bits in the Ethernet header within the packet.

```
Router(config)# class-map match-all COS2
Router(config-cmap)# match cos inner 2
Router(config-cmap)# exit
Router(config)# policy-map ATM-CLP-COS
Router(config-pmap)# class-map COS2
Router(config-pmap-c)# set atm-clp
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# class-map match-all IP-PREC2
Router(config-cmap)# match ip prec 2
Router(config)# policy-map ATM-CLP-IP
Router(config-pmap)# class-map IP-PREC2
Router(config-pmap-c)# set atm-clp
Router(config-pmap-c)# exit
Router(config-pmap)# exit
```

The following commands filter traffic from VLANs 100 and 101 and set the CoS bits of those packets. VLAN 100 traffic is assigned a CoS value of 3 and VLAN 101 traffic is assigned a CoS value of 7 (highest priority):

```
Router(config)# class-map match-all vlan100
Router(config-cmap)# match vlan inner 100
Router(config-cmap)# exit
Router(config)# class-map match-all vlan101
Router(config-cmap)# match vlan inner 101
Router(config-cmap)# exit

Router(config)# policy-map set-cos
Router(config-pmap)# class vlan100
Router(config-pmap-c)# set cos-inner 3
Router(config-pmap)# class vlan101
Router(config-pmap-c)# set cos-inner 7
Router(config-pmap-c)# exit
Router(config-pmap)# exit
```

The following commands perform policing on traffic in VLAN 101. The **police** command (two-rate) sets the traffic rate for VLAN 101 traffic as follows: the committed information rate is 1 Mbps and the peak information rate is 2 Mbps. In addition, the command sets the CoS bits to 6 for traffic that conforms to the specified rate. In this example, the policy is attached to a PVC on ATM interface 3/0/0, but it could be applied to other types of interfaces. **** 2 Mbps ok for police CIR? (I guessed) ****

```
Router(config)# class-map match-all vlan101
Router(config-cmap)# match vlan inner 101
Router(config-cmap)# exit
Router(config)# policy-map police101
Router(config-pmap)# class vlan101
Router(config-pmap-c)# police cir 1000000 bc 31250 pir 2000000 be 31250
  conform-action set-cos-inner-transmit 6 exceed-action drop
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface atm3/0/0
Router(config-if)# pvc 10/100
Router(config-if)# bridge-domain 101 dot1q
Router(config-if)# service-policy input police101
Router(config-if)# service-policy output police101
```

Hierarchical QoS

The following configuration example shows a hierarchical QoS policy that filters traffic from different VLANs (100, 200, 300) into different traffic classes and then allocates bandwidth to the traffic based on CoS values. This example uses child policies within a parent policy (hierarchical QoS).

1. The following commands create three class maps, which map traffic in VLANs 100, 200, and 300 into separate traffic classes:

```
Router(config)# class-map match-all vlan100
Router(config-cmap)# match vlan inner 100
Router(config-cmap)# exit
Router(config)# class-map match-all vlan200
Router(config-cmap)# match vlan inner 200
Router(config-cmap)# exit
Router(config)# class-map match-all vlan300
Router(config-cmap)# match vlan inner 300
Router(config-cmap)# exit
```

2. The following commands create several class maps to classify traffic based on CoS values:

```
Router(config)# class-map match-all cos0
Router(config-cmap)# match cos inner 0
Router(config-cmap)# exit
Router(config)# class-map match-all cos2
Router(config-cmap)# match cos inner 2
Router(config-cmap)# exit
Router(config)# class-map match-all cos4
Router(config-cmap)# match cos inner 4
Router(config-cmap)# exit
Router(config)# class-map match-all cos7
Router(config-cmap)# match cos inner 7
Router(config-cmap)# exit
```

3. The following commands create two policy maps to allocate bandwidth for VLAN 100 and VLAN 200 traffic. Each policy map uses CBWFQ and LLQ to allocate bandwidth based on CoS values.

```
Router(config)# policy-map vlan100
Router(config-pmap)# class cos2
Router(config-pmap-c)# bandwidth percent 10
Router(config-pmap)# class cos4
Router(config-pmap-c)# bandwidth percent 20
Router(config-pmap)# class cos7
Router(config-pmap-c)# priority percent 30
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# policy-map vlan200
Router(config-pmap)# class cos2
Router(config-pmap-c)# bandwidth percent 10
Router(config-pmap)# class cos4
Router(config-pmap-c)# bandwidth percent 20
Router(config-pmap)# class cos7
Router(config-pmap-c)# priority percent 30
Router(config-pmap-c)# exit
Router(config-pmap)# exit
```

4. The following commands create a hierarchical policy map (a policy within a policy). The parent policy map (egress_mpb) assigns a percentage of the link's bandwidth to traffic on VLANs 100 and 200. Two child policy maps (vlan100 and vlan200) further allocate each VLAN's bandwidth to different traffic classes in the VLAN based on CoS values.

```
Router(config)# policy-map egress_mpb
```

```
Router(config-pmap)# class vlan100
Router(config-pmap-c)# bandwidth percent 30
Router(config-pmap-c)# service-policy vlan100
Router(config-pmap-c)# exit
Router(config-pmap)# class vlan200
Router(config-pmap-c)# bandwidth percent 40
Router(config-pmap-c)# service-policy vlan200
Router(config-pmap-c)# exit
```

5. The following commands create a policy map that assign a CoS value of 5 to VLAN 100 traffic and a CoS value of 3 to VLAN 200 traffic:

```
Router(config)# policy-map ingress_mpb
Router(config-pmap)# class vlan100
Router(config-pmap-c)# set cos-inner 5
Router(config-pmap-c)# exit
Router(config-pmap)# class vlan200
Router(config-pmap-c)# set cos-inner 3
Router(config-pmap-c)# exit
Router(config-pmap)# exit
```

6. The following commands create a policy map that sets the ATM CLP (cell loss priority) bit for traffic with a CoS value of 2:

```
Router(config)# policy-map atm-clp
Router(config-pmap)# class cos2
Router(config-pmap-c)# set atm-clp
Router(config-pmap-c)# exit
```

7. The following commands assign the policy maps egress_mpb and ingress_mpb to a POS interface that has been configured for bridging (BCP trunk mode). Note that the BCP trunk on the interface is configured to carry traffic from VLANs 100, 200, and 300.

```
Router(config)# interface POS3/0/0
Router(config-if)# switchport
Router(config-if)# switchport trunk allowed vlan 100,200,300
Router(config-if)# service-policy output egress_mpb
Router(config-if)# service-policy input ingress_mpb
Router(config-if)# encapsulation ppp
Router(config-if)# shutdown
Router(config-if)# no shutdown
Router(config-if)# exit
```

8. Next, we assign the policy map atm-clp to an ATM interface that VLAN 100 traffic has been mapped to:

```
Router(config)# interface ATM4/1/0
Router(config-if)# pvc 15/100
Router(config-if)# bridge-domain 100 dot1q
Router(config-if)# service-policy output atm-clp
Router(config-if)# exit
Router(config)# exit
```

Understanding MPLS QoS

MPLS QoS allows a network administrator to provide differentiated levels of service across an MPLS network. Network administrators can satisfy a wide range of networking requirements by specifying the class of service applicable to each transmitted frame or packet. Different classes of service can be established for frames and packets by setting EXP bits in the attached MPLS label.

Note All of the MPLS QoS features available for the FlexWAN or Enhanced FlexWAN modules are managed from the modular QoS command-line interface (CLI). The modular QoS CLI (MQC) is a command-line interface that allows you to define traffic classes, create and configure traffic policies (policy maps), and then attach those traffic policies to interfaces. The modular QoS command-line interface is described in the *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2*. See: http://www.cisco.com/en/US/products/sw/iosswrel/ps5014/products_feature_guide_chapter09186a008008813a.html

FlexWAN and Enhanced FlexWAN MPLS QoS Feature Summary

The following paragraphs provide a summary of MPLS QoS features for the FlexWAN and Enhanced FlexWAN modules.

Trust

Trust is a port assignment instructing the port whether to trust (leave) existing priorities as is on incoming frames or to rewrite the priority back to zero. Ports on the FlexWAN or Enhanced FlexWAN modules have a default trust state of IP precedence that cannot be changed. The FlexWAN or Enhanced FlexWAN module defaults to accepting the IP precedence or DSCP value contained in the received packets. If you want to mark these values, you can configure a distributed class-based packet marking policy that is applied to ingress traffic while the traffic is still on the FlexWAN or Enhanced FlexWAN module.

Note The FlexWAN or Enhanced FlexWAN module preserves ToS by default.

Classification

Classification is the process of generating a distinct path for a packet by associating it with a QoS label. For received MPLS packets, the FlexWAN or Enhanced FlexWAN modules match on the EXP in the received topmost label using the **match mpls experimental** command. The QoS label that is generated identifies all future QoS actions to be performed on this packet.

Note The FlexWAN or Enhanced FlexWAN modules support all match criteria except qos-group, match input-interface, discard-class, FR-DE, ATM CLP, COS, source address, and destination address.

Marking and Policing

Policing decides whether a packet is in or out of profile by comparing the rate of the inbound traffic to the configured policer. The policer limits the bandwidth consumed by a flow of traffic. The result is passed to the marker.

Marking evaluates the policer configuration information for the action to take when a packet is out of profile. Marking actions are to pass through a packet without modification, to mark down the QoS label in the packet, or to drop the packet.

The FlexWAN or Enhanced FlexWAN modules perform MPLS marking and policing; interfaces are trusted by default.

Note The FlexWAN and Enhanced FlexWAN modules support all forms of the **set** command except qos_group, ATM CLP bit, set cos, FR-DE, and discard-class.
**** Rockies (or Cascades) introduced support for set ATM CLP, set FR-DE, and set cos. does that mean that these options should be deleted here since they are now supported? or is this somehow related to MPLS (but I don't think so) ****

Preserving IP ToS

The FlexWAN or Enhanced FlexWAN module automatically preserves the IP ToS during all MPLS operations including imposition, swapping, and disposition. You do not need to configure a command to do this.

Using MPLS QoS with FlexWAN and Enhanced FlexWAN Modules

This section describes how to use FlexWAN and Enhanced FlexWAN MPLS QoS for the following:

- IP to MPLS Edge (Ingress Interface) QoS Features, page 42
- MPLS to IP Edge (Egress Interface) QoS Features, page 43
- MPLS Core QoS Features, page 43

Note For Information on Ethernet to MPLS (EoMPLS) QoS with the FlexWAN and Enhanced FlexWAN modules, see the "Ethernet over MPLS and EXP Bits" section on page 51.

IP to MPLS Edge (Ingress Interface) QoS Features

This section describes the MPLS QoS features supported at the ingress to the MPLS network.

Note By default, the IP DSCP is preserved throughout the LSP.

Ingress Port Features

Classification based on the packet length is only supported for IPv4 and IPv6 packets.

EXP Marking Support

With the Cisco Supervisor Engine 2 (Sup2), EXP marking is supported for basic MPLS and MPLS/VPN. For MPLS VPN, there is EXP, IP Precedence, and DSCP marking based on IP information (IP Precedence, DSCP, ACL) for imposed (pushed) label only.

With the Sup720, Sup32, or RSP720, for MPLS and MPLS VPN, EXP marking is supported based on IP information (IP Precedence, DSCP, ACL) for imposed (pushed) label only.

Note The **set mpls experimental** command is not supported as an input policy at the IP to MPLS edge.

Egress Port Features

The FlexWAN and Enhanced FlexWAN module port features classify using the **match mpls experimental** command. The **match mpls experimental** command does not match on the EXP in the topmost label.

The Supervisor Engine 2 supports EXP marking at the output QoS policy based on the EXP value for all labels in the label stacks.

The Sup720, Sup32, and RSP720 support EXP marking at the output QoS policy based on EXP value for all labels in the label stacks.

MPLS to IP Edge (Egress Interface) QoS Features

This section describes the MPLS QoS features are supported at the egress interface to the MPLS network.

Ingress Port Features

The FlexWAN and Enhanced FlexWAN module port features classify using the **match mpls experimenta**l command. The **match mpls experimenta**l command matches on the EXP in the received topmost label.

EXP matching is supported.

Egress Port Features

The FlexWAN and Enhanced FlexWAN module port features classify on information in the transmitted IP header.

EXP matching is supported.

MPLS Core QoS Features

This section describes the MPLS QoS features are supported at the MPLS to MPLS core.

Ingress Port Features

The FlexWAN and Enhanced FlexWAN module port features classify using the **match mpls experimenta**l command. The **match mpls experimenta**l command matches on the EXP in the received topmost label.

EXP marking is supported at the input QoS policy based on EXP bits.

Egress Port Features

The FlexWAN and Enhanced FlexWAN module port features classify using the **match mpls experimental** command.

EXP marking is supported.

Configuring MPLS QoS

Note See the "Using MPLS QoS with FlexWAN and Enhanced FlexWAN Modules" section on page 42 for information about ingress and egress port features at the edges and core of an MPLS VPN network.

QoS is configured through the Modular QoS command-line interface (class maps and policy maps). QoS functionality is distributed and performed on the FlexWAN and Enhanced FlexWAN modules.

These sections provide configuration information for MPLS QoS:

FlexWAN MPLS QoS

Cisco 7600 Supervisor Engines and the Route Switch Processor 720 provide MPLS QoS capabilities; however, the FlexWAN and Enhanced FlexWAN modules do not use them. All MPLS QoS features are distributed by the FlexWAN and Enhanced FlexWAN modules. In this respect, FlexWAN QoS behavior is similar to that of the Cisco 7500 VIP QoS. The FlexWAN and Enhanced FlexWAN modules perform all input and output QoS while the Supervisor Engine or RSP720 performs MPLS switching.

Note The Cisco 7500 VIP makes extensive use of qos-group and discard-class for MPLS DiffServ tunnelling modes. However, these functions are not supported by the Cisco 7600 FlexWAN or Enhanced FlexWAN.

The FlexWAN and Enhanced FlexWAN modules provide the following:

- Ability to map IP ToS / DSCP / EXP to the output COS bits
- Output queuing for MPLS packets based on the output CoS bits. The Policy Feature Card copies the topmost outgoing EXP bits to the output CoS value. The FlexWAN and Enhanced Flexwan modules use the **match mpls experimental** command to match on the output CoS bits.
- Configuration commands follow the MQC guidelines

FlexWAN MPLS QoS with Supervisor Engine 2

With the Supervisor Engine 2, the FlexWAN and Enhanced FlexWAN modules perform the label lookup and label operations. With the Supervisor Engine 2, the FlexWAN and Enhanced FlexWAN modules provide the following:

- Ability to map IP ToS / DSCP to the MPLS EXP bits
- Output queuing based on the EXP bits in the label
- Configuration commands follow the MQC guidelines

Supported FlexWAN MPLS QoS Features

The FlexWAN and Enhanced FlexWAN modules support the following MPLS QoS features:

- QoS features using MPLS EXP classification
- MPLS EXP policing and marking

The following additional QoS features are supported in MPLS QoS in the same manner as in IP QoS:

- Weighted Fair Queuing (WFQ)
- Class Based Weighted Fair Queuing (CBWFQ)
- Low Latency Queuing (LLQ)
- Weighted Random Early Detection (WRED)
- Traffic Shaping
- Policing

Understanding the MPLS Experimental Field

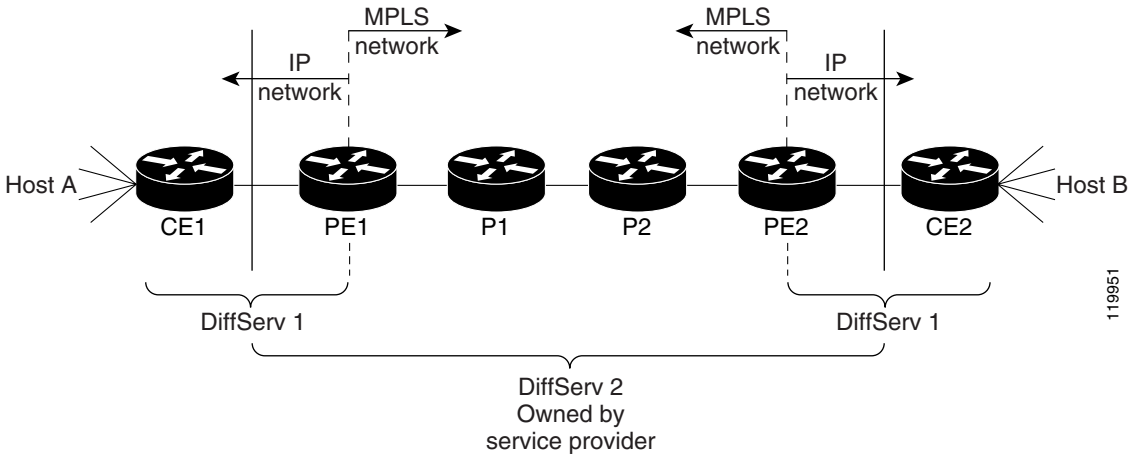
Setting the MPLS experimental field value satisfies the requirement of service providers that do not want the value of the IP precedence field modified within IP packets transported through their networks.

By choosing different values for the MPLS experimental field, you can mark packets so that packets have the priority that they require during periods of congestion.

By default, the IP precedence value is copied into the MPLS experimental field during imposition.You can mark the MPLS EXP bits with a policy.

Figure 5 shows a service provider MPLS network connecting two sites of a customer network.

Figure 5 MPLS Network Connecting Two Sites of a Customer's IP Network



Setting the MPLS Experimental Field

Use the **set mpls experimental** command on the input interface to set the pushed label entry's value during label imposition.

By default, the label edge router copies the IP Precedence of the IP packet to the MPLS EXP field in all pushed label entries.

You can optionally map the IP Precedence or DSCP field to the MPLS EXP field in the MPLS header by using the **set mpls experimental** command.

Configuring Class-Based Marking for MPLS

To configure Class-based Marking for MPLS, perform the tasks described in the following sections:

- Configuring a Class Map to Classify MPLS Packets, page 46
- Configuring a Policy Map to Set the MPLS Experimental Field, page 47
- Attaching the Service Policy, page 47
- Verifying QoS Operation, page 48

Note Class-based marking for MPLS (with Supervisor Engine 2) is supported only on the P-facing interface of the ingress PE.

Configuring a Class Map to Classify MPLS Packets

To configure a class map, perform this task beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# class-map class-name`	Specifies the class map to which packets will be matched.
Step 2	`Router(config-cmap)# match mpls experimental value`	Specifies the packet characteristics that will be matched to the class.
Step 3	`Router(config-cmap)# exit`	Exits class-map configuration mode.

This example shows that all packets that contain MPLS experimental value 3 match any with the class name exp-class:

```
Router(config)# class-map match-any exp-class
Router(config-cmap)# match mpls experimental 3
Router(config-cmap)# exit
```

Configuring a Policy Map to Set the MPLS Experimental Field

To configure a policy map, perform this task beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# policy-map policy-name`	Creates a policy map that can be attached to one or more interfaces to specify a service policy.
Step 2	`Router(config-pmap)# class class-name`	Specifies the name of the class map previously designated in the **class-map** command.
Step 3	`Router(config-pmap-c)# set mpls experimental value`[1]	Designates the value to which the MPLS bits are set if the packets match the specified policy map.
Step 4	`Router(config-pmap-c)# exit`	Exits policy-map configuration mode.

1. You can also configure additional supported features, such as shaping.

This example shows that the value in the MPLS experimental field of each packet that is matched by the policy-map MARKING is set to 5:

```
Router(config)# policy-map MARKING
Router(config-pmap)# class exp-class
Router(config-pmap-c)# set mpls experimental 5
Router(config-pmap-c)# exit
Router(config-pmap)# exit
```

Attaching the Service Policy

To attach the service policy to an interface, perform the following steps from global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface name`	Designates the output interface.
Step 2	`Router(config-if)# service-policy {input | output} policy-name`	Attaches the specified policy map to the interface.
Step 3	`Router(config-if)# exit`	Exits interface configuration mode.

This example shows that the service policy set_experimental_5 is applied to outgoing traffic on the POS interface POS6/0/0:

```
Router(config)# policy-map MARKING
Router(config-pmap)# class exp-class
Router(config-pmap-c)# set mpls experimental 5
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# interface POS6/0/0
Router(config-if)# service-policy output set_experimental_5
Router(config-if)# exit
```

Verifying QoS Operation

To verify the operation of MPLS QoS, perform this task:

Command	Purpose
Router# **show policy-map** ***interface*** *[interface-name]*	Displays detailed information about QoS.

Configuration Examples

Sample configurations provided in this section can be applied to FlexWAN and Enhanced FlexWAN modules supported on the Cisco 7600 series routers.

Ingress PE Router Configuration

In the following example, IP packets with IP precedence 1 entering an MPLS network are shaped to 2,000,000 bits per second and set to MPLS experimental field 3.

Step 1 Define the traffic class:

```
Router(config)# class-map exp1
Router(config-cmap)# match mpls experimental 1
Router(config-cmap)# exit
```

Note Traffic classes should be defined to match on MPLS experimental values instead of IP precedence.

Step 2 Define a policy to take actions on traffic classes:

```
Router(config)# policy-map gold1
Router(config-pmap)# class exp 1
Router(config-pmap-c)# shape average 2000000
Router(config-pmap-c)# set mpls experimental 3
Router(config-pmap-c)# exit
```

Step 3 Apply the policy to the output interface of a PE router:

```
Router(config)# interface POS6/1/0
Router(config-if)# ip address 153.61.0.2 255.255.0.0
Router(config-if)# tag-switching ip
Router(config-if)# service-policy output gold1
Router(config-if)# exit
```

Core Router Ingress and Egress Interface Configuration

The following example provides ingress and egress interface configurations at the P router:

Step 1 Configure the ingress interface:

```
Router(config)# interface pos 5/0/0
Router(config-if)# ip address 153.61.0.1 255.255.0.0
Router(config-if)# tag-switching ip
Router(config-if)# service-policy input gold1
```

Step 2 Configure the egress interface:

```
Router(config)# interface pos 5/0/3
Router(config-if)# ip address 153.62.0.1 255.255.0.0
Router(config-if)# tag-switching ip
Router(config-if)# service-policy input gold1
```

Configuring MPLS VPN QoS

Note See the "Using MPLS QoS with FlexWAN and Enhanced FlexWAN Modules" section on page 42 for information about ingress and egress port features at the edges and core of an MPLS VPN network.

The FlexWAN and Enhanced FlexWAN modules support the following MPLS VPN QoS features:

- FlexWAN QoS features using MPLS EXP classification.
- The Supervisor Engine 2 supports MPLS EXP marking done by the FlexWAN and Enhanced FlexWAN modules. See the "Configuring Class-Based Marking for MPLS" section on page 46.
- All other Supervisor Engines and the RSP720 support MPLS EXP policing and marking done by the FlexWAN and Enhanced FlexWAN modules.

In addition to these features, with a Supervisor Engine 2, MPLS VPN also supports the **set ip precedence** command on the input WAN interfaces on the FlexWAN and Enhanced FlexWAN modules.

The following restrictions apply to the support for MPLS VPN QoS on the FlexWAN and Enhanced FlexWAN modules:

- PFC2 QoS features are not supported with MPLS VPN.
- MPLS VPN QoS is supported on the VPN interfaces only.
- Match IP precedence and Set IP precedence and MPLS Experimental values are supported on the input interface only.

Configuration Example

The following example shows how to configure QoS on an MPLS VPN:

```
Router# configure terminal
Router(config)# class-map match-any vpn-class
Router(config-cmap)# match ip precedence 3
Router(config-cmap)# exit
Router(config)# policy-map VPN-MARKING
```

```
Router(config-pmap)# class vpn-class
Router(config-pmap-c)# set ip precedence 5
Router(config-pmap-c)# set mpls exp 5
Router(config-pmap-c)# ^Z
Router# configure terminal
Router(config)# interface POS 6/0/0
Router(config-if)# service-policy output VPN-MARKING
Router(config-if)# ^Z
Router# show running-config interface p6/0/0
Building configuration...

Current configuration: 175 bytes
!
interface POS 6/0/0
 ip vrf forwarding TEST
 ip address 194.3.1.3 255.255.255.0
 negotiation auto
 service-policy input VPN-MARKING
 mls qos trust dscp
end
Router#
```

Configuring QoS with Any Transport over MPLS

The following QoS features are supported on Any Transport over MPLS:

- Marking on the CE-facing card—(imposition packets) with match criteria, match-dlci, match-any, or class-default.

Note For Marking on the CE-facing card, match-dlci applies to the FlexWAN module only.

- Shaping on the core-facing card, with match exp, and match-any.
- Shaping on the CE-facing card (disposition packets) with match-any.
- WRED on the core-facing card with match criteria, match-exp, or match-any

This section explains how to configure QoS with AToM and includes the following procedures:

- How to Set Experimental Bits with AToM, page 50
- Enabling Traffic Shaping, page 54

Note PFC QoS features do not apply to ATM over MPLS and Frame Relay over MPLS packets.

How to Set Experimental Bits with AToM

For configuration steps and examples, see the "How to Set Experimental Bits with AToM" section on page 50.

MPLS AToM uses the three experimental bits in a label to determine the queue of packets. You statically set the experimental bits in both the VC label and the LSP tunnel label, because the LSP tunnel label might be removed at the penultimate router. The following sections explain the transport-specific implementations of the EXP bits.

Ethernet over MPLS and EXP Bits

Note The information in this section is for FlexWAN-based EoMPLS only. For more information on PFC QoS, see the following URL.
http://www.cisco.com/univercd/cc/td/doc/product/core/cis7600/software/122sx/swcg/qos.htm

FlexWAN-based EoMPLS supports the following QoS implementations:

- VLAN interface policies
- Core-facing interface policy

You apply a VLAN interface policy to an individual VLAN. You may configure a unique policy for each individual VLAN. Within a policy, you can classify on 802.1q P bits to set the MPLS experimental bits. You can also implement a single traffic shaper that applies to all traffic within the VLAN.

You apply a core-facing interface policy to the EoMPLS uplink interface. This policy applies to traffic from all VLANs. It does not distinguish between different VLANs. Within a policy, you can classify on MPLS experimental bits and configure the following features:

- Class-based traffic shaping
- Class-based weighted fair queuing (CBWFQ)
- Low latency queuing (LLQ)
- Weighted random early detection (WRED)

Note You cannot use both VLAN interface policies and core-facing interface policies at the same time. If you configure QoS for FlexWAN-based EoMPLS, you must select either VLAN interface policies or a core-facing interface policy.

For more information on the commands used to enable Quality of Service, see the following documents:

- *Modular Quality of Service Command-Line Interface*
- *Cisco IOS Quality of Service Solutions Command Reference, Release 12.2*

Setting the Priority of Packets with the Experimental Bits

Ethernet over MPLS provides Quality of Service (QoS) using the three experimental bits in a label to determine the priority of packets. To support QoS between LERs, set the experimental bits in both the VC and tunnel labels. If you do not assign values to the experimental bits, the priority bits in the 802.1q header's “tag control information” field and are written into the experimental bit fields.

Perform the following steps to set the experimental bits:

	Command	Purpose
Step 1	`Router(config)# class-map class-name`	Specifies the user-defined name of the traffic class.
Step 2	`Router(config-cmap)# match cos 0-7`	Specifies that IEEE 802.1Q packets with the cos-values of 0-7 be matched. As an alternative, you can use the **match any** command.
Step 3	`Router(config-cmap)# policy-map policy-name`	Specifies the name of the traffic policy to configure.

	Command	Purpose
Step 4	`Router(config-pmap)# class class-name`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
Step 5	`Router (config-pmap-c)# set mpls experimental value`	Designates the value to which the MPLS bits are set if the packets match the specified policy map.
Step 6	`Router(config)# interface vlan vlan-number`	Enters the VLAN interface.
Step 7	`Router(config-if)# service-policy [input | output] policy-name`	Attaches a traffic policy to an interface.

Note You can enable traffic shaping and set experimental bits in the same policy-map.

Note You can configure the service-policy for either the input or the output direction. However, the policy is always implemented on the core-facing FlexWAN module port and is applied only to the traffic leaving the core-facing FlexWAN module port.

Enabling Traffic Shaping

Traffic shaping limits the rate of transmission of data. Average rate shaping limits the transmission rate to the committed information rate (CIR). To add traffic shaping, issue the following commands:

	Command	Purpose
Step 1	`Router(config)# class-map class-name`	Specifies the user-defined name of the traffic class.
Step 2	`Router(config-cmap)# match any`	Specifies that all packets will be matched. (Using the class-default in the policy-map would have the same effect.)
Step 3	`Router(config-cmap)# policy-map policy-name`	Specifies the name of the traffic policy to configure.
Step 4	`Router(config-pmap)# class class-name`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
Step 5	`Router (config-pmap-c)# shape average cir` [1]	Shapes traffic according to the bit rate you specify.
Step 6	`Router(config)# interface vlan vlan-number`	Enters the VLAN interface.
Step 7	`Router(config-if)# service-policy [input | output] policy-name`	Assigns a traffic policy to an interface.

1. Only supported parameters are shown.

The shape average rate is rounded to the nearest multiple of the link rate divided by 255. If the shape value is lower than the link rate divided by 255, it is rounded up to link rate divided by 255.

This example shows how the shape value is rounded:

```
Router# show pol p2
 Policy Map p2
  class  any-pkt
   shape average 2000000 8000 8000

Router# show pol int

 Vlan101

  service-policy input:p2

    class-map:any-pkt (match-all)
      2018169 packets, 4575195376 bytes
      30 second offered rate 295768000 bps, drop rate 0 bps
      match:any
      queue size 0, queue limit 0
      packets input 40492, packet drops 1977677
      tail/random drops 0, no buffer drops 0, other drops 1977677
      shape:cir 2000000,  Bc 8000,  Be 8000
      input bytes 40847436, shape rate 1874000 bps

    class-map:class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      match:any
        0 packets, 0 bytes
        30 second rate 0 bps
```

Displaying the Traffic Policy Assigned to an Interface

To display the traffic policy attached to an interface, issue the following command:

```
Router# show policy-map vlan50
service-policy input: badger

    class-map: blue (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      match: any
      queue size 0, queue limit 2
      packets input 0, packet drops 0
      tail/random drops 0, no buffer drops 0, other drops 0
      shape: cir 2000000,  Bc 8000,  Be 8000
        output bytes 0, shape rate 0 bps

    class-map: class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      match: any
        0 packets, 0 bytes
     30 second rate 0 bps
```

ATM AAL5 over MPLS and EXP Bits

- ATM AAL5 over MPLS allows you to statically set the experimental bits.
- If you do not assign values to the experimental bits, the priority bits in the header's “tag control information” field are set to zero.

ATM Cell Relay over MPLS and EXP Bits

- ATM Cell Relay over MPLS allows you to statically set the experimental bits in VC mode.
- If you do not assign values to the experimental bits, the priority bits in the header's *tag control information* field are set to zero.

Frame Relay over MPLS and EXP Bits

Frame Relay over MPLS provides QoS using the three experimental bits in a label to determine the priority of PDUs. If you do not assign values to the experimental bits, the priority bits in the header's *tag control information* field are set to zero. For Frame Relay over MPLS, you must set the experimental bits on a per-DLCI basis.

Setting the Priority of Packets with EXP Bits

Set the experimental bits in both the VC label and the LSP tunnel label. You set the experimental bits in the VC label, because the LSP tunnel label might be removed at the penultimate router.

Perform the following steps to set the experimental bits.

	Command or Action	Purpose
Step 1	`Router# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	`Router(config)# configure terminal`	Enters global configuration mode.
Step 3	`Router(config)# class-map class-name`	Specifies the user-defined name of the traffic class.
Step 4	`Router(config-cmap)# match any`	Specifies that all packets will be matched. Use only the **any** keyword. Other keywords might cause unexpected results.
Step 5	`Router(config-cmap)# policy-map policy-name`	Specifies the name of the traffic policy to configure.
Step 6	`Router(config-pmap)# class class-name`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
Step 7	`Router(config-pmap-c)# set mpls experimental value`	Designates the value to which the MPLS bits are set if the packets match the specified policy map.
Step 8	`Router(config)# interface slot/port`	Enters the interface.
Step 9	`Router(config-if)# service-policy input policy-name`	Attaches a traffic policy to an interface.

Enabling Traffic Shaping

Traffic shaping limits the rate of transmission of data. Average rate shaping limits the transmission rate to the committed information rate (CIR). To add traffic shaping, issue the following commands:

	Command or Action	Purpose
Step 1	`Router# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	`Router(config)# configure terminal`	Enters global configuration mode.
Step 3	`Router(config)# class-map class-name`	Specifies the user-defined name of the traffic class.
Step 4	`Router(config-cmap)# match any`	Specifies that all packets will be matched. Use only the **any** keyword. Other keywords might cause unexpected results.
Step 5	`Router(config-cmap)# policy-map policy-name`	Specifies the name of the traffic policy to configure.
Step 6	`Router(config-pmap)# class class-name`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
Step 7	`Router(config-pmap-c)# shape average bit value`	Shapes traffic according to the bit rate you specify.
Step 8	`Router(config)# interface slot/port`	Enters the interface.
Step 9	`Router(config-if)# service-policy input policy-name`	Attaches a traffic policy to an interface.

Note You can enable traffic shaping and set experimental bits in the same policy-map.

Note EoMPLS VLAN Policing Exclusion—traffic on the EoMPLS uplink port is excluded from a VLAN-based ingress policer.

Displaying the Traffic Policy Assigned to an Interface

To display the traffic policy attached to an interface, use the **show policy-map interface** command.

EoMPLS QoS Example

If the egress MPLS tunnel is carried on a FlexWAN interface configured for fair queuing, the shape value is rounded to the nearest multiple of the link rate divided by 255. If the shape value is lower than the link rate divided by 255, it is rounded up to link rate divided by 255.

This example shows how the shape value is rounded:

```
Router# show pol p2
 Policy Map p2
  class  any-pkt
   shape average 2000000 8000 8000

Router# show pol int

 Vlan101

  service-policy input:p2
```

```
    class-map:any-pkt (match-all)
      2018169 packets, 4575195376 bytes
      30 second offered rate 295768000 bps, drop rate 0 bps
      match:any
      queue size 0, queue limit 0
      packets input 40492, packet drops 1977677
      tail/random drops 0, no buffer drops 0, other drops 1977677
      shape:cir 2000000,  Bc 8000,  Be 8000
      (shape parameter is rounded to 2439000 due to granularity)
      input bytes 40847436, shape rate 1874000 bps

    class-map:class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      match:any
        0 packets, 0 bytes
        30 second rate 0 bps
```

EoMPLS QoS Example—Displaying the Traffic Policy Assigned to an Interface

To display the traffic policy attached to an interface, issue the following command:

```
Router# show policy-map vlan50
service-policy input: badger

    class-map: blue (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      match: any
      queue size 0, queue limit 2
      packets input 0, packet drops 0
      tail/random drops 0, no buffer drops 0, other drops 0
      shape: cir 2000000,  Bc 8000,  Be 8000
        output bytes 0, shape rate 0 bps

    class-map: class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      match: any
        0 packets, 0 bytes
        30 second rate 0 bps
```

Ethernet over MPLS QoS Example—Configuring QoS on VLAN

The following example show how to configure QoS on the VLAN.

```
class-map blue
match cos 1 2 3
!
policy-map badger
class blue
set mpls experimental 1
class class-default
shape average 2000000 8000 8000
!
interface vlan50
no ip address
no ip mroute-cache
load-interval 30
mpls l2transport route 192.168.255.255 50
service-policy input badger
no cdp enable
```

ATM over MPLS QoS Example—Configuring Ingress QoS

This example shows ingress QoS. On this side the policy attaches to a multipoint l2transport PVC. In this configuration the EXP bits are set to 5 for all packets on PVC 1/101.

```
class-map match-any anyclass
     match any
!
policy-map set-policy
  class anyclass
    set mpls experimental 5

interface ATM6/0/0
 no ip address
 logging event link-status
 atm clock INTERNAL
 pvc 1/101 l2transport
   encapsulation aal5
   mpls l2transport route 10.10.10.10 101
   service-policy input set-policy
```

For input shaping, the config is same as above but the action in the policy-map should be changed to shape.

```
policy-map shape-policy
  class anyclass
    shape average 16000 3200 3200

For output shaping based on MPLS EXP, the policy is configured on the main interface.
class-map match-any exp2
  match mpls experimental 2
!
policy-map shape-policy
  class exp2
    shape average 16000 32 32

interface POS4/0/1
ip address 20.1.1.1 255.255.255.0
 service-policy output shape-policy
 no ip mroute-cache
 load-interval 30
 no keepalive
 mpls label protocol ldp
 tag-switching ip
 mls qos trust dscp
 clock source internal
 no cdp enable
```

Frame Relay over MPLS QoS Example—Configuring Ingress QoS

On the ingress side, you attach the policy to a switched frame-relay DLCI. The configuration below matches frame relay packets with DLCI 10 and sets the EXP bits to 5 during label imposition for the matched packets.

```
class-map match-any anyclass
  match any
!
!
policy-map set-policy
  class anyclass
    set mpls experimental 5
```

```
map-class frame-relay dlci101
 service-policy input set-policy

interface POS1/0/1
 no ip address
 encapsulation frame-relay IETF
 no ip mroute-cache
 load-interval 30
 no keepalive
 mls qos trust dscp
 clock source internal
 frame-relay interface-dlci 101 switched
  class dlci101
```

For input shaping, the configuration is same as above but the action in the policy-map is changed to shape as follows:

```
policy-map shape-policy
  class anyclass
    shape average 1600000 6400 6400
```

For output shaping based on MPLS EXP, the policy is configured on the main interface.

```
class-map match-any exp2
  match mpls experimental 2
!
policy-map shape-policy
  class exp2
    shape average 1600000 6400 6400

interface POS4/0/1
ip address 20.1.1.1 255.255.255.0
 service-policy output shape-policy
 no ip mroute-cache
 load-interval 30
 no keepalive
 mpls label protocol ldp
 tag-switching ip
 mls qos trust dscp
 clock source internal
 no cdp enable
```

For WRED based on MPLS EXP, configure the policy on the main interface.

```
class-map match-any exp2
  match mpls experimental 2
!
policy-map wred-pol
  class exp2
    bandwidth percent 20
    random-detect

interface POS4/0/1
  service-policy output wred-pol
```

DE/CLP and EXP Mapping on Frame Relay and ATM over MPLS VC

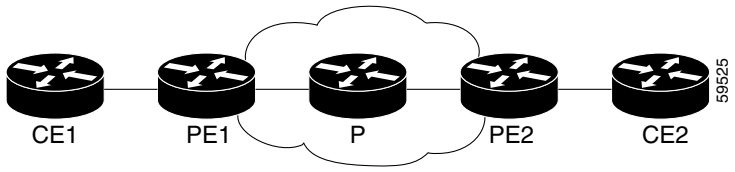
The DE/CLP and EXP Mapping on Frame Relay/ATM over MPLS VC feature allows you to map the Frame Relay Discard Eligibility (DE) bit or the ATM Congestion Loss Priority (CLP) bit to the MPLS EXP value at the ingress interface to an MPLS AToM network and to map the MPLS EXP value to the FR-DE or ATM CLP bit at the egress interface of an MPLS AToM network.

The DE bit indicates that a frame has lower importance than other frames. Similarly, the ATM CLP bit indicates whether the cell may be discarded if it encounters extreme congestion as it moves through the network.

In Figure 6, the PE1 tags the incoming packet with the MPLS EXP value and sends the packet to the next hop. At each hop, matching is done on the EXP value. At the PE2 egress interface, however, the packet is no longer MPLS but IP, so matching cannot occur on the EXP value.

Internally, the FlexWAN or Enhanced FlexWAN module preserves the EXP value in the QoS_group so matching on the QoS_group at the PE2 egress interface provides the same effect as matching on the EXP value.

Figure 6 DE/CLP and EXP Mapping



Match on ATM CLP Bit

Use Match on ATM CLP Bit at the ingress to an MPLS AToM network to map the ATM cell loss priority (CLP) of the packet arriving at an interface to the EXP value, and then apply the desired QoS functionality and actions (for example, traffic policing) to those packets.

Note This feature is supported on policy maps attached to ATM permanent virtual circuits (PVCs) only.

Configuring Match on ATM CLP Bit for Ingress Policy

Perform the following steps to configure Match on ATM CLP Bit for the ingress policy:

	Command or Action	Purpose
Step 1	`Router# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	`Router(config)# configure terminal`	Enters global configuration mode.
Step 3	`Router(config)# class-map class-name`	Specifies the user-defined name of the traffic class.
Step 4	`Router(config-cmap)# match atm clp`	Enables packet matching on the basis of the ATM CLP bit set to 1.

	Command or Action	Purpose
Step 5	`Router(config-cmap)# policy-map policy-name`	Specifies the name of the traffic policy to configure.
Step 6	`Router(config-pmap)# class class-name`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
Step 7	`Router(config-pmap-c)# set mpls experimental value`	Designates the value to which the MPLS bits are set if the packets match the specified policy map.
Step 8	`Router(config)# interface atm interface-number`	Enters interface configuration mode.
Step 9	`Router(config-if)# pvc [name] vpi/vci [l2transport]`	Enter ATM virtual circuit configuration mode.
Step 10	`Router(config-if)# service-policy input policy-name`	Attaches a traffic policy to an interface.

Match on ATM CLP Bit Configuration Example

The following provides an example a Match on ATM CLP Bit configuration:

```
CFLOW_PE1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
CFLOW_PE1(config)# class-map CLP
CFLOW_PE1(config-cmap)# match atm clp
CFLOW_PE1(config-cmap)# exit
CFLOW_PE1(config)# policy-map CLP2EXP
CFLOW_PE1(config-pmap)# class CLP
CFLOW_PE1(config-pmap-c)# set mpls experimental 1
CFLOW_PE1(config-pmap-c)# exit
CFLOW_PE1(config-pmap)# interface ATM3/0/0
CFLOW_PE1(config-if)# pvc 1/100
CFLOW_PE1(cfg-if-atm-l2trans-pvc)# service-policy input CLP2EXP
CFLOW_PE1(cfg-if-atm-l2trans-pvc)# end
CFLOW_PE1#
```

Verifying Match on ATM CLP Bit

Use the **show policy-map interface** command to verify the Match on ATM CLP Bit as in the following example:

```
CFLOW_PE1# show policy-map interface a3/0/0
ATM3/0/0: VC 1/100 -

Service-policy input: CLP2EXP

Class-map: CLP (match-all)
200 packets, 22400 bytes
5 minute offered rate 2000 bps, drop rate 0 bps
Match: atm clp
QoS Set
mpls experimental imposition 1
Packets marked 200

Class-map: class-default (match-any)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: any
CFLOW_PE1#
```

Match on FR-DE Bit

Use Match on FR-DE Bit at the ingress to an MPLS AToM network to map the Frame Relay discard eligible (DE) bit of the packet arriving at an interface to the EXP value.

Guideline for Match on FR-DE Bit

The following guideline applies to this feature:

- Use policy matching on the FR-DE as an input policy only.

Configuring Match on FR-DE Bit for Ingress Policy

Perform the following steps to configure Match on FR-DE Bit for the ingress policy:

	Command or Action	Purpose
Step 1	`Router# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	`Router(config)# configure terminal`	Enters global configuration mode.
Step 3	`Router(config)# class-map` *class-name*	Specifies the user-defined name of the traffic class.
Step 4	`Router(config-cmap)# match fr-de`	Matches on packets that have the Frame Relay DE bit set to 1.
Step 5	`Router(config-cmap)# policy-map` *policy-name*	Specifies the name of the traffic policy to configure.
Step 6	`Router(config-pmap)# class` *class-name*	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
Step 7	`Router(config-pmap-c)# set mpls experimental` *value*	Designates the value to which the MPLS bits are set if the packets match the specified policy map.
Step 8	`Router(config)# interface` *slot/port*	Enters the interface.
Step 9	`Router(config)# map-class frame-relay` *class-map-name*	Creates a Frame Relay map class where *class-map-name* is the name of the class map.
	Note In Step 10, you can apply the map-class policy to the main interface so that all DLCIs have the same policy or you can apply the map-class policy to each DLCI.	
Step 10	`Router(config-if)# service-policy input` *policy-name*	Attaches a traffic policy to an interface.

Example of Configuring a Match on the FR-DE Bit

The following example shows how to configure a match on the FR-DE bit for the ingress policy by applying the map-class policy to the main interface.

Applying Map-Class Policy to the Main Interface

```
osr3# show class-map match_fr-de
 Class Map match-all match_fr-de (id 2)
```

```
   Match fr-de

osr3# show policy fr-de_mpls4
  Policy Map fr-de_mpls4
    Class match_fr-de
      set mpls experimental imposition 4
    Class class-default
      set mpls experimental imposition 4

osr3# show run map-class | begin fr-de_mpls4
map-class frame-relay fr-de_mpls4
 service-policy input fr-de_mpls4
!
map-class frame-relay fr-de_mpls0
 service-policy input fr-de_mpls0
!

osr3# show run interface pos1/0/2
Building configuration...

Current configuration : 196 bytes
!
interface POS1/0/2
 mtu 5000
 no ip address
 encapsulation frame-relay IETF
 no keepalive
 clock source internal
 pos scramble-atm
 frame-relay intf-type dce
end

connect frompls_1 POS1/0/2 16 l2transport
 xconnect 11.11.11.11 2001 encapsulation mpls
 !
!
connect frompls_2 POS1/0/2 17 l2transport
 xconnect 11.11.11.11 2002 encapsulation mpls
 !
!
connect frompls_3 POS1/0/2 18 l2transport
 xconnect 11.11.11.11 2003 encapsulation mpls
 !
!
osr3# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
osr3(config)# interface POS1/0/2
osr3(config-if)# frame-relay class fr-de_mpls4
osr3(config-if)#
osr3(config-if)#^Z

osr3# show run interface pos1/0/2
Building configuration...

Current configuration : 196 bytes
!
interface POS1/0/2
 mtu 5000
 no ip address
 encapsulation frame-relay IETF
 no keepalive
 clock source internal
 pos scramble-atm
 frame-relay class fr-de_mpls4
```

```
 frame-relay intf-type dce
end
```

Examples of Matching on the FR-DE Bit Verification

Verify the configuration with the **show policy-map interface** command.

```
osr3# show policy interface pos1/0/2
 POS1/0/2: DLCI 16 -

  Service-policy input: fr-de_mpls4

    Class-map: match_fr-de (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: fr-de
      QoS Set
        mpls experimental imposition 4
           Packets marked 0

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        mpls experimental imposition 4
           Packets marked 0
 POS1/0/2: DLCI 1007 -

  Service-policy input: fr-de_mpls4

 --More--
```

Configuring a Match on the FR-DE Bit for the Ingress Policy

The following example shows how to configure a match on the FR-DE bit for the ingress policy by applying the map-class policy to the different DLCIs:

```
osr1# show policy fr-de_mpls2
  Policy Map fr-de_mpls2
    Class match_fr-de
      set mpls experimental imposition 2
    Class class-default
      set mpls experimental imposition 2
osr1# sh policy fr-de_mpls3
  Policy Map fr-de_mpls3
    Class match_fr-de
      set mpls experimental imposition 3
    Class class-default
      set mpls experimental imposition 3
osr1# show class-map match_fr-de
 Class Map match-all match_fr-de (id 1)
   Match fr-de

osr1# show run map-class | begin fr-de_mpls
map-class frame-relay fr-de_mpls2
 service-policy input fr-de_mpls2
!
map-class frame-relay fr-de_mpls3
 service-policy input fr-de_mpls3
!
osr1# configure terminal
```

```
Enter configuration commands, one per line.  End with CNTL/Z.
osr1(config)#int pos1/7
osr1(config-if)#frame-relay interface-dlci 16 switched
osr1(config-fr-dlci)#class fr-de_mpls2
osr1(config-fr-dlci)#exit
osr1(config-if)#
osr1(config-if)#frame-relay interface-dlci 17 switched
osr1(config-fr-dlci)#class fr-de_mpls3
osr1(config-fr-dlci)#
osr1(config-fr-dlci)#exit
osr1(config-if)#
osr1(config-if)#^Z
osr1#

osr1# show run interface pos1/7
Building configuration...

Current configuration : 39671 bytes
!
interface POS1/7
 mtu 5000
 no ip address
 encapsulation frame-relay IETF
 no keepalive
 mls qos trust dscp
 clock source internal
 pos scramble-atm
 frame-relay interface-dlci 16 switched
  class fr-de_mpls2
 frame-relay interface-dlci 17 switched
  class fr-de_mpls3
 frame-relay interface-dlci 18 switched
 frame-relay interface-dlci 19 switched
...
```

Verify the Configuration

Verify the configuration with the **show policy-map interface** command.

```
osr1# show policy interface pos1/7
 POS1/7: DLCI 16 -

  Service-policy input: fr-de_mpls2

    Class-map: match_fr-de (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: fr-de
      QoS Set
        mpls experimental imposition 2
           Packets marked 0

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        mpls experimental imposition 2
           Packets marked 0
 POS1/7: DLCI 17 -

  Service-policy input: fr-de_mpls3
```

```
    Class-map: match_fr-de (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: fr-de
      QoS Set
        mpls experimental imposition 3
           Packets marked 0

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        mpls experimental imposition 3
           Packets marked 0
osr1#
```

Set on ATM CLP Bit

Use Set on ATM CLP Bit at the egress interface of an MPLS AToM network to map the EXP value to the ATM CLP bit.

Note This feature is supported on policy maps attached to ATM permanent virtual circuits (PVCs) only.

Configuring Set on ATM CLP Bit for Egress Policy

Perform the following steps to configure Set on ATM CLP Bit for the egress policy:

	Command or Action	Purpose
Step 1	`Router# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	`Router(config)# configure terminal`	Enters global configuration mode.
Step 3	`Router(config)# class-map` *class-name*	Specifies the user-defined name of the traffic class.
Step 4	`Router(config-cmap)# match qos-group` *qos-group-value*	Identifies a specific quality of service (QoS) group value as a match criterion. The QoS group value has no mathematical significance. **Note** The QoS group concept is directly derived from the incoming MPLS EXP value and is valid only with AToM. You cannot use MQC to set QoS group value.
Step 5	`Router(config-cmap)# policy-map` *policy-name*	Specifies the name of the traffic policy to configure.
Step 6	`Router(config-pmap)# class` *class-name*	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
Step 7	`Router(config-pmap-c)# set atm-clp`	Sets the cell loss priority (CLP) bit when a policy map is configured.

	Command or Action	Purpose
Step 8	**Router(config)# interface** *slot/port*	Enters the interface.
Step 9	Router(config-if)# **service-policy input** *policy-name*	Attaches a traffic policy to an interface.

Set on ATM CLP Bit Configuration Example

The following provides a Set on ATM CLP Bit configuration example:

```
7600router# show policy qg2clp
 Policy Map qg2clp
   Class qg1
     set atm-clp
7600router# show class-map qg1
Class Map match-all qg1 (id 23)
  Match qos-group 1

7600router# show run interface a9/1
interface ATM9/1
no ip address
atm clock INTERNAL
atm mtu-reject-call
mls qos trust dscp
pvc 1/100 l2transport
  encapsulation aal5
  mpls l2transport route 101.101.101.101 1000
  service-policy out qg2clp
```

Match on ATM CLP Bit Verification Example

Verify the configuration with the **show policy-map interface** command.

```
7600router# show policy interface ATM9/1
ATM9/1: VC 1/100 -

 Service-policy output: qg2clp

   Class-map: qg1 (match-all)
     1000 packets, 0 bytes
     5 minute offered rate 0 bps, drop rate 0 bps
     Match: qos-group 1
     QoS Set
       atm-clp
         Packets marked 1000
```

Set on FR-DE Bit

Use Set on FR-DE Bit at the egress interface of an MPLS AToM network to map the EXP value to the FR-DE bit.

Configuring Set on FR-DE Bit for the Egress Policy

Perform the following steps to configure Set on FR-DE Bit for the egress policy:

	Command or Action	Purpose
Step 1	`Router# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	`Router(config)# configure terminal`	Enters global configuration mode.
Step 3	`Router(config)# class-map` *class-name*	Specifies the user-defined name of the traffic class.
Step 4	`Router(config-cmap)# match qos-group` *qos-group-value*	Identifies a specific quality of service (QoS) group value as a match criterion where the range of the *qos-group-value* is 0-7.
Step 5	`Router(config-cmap)# policy-map` *policy-name*	Specifies the name of the traffic policy to configure.
Step 6	`Router(config-pmap)# class` *class-name*	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
Step 7	`Router(config-pmap-c)# set fr-de`	Changes the DE bit setting in the address field of a Frame Relay frame to 1 for all traffic leaving an interface.
Step 8	`Router(config)# interface` *slot/port*	Enters the interface.
	Note In Step 9 below you can apply the map-class policy to the main interface so that all DLCIs have the same policy or you can apply the map-class policy to each DLCI.	
Step 9	`Router(config-if)# service-policy output` *policy-name*	Attaches a traffic policy to an interface.

Set on FR-DE Configuration Example

The following example shows a Set on FR-DE configuration example:

```
7600router# show policy qg2de
 Policy Map qg2de
   Class qg1
     set fr-de
7600router# show class-map qg1
Class Map match-all qg1 (id 23)
  Match qos-group 1

7600router# show run interface pos2/2/0
interface POS2/2/0
no ip address
encapsulation frame-relay
frame-relay interface-dlci 16 switched
   class QG2DE
```

Set on FR-DE Verification Example

Verify the configuration with the **show policy-map interface** command.

```
7600router# show policy interface POS2/2/0
POS2/2/0: DLCI 16 -
```

```
Service-policy output: qg2de

  Class-map: qg1 (match-all)
    1000 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: qos-group 1
    QoS Set
      fr-de
        Packets marked 1000
```

HQoS for Ethernet over MPLS Virtual Circuits

The Hierarchical Quality of Service (HQoS) for EoMPLS VCs feature enables hierarchical QoS services on WAN-based interfaces, allowing service providers to classify the traffic in customer Ethernet over MPLS networks before it is forwarded into the core network. This gives users of Cisco Catalyst 6500 series switches and Cisco 7600 series routers greater flexibility in providing QoS services to specific customers in their EoMPLS networks.

The HQoS for EoMPLS VCs feature allows you to classify EoMPLS networks in the following ways:

- Match on the VLAN ID that the packet contained when it was originally received at the input interface. You can match a single VLAN ID, a range of VLAN IDs, or a combination of the two, allowing you to match all or part of an EoMPLS network.
- Match on a QoS group value that is set to the same value of the IP precedence or CoS bits that are received with the packet at the input interface.

The use of hierarchical policy maps can simplify the configuration of the router, because the same child policy map can be used in multiple parent maps. You can also match multiple VLANs with one class map, as opposed to having separate class maps for each VLAN.

The HQoS for EoMPLS VCs feature does not require any upgrades to the customer-facing interfaces, because the HQoS policy map is applied to the WAN interface, allowing the customer-facing interfaces to be standard Ethernet interfaces.

Prerequisites for the HQoS for EoMPLS VCs Feature

- You must enable QoS on the router before using HQoS. To enable QoS globally on the router, use the **mls qos** command in global configuration mode. To enable QoS on an individual interface, use the **mls qos** interface configuration command. In addition, **mls trust** must be configured on the CE facing PE interfaces.

Restrictions for the HQoS for EoMPLS VCs Feature

The following section lists restrictions for the HQoS for EoMPLS VC feature. Other restrictions may also apply to QoS services in general, depending on the supervisor module and line cards being used.

- If a policy contains a class map with a **match input vlan** command, you cannot attach that policy map to an interface if you have already attached a service policy to a VLAN interface (a logical interface that has been created with the **interface vlan** command).

Note This restriction means that **match input vlan** configurations and **interface vlan** configurations are mutually exclusive.

- The HQoS for EoMPLS VCs feature is supported only for output (egress) interfaces (policy maps must be attached to the interface using the **service-policy output** command).
- The HQoS for EoMPLS VCs feature supports only point-to-point VCs, not point-to-multipoint VCs.
- If the parent class contains a class map with a **match input vlan** command, you cannot use a **match exp** command in a child policy map.
- Child and parent policy maps do not support any marking, such as the **match ip dscp** and **set** commands.
- The HQoS for EoMPLS VCs feature does not support multiple levels of parent and child policy map nesting. Each parent policy map supports only one level of nesting. In other words, a traffic class in a parent policy map can have a maximum of one child policy map, and child policy maps cannot have their own child policy maps.

Note You can mix flat traffic classes (that do not refer to child policy maps) and hierarchical traffic classes (that do refer to child policy maps) in the same HQoS parent policy maps.

- You cannot apply both HQoS output policy on a main interface (using the **service-policy output** command) and an output policy (**service-policy output** command) on a subinterface of that same interface. If you attempt to do so, then attaching the HQoS output policy fails with the following error message:

```
Attaching service policy to main and sub-interface concurrently is not allowed
```

- In Cisco IOS Release 12.2(18)SXE and later releases, policy maps can contain a maximum of 255 class maps.
- Child policy maps support only strict priority (the **priority** command without any options). Parent policy maps do not support any form of the **priority** command.
- When using both the **priority** and **police** commands in more than one class in a child priority map, you must configure the commands in the following order:
  - In the first class to be configured on the priority map, specify the **priority** command first, and then the **police** command.
  - In the second and any additional classes to be configured on the priority map, specify the **police** command first, and then the **priority** command.
  - The **police cir** command is not supported on the FlexWAN or Enhanced FlexWAN modules.

Note The **priority** command can be configured only with the **police** command. You cannot use priority together with any forms of the **bandwidth** or **shape** commands.

- Class maps that use the **match input vlan** command support only the **match-any** option. You cannot use the **match-all** option in class maps that use the **match input vlan** command.
- Classes using the **match input vlan** command should always be placed first in the policy maps, before any classes that use flat policies.
- Parent policy maps do not support the **fair-queue** command.
- You must use class-default for the input service policy on a CE-PE interface that uses the **qos-group** command to set CoS or IP Precedence.
  **** can now set COS/IP prec apart from qos-group, right? ****

Note For additional prerequisites and restrictions for HQoS in general, see the "Configuring Hierarchical Traffic Shaping" section on page 22.

Supported Features

The HQoS for Ethernet over MPLS VCs feature supports the following commands on the class maps and policy maps for output interfaces.

The following are supported on parent policy maps:

- **bandwidth**—Egress class-based weighted fair queuing (CBWFQ).
- **shape average**—Egress shaping

The following are supported on child policy maps:

- **bandwidth**—Egress class-based weighted fair queuing (CBWFQ)
- **priority**—Egress low latency queuing (LLQ) (Only strict priority is supported on child maps.)
- **queue-limit**—Queue throttling
- **random-detect**—Egress weighted random early detection (WRED)
- **shape average**—Egress shaping

Related Commands

Do not confuse the **match input vlan** command with the **match vlan** command, which is also a class-map configuration command.

- The **match vlan** command matches the VLAN ID on packets for the particular interface at which the policy map is applied. Policy maps using the **match vlan** command can be applied to either ingress or egress interfaces on the router, using the **service-policy** {**input** | **output**} command.
- The **match input vlan** command matches the VLAN ID that was on packets when they were received on the ingress interface on the router. Policy maps using the **match input vlan** command must be applied to egress interfaces on the router, using the **service-policy output** command.

The **match input vlan** command can also be confused with the **match input-interface vlan** command, which matches packets being received on a logical VLAN interface that is used for inter-VLAN routing.

Tip Because class maps also support the **match input-interface** command, you cannot abbreviate the **input** keyword when giving the **match input vlan** command.

Configuring the HQoS for EoMPLS VCs Feature

To use a hierarchical QoS policy map for EoMPLS traffic, you must perform the following tasks. (All tasks are required.)

- Apply a policy map to the input interface to set the QoS group value on incoming packets. See the "Creating and Assigning a Policy Map to Mark the QoS Group at the Incoming Interface" section on page 71.

- Create class maps that match packets on the basis of their QoS group values. See the "Configuring the Class Map to Match on a QoS Group" section on page 74.
- Create a child policy map that uses these class maps. See the "Creating the Child Policy Map for the Egress Interface" section on page 76.
- Create class maps that match packets on the basis of their input VLAN IDs. See the "Configuring the Class Maps for Matching on an Input VLAN" section on page 80.
- Create a parent policy map and apply it to the output interface. See the "Creating the Parent Policy Map and Attaching It to the Egress Interface" section on page 83.

Note For more information about hierarchical traffic shaping, see the "Configuring Hierarchical Traffic Shaping" section on page 22.

Creating and Assigning a Policy Map to Mark the QoS Group at the Incoming Interface

To be able to classify traffic on a QoS group, you must first create a policy map that marks incoming packets with the desired QoS group value. You can set the QoS group value to the value of either the IP precedence bits or 802.1P CoS bits of the incoming packets. You then must assign that policy map to the incoming interface (which must be a Layer 2 LAN interface).

To perform these tasks, use the following procedure.

Command Sequence Summary

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **description** *string*
5. **class class-default**
6. **set qos-group** {**cos** | **ip-precedence**}
7. **interface** *if-type* {*slot/port* | *slot/subslot/port*}
8. **service-policy input** *policy-map-name*
9. **end**
10. **show policy-map** *policy-map-name* [**class** *class-map*]

Detailed Steps

	Command or Action	Purpose

Step 1	**enable**
	
	Example:
	`Router> enable`
	`Router#`
		Enables privileged EXEC mode. Enter your password if prompted.

Step 2	**configure terminal**
	
	Example:
	`Router# configure terminal`
	`Router(config)#`
		Enters global configuration mode.

Step 3	**policy-map** *policy-map-name*
	
	Example:
	`Router(config)# policy-map cos-to-qosgrp-pmap`
	`Router(config-pmap)#`
		Creates a policy map with the specified name and enters policy-map configuration mode.
		• *policy-map-name*—Name of the policy map. The name must be a unique string of up to 40 alphanumeric characters.

Step 4	**description** *string*
	
	Example:
	`Router(config-pmap)# description Sets QoS group to IP precedence bits of incoming packets`
	`Router(config-pmap)#`
		(Optional) Defines an arbitrary string, up to 200 characters long, that describes this policy map.

Step 5	**class class-default**
	
	Example:
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)#`
		Specifies the default class to be used for traffic with this policy, and enters policy-map class configuration mode.

Step 6	**set qos-group** {**cos** | **ip-precedence**}
	
	Example:
	`Router(config-pmap-c)# set qos-group cos`
	`Router(config-pmap-c)#`
		Sets a quality of service (QoS) group identifier (ID) that can be used later to classify packets.
		• **cos**—Sets the packet's QoS group value to the same value as the packet's original 802.1P Class of Service (CoS) bits.
		• **ip-precedence**—Sets the packet's QoS group value to the same value as the packet's original IP precedence bits.
		Note The **set qos-group** command also supports setting the QoS group to an arbitrary value from 0 to 99, but this configuration is not supported when using the HQoS for EoMPLS VCs feature. This command also supports the option of specifying a table map, but the HQoS for EoMPLS VCs feature does not support this option, because it always uses the default mappings.

	Command or Action	Purpose

Step 7	**interface** *if-type* {*slot/port* | *slot/subslot/port*}
	
	Example:
	`Router(config-pmap-c)# interface GigabitEthernet 5/2`
	`Router(config-if)#`
		Enters interface configuration mode for the incoming interface.
		
		Note This interface must be a Layer 2 LAN interface. It cannot be a Layer 3 WAN interface.

Step 8	**service-policy input** *policy-map-name*
	
	Example:
	`Router(config-if)# service-policy input cos-to-qosgrp-pmap`
	`Router(config-if)#`
		Attaches the specified policy map to the interface for input (ingress) traffic.
		
		• *policy-map-name*—Name of the policy map that was created in Step 3.

	Note Repeat Step 7 and Step 8 for each interface that should be marking the QoS group value on incoming traffic.	

Step 9	**show policy-map** *policy-map-name* [**class** *class-map*]
	
	Example:
	`Router# show policy-map prec-to-qosgrp-pmap`
	`(command output)`
	`Router#`
		(Optional) Displays the configured class map to verify the configuration. To display all policy maps, enter the command without any options. To display a specific policy map, specify its name on the command line. You can also display a specific class that is part of a specific policy map by adding the **class** option.

The following policy map sets the QoS group value to match the CoS value of the incoming packets. The policy map is then assigned to two interfaces:

```
policy-map cos-to-qosgroup-pmap
   class class-default
      set qos-group cos
...
!
interface GE 6/0
 service-policy input cos-to-qosgroup-pmap
...
!
interface GE 6/1
 service-policy input cos-to-qosgroup-pmap
...
```

What to Do Next

After attaching the policy map to the input interface, create the class map to match on the QoS group value at the egress (outgoing) interface. See the next section, "Configuring the Class Map to Match on a QoS Group," for details.

Configuring the Class Map to Match on a QoS Group

To be able to match EoMPLS traffic using QoS groups, you must create class maps to match traffic on the basis of the QoS group value at the egress (outgoing) interface. To create these class maps, use the following procedure.

Prerequisites

- You must create policy maps that contain class maps that use the **set qos-group** command to mark incoming packets with the desired QoS group values. Then attach those policy maps to the input interfaces that are receiving the incoming traffic. See the "Creating and Assigning a Policy Map to Mark the QoS Group at the Incoming Interface" section on page 71.
- Input interfaces must also be configured with **mls trust** command.

Restrictions

- A policy map that refers to a class map that uses the **match qos-group** command cannot have other class maps that match on the following criteria:
  - **match ip prec match**
  - **match mpls exp**
- The allowable range of values for QoS groups is from 0 to 99. The only valid values for EoMPLS traffic are from 0 to 7. This is because the QoS group value is set to the IP precedence or CoS fields in the incoming packets, and both of these fields are only 3-bit values that can range from 0 to 7.

Command Sequence Summary

These are the commands you

1. **enable**
2. **configure terminal**
3. **class-map** [**match-all** | **match-any**] *class-map-name*
4. **match qos-group** *qos-group-value*
5. **end**
6. **show class-map** *class-map-name*

Detailed Steps

	Command or Action	Purpose

Step 1	**enable**
	
	Example:
	`Router> enable`
	`Router#`
		Enables privileged EXEC mode. Enter your password if prompted.

Step 2	**configure terminal**
	
	Example:
	`Router# configure terminal`
	`Router(config)#`
		Enters global configuration mode.

Step 3	**class-map** [**match-all** | **match-any**] *class-map-name*
	
	Example:
	`Router(config)# class-map group4`
	`Router(config-cmap)#`
		Creates a class map and enters class-map configuration mode.
		• **match-all**—(Optional) All match criteria must be matched for a packet to be matched by this class map. This is the default if no option is specified.
		• **match-any**—(Optional) Only one match criterion must be matched for a packet to be matched by this class map.
		• *class-map-name*—Arbitrary string that identifies this class map.

Step 4	**match qos-group** *qos-group-value*
	
	Example:
	`Router(config-cmap)# match qos-group 4`
	`Router(config-cmap)#`
		Matches packets with the specified QoS group marking.
		• *qos-group-value*—Specifies the QoS group value to be matched. The allowable range is from 0 to 99, but for EoMPLS traffic, the only valid values are from 0 to 7, because the QoS group value is set to the value of the IP precedence or CoS bits in the incoming packets.

Step 5	**end**
	
	Example:
	`Router(config-cmap)# end`
	`Router#`
		Exits class-map configuration mode and returns to privileged EXEC mode.

Step 6	**show class-map** *class-map-name*
	
	Example:
	`Router# show class-map group4`
	`(command output)`
	`Router#`
		(Optional) Displays the configured class map to verify the configuration.

The following example configuration shows all of the class maps that are allowed for matching on QoS groups for EoMPLS traffic.

```
class-map match-all group0
  match qos-group 0
class-map match-all group1
  match qos-group 1
class-map match-all group2
```

```
  match qos-group 2
class-map match-all group3
  match qos-group 3
class-map match-all group4
  match qos-group 4
class-map match-all group5
  match qos-group 5
class-map match-all group6
  match qos-group 6
class-map match-all group7
  match qos-group 7
```

What to Do Next

After creating all of the desired class maps, you must include them in a child policy map. See the next section, "Creating the Child Policy Map for the Egress Interface," for details.

Creating the Child Policy Map for the Egress Interface

A hierarchical policy map is identical to a flat policy map, except that at least one of the traffic class maps in the parent policy map refers to a child policy map. You must create the child policy maps before creating the parent policy maps.

To create a child policy map, use the following procedure. Repeat as needed to create the desired number of child policy maps.

Tip Different parent policy maps can use the same child policy maps, if desired.

Prerequisites

You must first create the class maps to be used by this policy map. See the "Configuring the Class Map to Match on a QoS Group" section on page 74.

Restrictions

Child policy maps for EoMPLS traffic have the following restrictions:

- The **set** command is not supported on the child policy map.
- Child policy maps support only strict priority (the **priority** command without any options). Parent policy maps do not support any form of the **priority** command.
- When using both the **priority** and **police** commands in more than one class in a priority map, you must configure the commands in the following order:
  - In the first class to be configured on the priority map, specify the **priority** command first, and then the **police** command.
  - In the second and any additional classes to be configured on the priority map, specify the **police** command first, and then the **priority** command.
- You cannot use the **service-policy** *child-pmap-name* command in child policy maps, because multi-level nesting is not supported for HQoS for EoMPLS VCs policy maps.

Command Sequence Summary

1. **enable**

2. **configure terminal**
3. **policy-map** *child-pmap-name*
4. **description** *string*
5. **class** {*class-map-name* | **class-default**}

Note Each class action below must be preceded by a **class** command.

6. **shape** {**average**} *mean-rate*]
7. **class** {*class-map-name* | **class-default**}
8. **priority**
9. **police** *bps* [*burst-normal*] [*burst-max*] **conform-action** *action* **exceed-action** *action* [**violate-action** *action*]
10. **class** {*class-map-name* | **class-default**}
11. **bandwidth** {*bandwidth-kbps* | **remaining percent** *percentage* | **percent** *percentage*}
12. **end**
13. **show policy-map** *child-pmap-name*

Detailed Steps

	Command or Action	Purpose

Step 1	**enable**
	
	Example:
	`Router> enable`
	`Router#`
		Enables privileged EXEC mode. Enter your password if prompted.

Step 2	**configure terminal**
	
	Example:
	`Router# configure terminal`
	`Router(config)#`
		Enters global configuration mode.

Step 3	**policy-map** *child-pmap-name*
	
	Example:
	`Router(config)# policy-map child-pmap-name`
	`Router(config-pmap)#`
		Creates a policy map with the specified name, for use as a child policy map, and enters policy-map configuration mode.
		• *child-pmap-name*—Name of the child policy map. The name must be a unique string of up to 40 alphanumeric characters.

Step 4	**description** *string*
	
	Example:
	`Router(config-pmap)# description Child policy map for input VLAN parent class`
	`Router(config-pmap)#`
		(Optional) Defines an arbitrary string, up to 200 characters long, that describes this policy map.

	Command or Action	Purpose

Step 5	**class** {*class-map-name* | **class-default**}
	
	Example:
	`Router(config-pmap)# class qosgroup4`
	`Router(config-pmap-c)#`
	`or`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)#`
		Specifies the name of a class map that should be used with this policy, and enters policy-map class configuration mode.
		• *class-map-name*—Name of the class map to be used. This should be a class map that was created using the **class-map** command in previous configuration tasks.
		• **class-default**—Specifies the default class that should be used for this policy for unclassified traffic that does not match the other class maps for this policy.

Note Each class action below must be preceded by a **class** command.

Step 6	**shape** {**average**} *mean-rate*]
	
	Example:
	`Router(config-pmap-c)# shape average 10000000`
	`Router(config-pmap-c)#`
		(Optional) Shapes the traffic in this class by the limits specified.
		• **average**—Limits traffic to the maximum bit rate that is specified by the *mean-rate* parameter.
		• *mean-rate*—Maximum number of bits to transmitted, in bits per second. Also called the Committed Information Rate (CIR). The valid range is from 8000 to 4,000,000,000 bits per second, with no default.

Step 7	**priority**
	
	Example:
	`Router(config-pmap-c)# priority`
	`Router(config-pmap-c)#`
		(Optional) Specifies that traffic in this class is priority traffic.
		Note You cannot configure both the **shape** and the **priority** commands in the same class.

Note When using both the **priority** and **police** commands in a class, you must configure them in the following order: In the first class to be configured on the priority map, specify the **priority** command first, and then the **police** command. In the second and any additional classes to be configured on the priority map, specify the **police** command first, and then the **priority** command.

	Command or Action	Purpose

Step 8	**police** *bps* [*burst-normal*] [*burst-max*] **conform-action** *action* **exceed-action** *action* [**violate-action** *action*]
	
	Example:
	`Router(config-pmap-c)# police 8000 1000 conform-action transmit exceed-action drop`
	`Router(config-pmap-c)#`
		(Optional) Specifies the policing policy that should be used for traffic in this class.
		• *bps*—Average rate in bits per second. The valid range is from 8,000 to 200,000,000.
		• *burst-normal*—(Optional) The normal maximum burst size in bytes. The valid range is from 1,000 to 51,200,000 bytes, with a default value of 1,500 bytes.
		• *burst-max*—(Optional) Excess burst size in bytes. The valid range is from 1,000 to 51,200,000.
		• **conform-action**—Specifies the action to take for packets that are within the specified rate limit.
		• **exceed-action**—Specifies the action to take for packets that exceed the specified rate limit.
		• **violate-action**—(Optional) Specifies the action to take for packets that violate the normal and maximum burst sizes.
		• *action*—Action to be taken for the specified condition. The most common values are **drop** (drop the packet) or **transmit** (transmits the packet without change). Additional values are possible for setting different class of service (CoS) parameters.

Step 9	**bandwidth** {*bandwidth-kbps* | **remaining percent** *percentage* | **percent** *percentage*}
	
	Example:
	`Router(config-pmap-c)# bandwidth percent 50`
	`Router(config-pmap-c)#`
		(Optional) Specifies the bandwidth that is allowed for traffic in this class.
		• *bandwidth-kbps*—Amount of bandwidth, in kbps, to be assigned to the class. The valid range is from 1 to 2000000, but the allowable values vary according to the interface and platform in use.
		• **remaining percent**—Amount of guaranteed bandwidth, based on a relative percent of available bandwidth. The valid range for *percentage* is from 1 to 100.
		• **percent**—Amount of guaranteed bandwidth, based on an absolute percent of available bandwidth. The valid range for *percentage* is from 1 to 100.
		Note Cisco IOS Release 12.2(18)SXE and earlier releases did not support the **bandwidth** command in parent policy maps. This restriction was removed in Cisco IOS Release 12.2(18)SXE and later releases for OC-3 and OC-12 POS OSM and OSM-2+4GE-WAN-GBIC+ interfaces only.

Note	Repeat Step 5 through Step 9 for each class to be used in this child policy map.	

	Command or Action	Purpose

Step 10	**end**
	
	Example:
	`Router(config-pmap-c)# end`
	`Router#`
		Exits policy-map class configuration mode and returns to privileged EXEC mode.

Step 11	**show policy-map** *child-pmap-name* [**class** *class-map*]
	
	Example:
	`Router# show policy-map child-policy1`
	`(command output)`
	`Router#`
		(Optional) Displays the configured class map to verify the configuration. To display all policy maps, enter the command without any options. To display a specific policy map, specify its name on the command line. You can also display a specific class that is part of a specific policy map by adding the **class** option.

The following sample configuration shows a typical child policy map that refers to two of the QoS group class maps that were defined in the "Configuring the Class Map to Match on a QoS Group" section on page 74.

```
policy-map child
! Class for QoS Group 3 performs LLQ
 class group3
  priority
  police 20000000 625000 625000 conform-action transmit exceed-action drop
! Class for QoS Group 4 performs CBWFQ when bandwidth usage is at 30 percent
 class group4
  bandwidth percent 30
```

When Using the Priority and Police Commands in a Class

When using both the **priority** and **police** commands in a class, you must configure them in the following order:

- In the first class to be configured on the priority map, specify the **priority** command first, and then the **police** command.
- In the second and any additional classes to be configured on the priority map, specify the **police** command first, and then the **priority** command.

What to Do Next

After creating the child policy map, you must create the parent policy map. See "Creating the Parent Policy Map and Attaching It to the Egress Interface" section on page 83 for details.

Configuring the Class Maps for Matching on an Input VLAN

To match EoMPLS packets that are tagged with one or more specific VLAN IDs, you must create a class map that matches on those VLAN IDs. To do this, use the following procedure.

Command Sequence Summary

1. **enable**
2. **configure terminal**
3. **class-map match-any** *class-map-name*
4. **match input vlan** *input-vlan-list*

5. **end**
6. **show class-map** *class-map-name*

Detailed Procedure

	Command or Action	Purpose

Step 1	**enable**
	
	Example:
	`Router> enable`
	`Router#`
		Enables privileged EXEC mode. Enter your password if prompted.

Step 2	**configure terminal**
	
	Example:
	`Router# configure terminal`
	`Router(config)#`
		Enters global configuration mode.

Step 3	**class-map match-any** *class-map-name*
	
	Example:
	`Router(config)# class-map vlan-map`
	`Router(config-cmap)#`
		Creates a class map and enters class-map configuration mode.
		• *class-map-name*—Arbitrary string that identifies this class map.
		Note Class maps that use the **match input vlan** command support only the **match-any** option. You cannot use the **match-all** option in these class maps.

Step 4	**match input vlan** *input-vlan-list*
	
	Example:
	`Router(config-cmap)# match input vlan 10 20 30 100-1999`
	`Router(config-cmap)#`
		Matches packets that are tagged with a VLAN ID specified in the *input-vlan-list*, which can be one or both of the following:
		• Single VLAN IDs, separated by spaces. The valid range is 0 to 4094.
		• One or more ranges of VLAN IDs, separated by spaces. The allowable values are between 0 and 4094.
		Note Repeat this command, if desired, to specify additional VLANs. If you use multiple **match input vlan** commands, be sure to use the **match-any** keyword in Step 3 so that the class map can match on any of the VLAN IDs.

Step 5	**end**
	
	Example:
	`Router(config-cmap)# end`
	`Router#`
		Exits class-map configuration mode and returns to privileged EXEC mode.

Step 6	**show class-map** *class-map-name*
	
	Example:
	`Router# show class-map vlan-map`
	`(command output)`
	`Router#`
		(Optional) Displays the configured class map to verify the configuration.

Examples

The following configuration example shows a number of class maps that match either one specific VLAN ID, or a range of VLAN IDs. The last class map matches all valid VLAN IDs.

```
class-map match-any vlan1
  match input vlan 1
class-map match-any vlan2
  match input vlan 2
class-map match-any vlan3
  match input vlan 3
class-map match-any vlan4
  match input vlan 4
class-map match-any vlans1-4
  match input vlan 1-4
class-map match-any vlans-all
  match input vlan 1-4094
```

The following sample configuration shows multiple **match input vlan** commands being used in the traffic class map.

```
class-map match-any vlans-even
  match input vlan 2 4 6 8
  match input vlan 102 104 106 108
  match input vlan 202 204 206 208
```

What to Do Next

After creating all desired class maps, you must then create the parent policy map and assign it to the egress interface. See the next section, "Creating the Parent Policy Map and Attaching It to the Egress Interface," for details.

Creating the Parent Policy Map and Attaching It to the Egress Interface

After creating the class maps and child policy maps, you must create a parent policy map and attach it to the appropriate egress (output) interface. To create and attach a parent policy map, use the following procedure. Repeat as needed to create the desired number of parent policy maps.

Prerequisites

Create at least one child policy map to be used in this parent policy map. See the "Creating the Child Policy Map for the Egress Interface" section on page 76 for details. (Different parent policies can use the same child policy maps, if desired.)

Restrictions

Parent policy maps have the following restrictions:

- You cannot attach a policy with the **match input vlan command** to an interface if you have already attached a service policy to its VLAN interface (a logical interface that has been created with the **interface vlan** command). If you attempt to do so, you must then remove both types of policy maps from all interfaces, and then reattach only one type of policy map to the interfaces.
- The **priority** and **fair-queue** commands are not supported in parent policy maps.
- Only the **shape** command and the **bandwidth** command are supported in parent classes; other actions are not supported.
- The **bandwidth** command is supported on parent policy maps.

Note Cisco IOS Release 12.2(18)SXE and earlier releases do not support the **bandwidth** command in parent policy maps with HQoS configurations. This restriction no longer exists in Cisco IOS Release 12.2(18)SXE and later releases.

Command Sequence Summary

The command sequence summary is as follows:

1. **enable**
2. **configure terminal**
3. **policy-map** *parent-pmap-name*
4. **description** *string*
5. **class** {*class-map-name*}
6. **shape** {**average** | **peak**} *mean-rate* [*Bc* [*Be*]]
7. **bandwidth** {*bandwidth-kbps* | **remaining percent** *percentage* | **percent** *percentage*}
8. **service-policy** *child-pmap-name*
9. **interface** *if-type* {*slot/port* | *slot/subslot/port*}
10. **service-policy output** *parent-pmap-name*
11. **end**
12. **show policy-map** *parent-pmap-name*

Detailed Steps

	Command or Action	Purpose

Step 1	**enable**
	
	Example:
	`Router> enable`
	`Router#`
		Enables privileged EXEC mode. Enter your password if prompted.

Step 2	**configure terminal**
	
	Example:
	`Router# configure terminal`
	`Router(config)#`
		Enters global configuration mode.

Step 3	**policy-map** *parent-pmap-name*
	
	Example:
	`Router(config)# policy-map parent-policy1`
	`Router(config-pmap)#`
		Creates a policy map with the specified name, for use as a parent policy map, and enters policy-map configuration mode.
		• *parent-pmap-name*—Name of the parent policy map. The name must be a unique string of up to 40 alphanumeric characters.

Step 4	**description** *string*
	
	Example:
	`Router(config-pmap)# description Parent Policy Map`
	`Router(config-pmap)#`
		(Optional) Defines an arbitrary string, up to 200 characters long, that describes this policy map.

Step 5	**class** {*class-map-name*}
	
	Example:
	`Router(config-pmap)# class vlan100`
	`Router(config-pmap-c)#`
	`or`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)#`
		Specifies the name of a class map that should be used with this policy, and enters policy-map class configuration mode.
		• *class-map-name*—Name of the class map to be used. This should be a class map that was created using the **class-map** command in the "Configuring the Class Maps for Matching on an Input VLAN" section on page 80.

Step 6	**shape** {**average**} *mean-rate*]
	
	Example:
	`Router(config-pmap-c)# shape average 10000000`
	`Router(config-pmap-c)#`
		(Optional) Shapes the traffic in this class by the limits specified.
		• **average**—Limits traffic to the maximum bit rate that is specified by the *mean-rate* parameter.
		• *mean-rate*—Maximum number of bits to transmitted, in bits per second. Also called the Committed Information Rate (CIR). The valid range is from 8,000 to 4,000,000,000 bits per second, with no default.

	Command or Action	Purpose

Step 7	**bandwidth** {*bandwidth-kbps* | **remaining percent** *percentage* | **percent** *percentage*}
	
	Example:
	`Router(config-pmap-c)# bandwidth percent 50`
	`Router(config-pmap-c)#`
		(Optional) Specifies the bandwidth that is allowed for traffic in this class.
		• *bandwidth-kbps*—Amount of bandwidth, in kbps, to be assigned to the class. The valid range is from 1 to 2,000,000, but the allowable values vary according to the interface and platform in use.
		• **remaining percent**—Amount of guaranteed bandwidth, based on a relative percent of available bandwidth. The valid range for *percentage* is from 1 to 100.
		• **percent**—Amount of guaranteed bandwidth, based on an absolute percent of available bandwidth. The valid range for *percentage* is from 1 to 100.
		Note Cisco IOS Release 12.2(18)SXE and earlier releases did not support the **bandwidth** command in parent policy maps. This restriction was removed in Cisco IOS Release 12.2(18)SXE and later releases for OC-3 and OC-12 OSM POS, and OSM-2+4GE-WAN-GBIC+ interfaces only.

Step 8	**service-policy** *child-pmap-name*
	
	Example:
	`Router(config-pmap-c)# service-policy child-pmap-name`
	`Router(config-pmap-c)#`
		Specifies a child policy map that should be applied to the traffic in this class:
		• *child-pmap-name*—Name of a child policy map that was created previously in the "Creating the Child Policy Map for the Egress Interface" section on page 76. (The child policy map cannot be another parent policy map—that is, it cannot be a policy map that also uses the **service-policy** command.)

	Note Repeat Step 5 through Step 8 for each class to be used to match VLANs in this parent policy map.	

Step 9	**interface** *if-type* {*slot/port* | *slot/subslot/port*}
	
	Example:
	`Router(config)# interface ge-wan 5/2`
	`Router(config-if)#`
		Enters interface configuration mode for the specified interface.

Step 10	**service-policy output** *parent-pmap-name*
	
	Example:
	`Router(config-pmap)# service-policy output parent-policy1`
	`Router(config-pmap)#`
		Attaches the specified parent policy map to the interface for outgoing traffic.
		• *parent-pmap-name*—Name of the policy map that was created in the Step 3.

	Command or Action	Purpose

Step 11	**end**
	
	Example:
	`Router(config-pmap-c)# end`
	`Router#`
		Exits policy-map class configuration mode and returns to privileged EXEC mode.

Step 12	**show policy-map** *parent-pmap-name* [**class** *class-map*]
	
	Example:
	`Router# show policy-map vlan-map`
	`(command output)`
	`Router#`
		(Optional) Displays the configured class map to verify the configuration. To display all policy maps, enter the command without any options. To display a specific policy map, specify its name on the command line. You can also display a specific class that is part of a specific policy map by adding the **class** option.

The following sample configuration shows a parent policy map that shapes all of the traffic for three VLANs to specific maximum values. Each class in the parent policy map also specifies a child policy map that further shapes the VLAN traffic on the basis of each packet's QoS group value.

```
!
! Class maps to match on QoS groups (to be used in child policy map)
class-map match-all qosgroup0
  match qos-group 0
class-map match-all qosgroup1
  match qos-group 1
class-map match-all qosgroup2
  match qos-group 2
class-map match-all qosgroup3
  match qos-group 3
class-map match-all qosgroup4
  match qos-group 4
class-map match-all qosgroup5
  match qos-group 5
class-map match-all qosgroup6
  match qos-group 6
class-map match-all qosgroup7
  match qos-group 7
!
! Class maps to match on input vlan IDs (to be used in parent policy map)
class-map match-all vlan101
  match input  vlan 101
class-map match-all vlan102
  match input  vlan 102
class-map match-all vlan103
  match input  vlan 103
!
policy-map child-pmap
   description Child policy map to shape on the basis of the QoS group values
   class qosgroup1
      shape average 10000000
   class qosgroup2
      shape average 20000000
   class qosgroup5
      shape average 40000000
   class class-default
      shape average 10000000
!
policy-map parent-pmap
   description Parent pmap that shapes traffic for individual VLANs
   class vlan101
```

```
        shape average 70000000
           service-policy child-pmap
    class vlan102
        shape average 80000000
           service-policy child-pmap
    class vlan103
        shape average 90000000
           service-policy child-pmap
    class class-default
        shape average 10000000
```

Configuration Examples for the HQoS for EoMPLS VCs Feature

This section contains the following sample configurations for the HQoS for EoMPLS VC feature:

Simple Hierarchical Configuration Example

The following example shows a simple hierarchical QoS configuration with one parent policy and one child policy. This configuration performs the following:

- The parent policy shapes all outgoing traffic for VLAN 101 on the GE7/1 interface to a total maximum of 90 Mbps.
- The child policy performs LLQ on the VLAN 101 traffic that has the QoS bits set to 1, giving it 10 percent of the bandwidth.
- The child policy allocates 10 percent of the bandwidth of the VLAN 101 traffic that has the QoS bits set to 2.
- The child policy performs WRED on the remaining VLAN 101 traffic.

```
class-map match-any vlan101
  match input vlan 101
class-map match-all qos1
  match qos-group1
class-map match-all qos-group2
  match mpls experimental topmost 2
!
policy-map child-pmap
 class qos1
  priority police percent 10
 class qos-group2
  bandwidth percent 10
 class class-default
  random-detect
policy-map vlan101-pmap
 class vlan101
  shape average 90000000 360000 360000
  service-policy child-pmap

interface GigabitEthernet 7/1
 service-policy output vlan101-pmap
...
```

Complete Hierarchical QoS Example

The following example shows a hierarchical QoS configuration with one parent policy map and two child policy maps. This configuration does the following:

- The input interface (Gigabit Ethernet 2/2) uses the cos-to-qosgroup-pmap policy map to set the QoS group value of incoming packets to match the packets' original 802.1P CoS values.
- The parent policy map shapes traffic for VLAN 101 and 102 to different bandwidths, and applies separate child policy maps to each. The rest of the traffic on the interface is shaped and made subject to the random-detect method.
- The child policy map for VLAN 101 allocates different bandwidth to traffic for QoS groups 1 and 2, and transmits all other traffic on that VLAN unchanged (subject to the parent policy map's bandwidth limitations).
- The child policy map for VLAN 102 marks traffic with an IP precedence value of 4 as priority traffic, and limits all other traffic to 40 percent of the bandwidth (subject to the parent policy map's bandwidth limitations).
- The outgoing interface (POS 8/7) attaches the parent policy map (vlan-parent) for outgoing traffic.

```
class-map match-any vlan101
  match input vlan 101
class-map match-any vlan102
  match input vlan 102
class-map match-all group1
  match qos-group 1
class-map match-all group2
  match qos-group 2

!
policy-map cos-to-qosgroup-pmap
   class class-default
      set qos-group cos

!
policy-map vlan-parent
  description top-level parent policy map
  class vlan101
   shape average 50000000 200000 200000
   service-policy 101qos
  class vlan102
   shape average 100000000 400000 400000
   service-policy 102qos
  class class-default
   shape average 50000000 200000 200000
   random-detect
!
policy-map 101qos
  description child-level policy map for VLAN 101
  class group1
    bandwidth percent 10
  class group2
    bandwidth percent 30
policy-map 102qos
  description child-level policy map for VLAN 102
  class group2
    police percent 10
    priority
  class class-default
    bandwidth percent 40
```

```
!
! Customer-facing interface - the cos-to-qosgroup-pmap policy map sets the
! packet's QoS group value to match the customer's original CoS values.
interface GigabitEthernet2/2
 description Customer-facing interface
 ip address 192.168.100.13 255.255.255.0
 switchport
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 10
 switchport trunk allowed vlan 101-1000,1002-1005
 switchport mode trunk
 mls qos trust cos
 no cdp enable
 service-policy input cos-to-qosgroup-pmap

...

!
interface POS8/7
 description Network-Facing OSM POS
 ip address 10.11.0.5 255.255.255.0
 encapsulation ppp
 tag-switching ip
 mls qos trust dscp
 service-policy output vlan-parent
...
```

Multiple Parent Policies Using the Same Child Policy Example

This portion of a sample configuration file shows several parent policy maps using the same child map.

```
! You can enable QoS globally or per-interface
mls qos
!
class-map match-all group1
  match qos-group 1
class-map match-all group2
  match qos-group 2
class-map match-any vlan101
  match input vlan 101
class-map match-any vlan102
  match input vlan 102
class-map match-any vlan103
  match input vlan 103
class-map match-all exp-3
  match mpls experimental topmost 3
!
policy-map child-pmap
  class group1
    shape average 10000000
  class group2
    shape average 20000000
!
policy-map parent1-pmap
  class vlan101
    shape average 60000000
   service-policy child-pmap
  class vlan102
    shape average 80000000
   service-policy child-pmap
  class class-default
    shape average 100000000
```

```
!
policy-map parent2-pmap
  class vlan103
    shape average 55000000
   service-policy child-pmap
  class exp-3
    shape average 60000000
...
```

Common Class-Map Templates Example

This excerpt from a configuration file gives some common templates for class maps that can be used with your own policy maps.

```
! You can enable QoS globally or per-interface
mls qos

...

! Class Maps to Match on IP Precedence Bits
class-map match-any prec0
  match ip precedence 0
class-map match-any prec1
  match ip precedence 1
class-map match-any prec2
  match ip precedence 2
class-map match-any prec3
  match ip precedence 3
class-map match-any prec4
  match ip precedence 4
class-map match-any prec5
  match ip precedence 5
class-map match-any prec6
  match ip precedence 6
class-map match-any prec7
  match ip precedence 7
! Matches all non-priority precedence values
class-map match-any prec0-4
  match ip precedence 0 1 2 3 4
!
! Class-Maps to Match on QoS Groups
class-map match-all group0
  match qos-group 0
class-map match-all group1
  match qos-group 1
class-map match-all group2
  match qos-group 2
class-map match-all group3
  match qos-group 3
class-map match-all group4
  match qos-group 4
class-map match-all group5
  match qos-group 5
class-map match-all group6
  match qos-group 6
class-map match-all group7
  match qos-group 7
!
! Class Maps to Match on MPLS EXP Bits
class-map match-all exp0
  match mpls experimental topmost 0
class-map match-all exp1
```

```
  match mpls experimental topmost 1
class-map match-all exp2
  match mpls experimental topmost 2
class-map match-all exp3
  match mpls experimental topmost 3
class-map match-all exp4
  match mpls experimental topmost 4
class-map match-all exp5
  match mpls experimental topmost 5
class-map match-all exp6
  match mpls experimental topmost 6
class-map match-all exp7
  match mpls experimental topmost 7
class-map match-all exp1-4
  match mpls experimental topmost 1  2  3  4
!
! Sample Class-MapS to Match on VLAN
! Copy and Change the VLAN Number as Desired
class-map match-any vlan101
  match input vlan 101
class-map match-any vlan102
  match input vlan 102
class-map match-any vlan103
  match input vlan 103
class-map match-any vlan104
  match input vlan 104
class-map match-any vlans101-104
  match input vlan 101-104
!
! Class-Map to Match Any Packet
! (Equivalent to class-default in policy maps)
class-map match-all any-pkt
  match any
```

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.