Applying QoS Features Using the MQC

Restrictions for Applying QoS Features Using the MQC

The MQC-based QoS does not support classification of legacy Layer 2 protocol packets such as Internetwork Packet Exchange (IPX), DECnet, or AppleTalk. When these types of packets are being forwarded through a generic Layer 2 tunneling mechanism, the packets can be handled by MQC but without protocol classification. As a result, legacy protocol traffic in a Layer 2 tunnel is matched only by a "match any" class or class-default.

The number of QoS policy maps and class maps supported varies by platform and release.

Note The policy map limitations do not refer to the number of applied policy map instances, only to the definition of the policy maps.

About

The MQC Structure

The MQC (Modular Quality of Service (QoS) Command-Line Interface (CLI)) enables you to set packet classification and marking based on a QoS group value. MQC CLI allows you to create traffic classes and policies, enable a QoS feature (such as packet classification), and attach these policies to interfaces.

The MQC structure necessitates developing the following entities: traffic class, policy map, and service policy.

Elements of a Traffic Class

A traffic class contains three major elements: a traffic class name, a series of **match** commands, and, if more than one **match** command is used in the traffic class, instructions on how to evaluate these **match** commands.

The **match** commands are used for classifying packets. Packets are checked to determine whether they meet the criteria specified in the **match**commands; if a packet meets the specified criteria, that packet is considered a member of the class. Packets that fail to meet the matching criteria are classified as members of the default traffic class.

Available match Commands

The table below lists *some* of the available **match** commands that can be used with the MQC. The available **match** commands vary by Cisco IOS XE release. For more information about the commands and command syntax, see the *Cisco IOS Quality of Service Solutions* Command Reference.

Table 1: match Commands That Can Be Used with the MQC

Command	Purpose
match access-group	Configures the match criteria for a class map on the basis of the specified access control list (ACL).
match any	Configures the match criteria for a class map to be successful match criteria for all packets.
match cos	Matches a packet based on a Layer 2 class of service (CoS) marking.
match destination-address mac	Uses the destination MAC address as a match criterion.
match discard-class	Matches packets of a certain discard class.
match [**ip**] **dscp**	Identifies a specific IP differentiated service code point (DSCP) value as a match criterion. Up to eight DSCP values can be included in one match statement.
match fr-dlci	Specifies the Frame Relay data-link connection identifier (DLCI) number as a match criterion in a class map.
match input-interface	Configures a class map to use the specified input interface as a match criterion.
match ip rtp	Configures a class map to use the Real-Time Transport Protocol (RTP) port as the match criterion.
match mpls experimental	Configures a class map to use the specified value of the Multiprotocol Label Switching (MPLS) experimental (EXP) field as a match criterion.
match mpls experimental topmost	Matches the MPLS EXP value in the topmost label.

Command	Purpose
match not	Specifies the single match criterion value to use as an unsuccessful match criterion. **Note** The **match not** command, rather than identifying the specific match parameter to use as a match criterion, is used to specify a match criterion that prevents a packet from being classified as a member of the class. For instance, if the **match not qos-group 6** command is issued while you configure the traffic class, QoS group 6 becomes the only QoS group value that is not considered a successful match criterion. All other QoS group values would be successful match criteria.
match packet length	Specifies the Layer 3 packet length in the IP header as a match criterion in a class map.
match port-type	Matches traffic on the basis of the port type for a class map.
match [**ip**] **precedence**	Identifies IP precedence values as match criteria.
match protocol	Configures the match criteria for a class map on the basis of the specified protocol. **Note** A separate **match protocol** (NBAR) command is used to configure network-based application recognition (NBAR) to match traffic by a protocol type known to NBAR.
match protocol fasttrack	Configures NBAR to match FastTrack peer-to-peer traffic.
match protocol gnutella	Configures NBAR to match Gnutella peer-to-peer traffic.
match protocol http	Configures NBAR to match Hypertext Transfer Protocol (HTTP) traffic by URL, host, Multipurpose Internet Mail Extension (MIME) type, or fields in HTTP packet headers.
match protocol rtp	Configures NBAR to match RTP traffic.
match qos-group	Identifies a specific QoS group value as a match criterion.
match source-address mac	Uses the source MAC address as a match criterion.

Multiple match Commands in One Traffic Class

If the traffic class contains more than one **match** command, you need to specify how to evaluate the **match** commands. You specify this by using either the **match-any** or **match-all** keyword of the **class-map** command. Note the following points about the **match-any** and **match-all** keywords:

- If you specify the **match-any** keyword, the traffic being evaluated by the traffic class must match *one* of the specified criteria.
- If you specify the **match-all** keyword, the traffic being evaluated by the traffic class must match *all* of the specified criteria.

- If you do not specify either keyword, the traffic being evaluated by the traffic class must match *all* of the specified criteria (that is, the behavior of the **match-all** keyword is used).

Elements of a Traffic Policy

A traffic policy contains three elements: a traffic policy name, a traffic class (specified with the **class** command), and the command used to enable the QoS feature.

The traffic policy (policy map) applies the enabled QoS feature to the traffic class once you attach the policy map to the interface (by using the **service-policy** command).

Note A packet can match only *one* traffic class within a traffic policy. If a packet matches more than one traffic class in the traffic policy, the *first* traffic class defined in the policy will be used.

Commands Used to Enable QoS Features

The commands used to enable QoS features vary by Cisco IOS XE release. The table below lists *some* of the available commands and the QoS features that they enable. For complete command syntax, see the *Cisco IOS QoS Command Reference*.

For more information about a specific QoS feature that you want to enable, see the appropriate module of the Cisco IOS XE Quality of Service Solutions Configuration Guide.

Table 2: Commands Used to Enable QoS Features

Command	Purpose
bandwidth	Configures a minimum bandwidth guarantee for a class.
bandwidth remaining	Configures an excess weight for a class.
fair-queue	Enables the flow-based queueing feature within a traffic class.
fair-queue pre-classify	Configures and checks whether the **qos pre-classify** command can be used for fair queue. When the **qos pre-classify** command is enabled on the tunnel inteface, and then the **fair-queue pre-classify** command is enabled for the policy-map, the policy-map is attached to either the tunnel interface or the physical interface. The inner IP header of the tunnel will be used for the hash algorithm of the fair queue.
drop	Discards the packets in the specified traffic class.
police	Configures traffic policing.
police (percent)	Configures traffic policing on the basis of a percentage of bandwidth available on an interface.
police (two rates)	Configures traffic policing using two rates, the committed information rate (CIR) and the peak information rate (PIR).

Command	Purpose
priority	Gives priority to a class of traffic belonging to a policy map.
queue-limit	Specifies or modifies the maximum number of packets the queue can hold for a class configured in a policy map.
random-detect	Enables Weighted Random Early Detection (WRED).
random-detect discard-class	Configures the WRED parameters for a discard-class value for a class in a policy map.
random-detect discard-class-based	Configures WRED on the basis of the discard class value of a packet.
random-detect exponential-weighting-constant	Configures the exponential weight factor for the average queue size calculation for the queue reserved for a class.
random-detect precedence	Configure the WRED parameters for a particular IP Precedence for a class policy in a policy map.
service-policy	Specifies the name of a traffic policy used as a matching criterion (for nesting traffic policies [hierarchical traffic policies] within one another).
set atm-clp	Sets the cell loss priority (CLP) bit when a policy map is configured.
set cos	Sets the Layer 2 class of service (CoS) value of an outgoing packet.
set discard-class	Marks a packet with a discard-class value.
set [**ip**] **dscp**	Marks a packet by setting the differentiated services code point (DSCP) value in the type of service (ToS) byte.
set fr-de	Changes the discard eligible (DE) bit setting in the address field of a Frame Relay frame to 1 for all traffic leaving an interface.
set mpls experimental	Designates the value to which the MPLS bits are set if the packets match the specified policy map.
set precedence	Sets the precedence value in the packet header.
set qos-group	Sets a QoS group identifier (ID) that can be used later to classify packets.
shape	Shapes traffic to the indicated bit rate according to the algorithm specified.

Nested Traffic Classes

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

In a scenario where packets satisfy more than one match criterion, the MQC enables you to associate multiple traffic classes with a single traffic policy (also termed nested traffic classes) using the **match class-map** command. (We term these *nested class maps* or *MQC Hierarchical class maps*.) This command provides the only method of combining match-any and match-all characteristics within a single traffic class. By doing so, you can create a traffic class using one match criterion evaluation instruction (either match-any or match-all) and then use that traffic class as a match criterion in a traffic class that uses a different match criterion type. For example, a traffic class created with the match-any instruction must use a class configured with the match-all instruction as a match criterion, or vice versa.

Consider this likely scenario: Suppose A, B, C, and D were all separate match criterion, and you wanted traffic matching A, B, or C and D (i.e., A or B or [C and D]) to be classified as belonging to a traffic class. Without the nested traffic class, traffic would either have to match all four of the match criterion (A and B and C and D) or match any of the match criterion (A or B or C or D) to be considered part of the traffic class. You would not be able to combine "and" (match-all) and "or" (match-any) statements within the traffic class; you would be unable to configure the desired configuration.

The solution: Create one traffic class using match-all for C and D (which we will call criterion E), and then create a new match-any traffic class using A, B, and E. The new traffic class would have the correct evaluation sequence (A or B or E, which is equivalent to A or B or [C and D]).

match-all and match-any Keywords of the class-map Command

One of the commands used when you create a traffic class is the **class-map**command. The command syntax for the **class-map** command includes two keywords: **match-all** and **match-any**. The **match-all** and **match-any** keywords need to be specified only if more than one match criterion is configured in the traffic class. Note the following points about these keywords:

- The **match-all** keyword is used when *all* of the match criteria in the traffic class must be met in order for a packet to be placed in the specified traffic class.
- The **match-any** keyword is used when only *one* of the match criterion in the traffic class must be met in order for a packet to be placed in the specified traffic class.
- If neither the **match-all** keyword nor **match-any** keyword is specified, the traffic class will behave in a manner consistent with the **match-all** keyword.

input and output Keywords of the service-policy Command

As a general rule, the QoS features configured in the traffic policy can be applied to packets entering the interface or to packets leaving the interface. Therefore, when you use the **service-policy** command, you need to specify the direction of the traffic policy by using the **input** or **output** keyword.

For instance, the **service-policy output policy-map1** command would apply the QoS features in the traffic policy to the interface in the output direction. All packets leaving the interface (output) are evaluated according to the criteria specified in the traffic policy named policy-map1.

Note For Cisco releases, queueing mechanisms are not supported in the input direction. Nonqueueing mechanisms (such as traffic policing and traffic marking) are supported in the input direction. Also, classifying traffic on the basis of the source MAC address (using the **match source-address mac** command) is supported in the input direction only.

Benefits of Applying QoS Features Using the MQC

The MQC structure allows you to create the traffic policy (policy map) once and then apply it to as many traffic classes as needed. You can also attach the traffic policies to as many interfaces as needed.

How to Apply QoS Features Using the MQC

Creating a Traffic Class

To create a traffic class, use the **class-map** command to specify the traffic class name. Then use one or more **match** commands to specify the appropriate match criteria. Packets matching the criteria that you specify are placed in the traffic class. For more information about the **match-all** and **match-any** keywords of the class-map comand, see the “match-all and match-any Keywords of the class-map Command” section.

Note The **match cos** command is shown in Step 4. The **match cos** command is simply an example of one of the **match** commands that you can use. For information about the other available **match** commands, see the “match-all and match-any Keywords of the class-map Command” section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **class-map** [**match-all** | **match-any**] *class-map-name*
4. **match cos** *cos-number*
5. Enter additional match commands, if applicable; otherwise, continue with step 6.
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**class-map** [**match-all** | **match-any**] *class-map-name* **Example:** `Router(config)# class-map match-any class1`	Creates a class to be used with a class map and enters class-map configuration mode. • The class map is used for matching packets to the specified class.

	Command or Action	Purpose
		• Enter the class name. **Note** The **match-all** keyword specifies that all match criteria must be met. The **match-any** keyword specifies that one of the match criterion must be met. Use these keywords only if you will be specifying more than one **match** command.
Step 4	**match cos** *cos-number* **Example:** `Router(config-cmap)# match cos 2`	Matches a packet on the basis of a Layer 2 class of service (CoS) number. • Enter the CoS number. **Note** The **match cos** command is an example of the **match** commands you can use. For information about the other **match** commands that are available, see the "match-all and match-any Keywords of the class-map Command" section.
Step 5	Enter additional match commands, if applicable; otherwise, continue with step 6.	--
Step 6	**end** **Example:** `Router(config-cmap)# end`	(Optional) Exits QoS class-map configuration mode and returns to privileged EXEC mode.

Creating a Traffic Policy

Note The **bandwidth** command is shown in Step 5. The **bandwidth** command is an example of the commands that you can use in a policy map to enable a QoS feature (in this case, Class-based Weighted Fair Queuing (CBWFQ). For information about other available commands, see the "Elements of a Traffic Policy" section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class** {*class-name* | **class-default**}
5. **bandwidth** {*bandwidth-kbps* | **percent** *percent*}
6. Enter the commands for any additional QoS feature that you want to enable, if applicable; otherwise, continue with Step 7.
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:** `Router(config)# policy-map policy1`	Creates or specifies the name of the traffic policy and enters QoS policy-map configuration mode. • Enter the policy map name.
Step 4	**class** {*class-name* | **class-default**} **Example:** `Router(config-pmap)# class class1`	Specifies the name of a traffic class and enters QoS policy-map class configuration mode. **Note** This step associates the traffic class with the traffic policy.
Step 5	**bandwidth** {*bandwidth-kbps* | **percent** *percent*} **Example:** `Router(config-pmap-c)# bandwidth 3000`	(Optional) Specifies a minimum bandwidth guarantee to a traffic class in periods of congestion. • A minimum bandwidth guarantee can be specified in kb/s or by a percentage of the overall available bandwidth. **Note** The **bandwidth** command enables CBWFQ. The **bandwidth** command is an example of the commands that you can use in a policy map to enable a QoS feature. For information about the other commands available, see the “Elements of a Traffic Policy” section.
Step 6	Enter the commands for any additional QoS feature that you want to enable, if applicable; otherwise, continue with Step 7.	--
Step 7	**end** **Example:** `Router(config-pmap-c)# end`	(Optional) Exits QoS policy-map class configuration mode and returns to privileged EXEC mode.

Attaching a Traffic Policy to an Interface Using the MQC

Note Cisco IOS XE Release 2.3.0 and later releases do not support the attachment of policies for ATM interfaces that have unspecified bit rate (UBR) configured as the default mode on their VC or virtual path (VP). An attempt to use this configuration results in an error message: CBWFQ: Not supported on ATM interfaces with UBR configuration. You can also specify UBR with a rate in the UBR configuration, if you do not want to use the default UBR value.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **service-policy** {**input** | **output**} *policy-map-name*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface serial 0/0/1`	Configures an interface type and enters interface configuration mode. • Enter the interface type and interface number.
Step 4	**service-policy** {**input** | **output**} *policy-map-name* **Example:** `Router(config-if)# service-policy input policy1`	Attaches a policy map to an interface. • Enter either the **input** or **output** keyword and the policy map name.
Step 5	**end** **Example:** `Router(config-if)# end`	(Optional) Exits interface configuration mode and returns to privileged EXEC mode.

Verifying the Traffic Class and Traffic Policy Information

The show commands described in this section are optional and can be entered in any order.

SUMMARY STEPS

1. **enable**
2. **show class-map**
3. **show policy-map** *policy-map-name* **class** *class-name*
4. **show policy-map**
5. **show policy-map interface** *type number*
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show class-map** **Example:** `Router# show class-map`	(Optional) Displays all class maps and their matching criteria.

Step 3	**show policy-map** *policy-map-name* **class** *class-name*
	Example:
	`Router#`
	`show policy-map policy1 class class1`
		(Optional) Displays the configuration for the specified class of the specified policy map.
		• Enter the policy map name and the class name.

Step 4	**show policy-map** **Example:** `Router# show policy-map`	(Optional) Displays the configuration of all classes for all existing policy maps.
Step 5	**show policy-map interface** *type number* **Example:** `Router# show policy-map interface serial 0/0/1`	(Optional) Displays the statistics and the configurations of the input and output policies that are attached to an interface. • Enter the interface type and number.
Step 6	**exit** **Example:** `Router# exit`	(Optional) Exits privileged EXEC mode.

Configuration Examples for Applying QoS Features Using the MQC

Creating a Traffic Class

In the following example, we create traffic classes and define their match criteria. For the first traffic class (class1), we use access control list (ACL) 101 as match criteria; for the second traffic class (class2), ACL 102. We check the packets against the contents of these ACLs to determine if they belong to the class.

```
class-map class1
  match access-group 101
  exit
class-map class2
  match access-group 102
  end
```

Creating a Traffic Class

In the following example, we create traffic classes and define their match criteria. For the first traffic class (class1), we use access control list (ACL) 101 as match criteria; for the second traffic class (class2), ACL 102. We check the packets against the contents of these ACLs to determine if they belong to the class.

```
class-map class1
  match access-group 101
  exit
class-map class2
  match access-group 102
  end
```


Example: Attaching a Traffic Policy to an Interface

The following example shows how to attach an existing traffic policy to an interface. After you define a traffic policy with the **policy-map** command, you can attach it to one or more interfaces by using the **service-policy** command in interface configuration mode. Although you can assign the same traffic policy to multiple interfaces, each interface can have only one traffic policy attached in the input direction and only one traffic policy attached in the output direction.

```
Router(config)# interface fastethernet 1/1/1
Router(config-if)# service-policy output policy1
Router(config-if)# exit
Router(config)# interface fastethernet 1/0/0
Router(config-if)# service-policy output policy1
Router(config-if)# end
```

Using the match not Command

Use the **match not** command to specify a QoS policy value that is not used as a match criterion. All other values of that QoS policy become successful match criteria. For instance, if you issue the **match not qos-group**

4 command in QoS class-map configuration mode, the specified class will accept all QoS group values except 4 as successful match criteria.

In the following traffic class, all protocols except IP are considered successful match criteria:

```
class-map noip
  match not protocol ip
  end
```

Configuring a Default Traffic Class

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

If you do not configure a default class, packets are still treated as members of that class. The default class has no QoS features enabled so packets belonging to this class have no QoS functionality. Such packets are placed into a first-in, first-out (FIFO) queue managed by tail drop, which is a means of avoiding congestion that treats all traffic equally and does not differentiate between classes of service. Queues fill during periods of congestion. When the output queue is full and tail drop is active, packets are dropped until the congestion is eliminated and the queue is no longer full.

The following example configures a policy map (policy1) for the default class (always called class-default) with these characteristics: 10 queues for traffic that does not meet the match criteria of other classes whose policy is defined by class policy1, and a maximum of 20 packets per queue before tail drop is enacted to handle additional queued packets.

In the following example, we configure a policy map (policy1) for the default class (always termed class-default) with these characteristics: 10 queues for traffic that does not meet the match criterion of other classes whose policy is defined by the traffic policy policy1.

```
policy-map policy1
  class class-default
    shape average 100m
```

How "fair-queue" Supports "pre-classify" Command

Prior to the Cisco IOS 16.4 release, when you configure fair-queue on the tunnel interface, the outer IP header of the tunnel was used for the hash algorithm of fair queue. Therefore, the packets of all flows on the tunnel were put into the same flow queue. This is the behavior seen even when the **qos pre-classify** command is configured on the tunnel interface

From the Cisco IOS 16.4 release onwards, **fair-queue** supports the **pre-classify** command. This command is added so that **qos pre-classify** can be used with the **fair-queue** command.

The following example configures **fair-queue pre-classify** command for policy-map under class configuration:

```
interface tunnel 0
    qos pre-classify
policy-map po1
    class c1
        shape average percentage 10
        fair-queue pre-classify
```

When **qos pre-classify** is enabled on the tunnel interface, and the **fair-queue pre-classify** is enabled for policy-map, then the policy-map is attached to either the tunnel interface or the physical interface. The inner IP header of the tunnel is used for the hash algorithm of the fair queue.

To disable this feature, use the **fair-queue** command without the **pre-classify** keyword.

The default behavior of fair queue remains unchanged.

How Commands "class-map match-any" and "class-map match-all" Differ

This example shows how packets are evaluated when multiple match criteria exist. It illustrates the difference between the **class-map match-any** and **class-map match-all** commands. Packets must meet either all of the match criteria (**match-all**) or one of the match criteria (**match-any**) to be considered a member of the traffic class.

The following examples show a traffic class configured with the **class-map match-all** command:

```
class-map match-all cisco1
  match qos-group 4
  match access-group 101
```

If a packet arrives on a router with traffic class cisco1 configured on the interface, we assess whether it matches the IP protocol, QoS group 4, and access group 101. If all of these match criteria are met, the packet is classified as a member of the traffic class cisco1 (a logical AND operator; Protocol IP AND QoS group 4 AND access group 101).

```
class-map match-all vlan
  match vlan 1
  match vlan inner 1
```

The following example illustrates use of the **class-map match-any** command. Only one match criterion must be met for us to classify the packet as a member of the traffic class (i.e., a logical OR operator; protocol IP OR QoS group 4 OR access group 101):

```
class-map match-any cisco2
  match protocol ip
  match qos-group 4
  match access-group 101
```

In the traffic class cisco2, the match criterion are evaluated consecutively until a successful match is located. The packet is first evaluated to determine whether the IP protocol can be used as a match criterion. If so, the packet is matched to traffic class cisco2. If not, then QoS group 4 is evaluated as a match criterion and so on. If the packet matches none of the specified criteria, the packet is classified as a member of the default traffic class (*class default-class*).

Establishing Traffic Class as a Match Criterion (Nested Traffic Classes)

There are two reasons to use the **match class-map** command. One reason is maintenance; if a large traffic class currently exists, using the traffic class match criterion is easier than retyping the same traffic class configuration. The second and more common reason is to mix match-all and match-any characteristics in one traffic policy. This enables you to create a traffic class using one match criterion evaluation instruction (either match-any or match-all) and then use that traffic class as a match criterion in a traffic class that uses a different match criterion type.

Consider this likely scenario: Suppose A, B, C, and D were all separate match criterion, and you wanted traffic matching A, B, or C and D (i.e., A or B or [C and D]) to be classified as belonging to a traffic class. Without the nested traffic class, traffic would either have to match all four of the match criterion (A and B and C and D) or match any of the match criterion (A or B or C or D) to be considered part of the traffic class. You would not be able to combine “and” (match-all) and “or” (match-any) statements within the traffic class; you would be unable to configure the desired configuration.

The solution: Create one traffic class using match-all for C and D (which we will call criterion E), and then create a new match-any traffic class using A, B, and E. The new traffic class would have the correct evaluation sequence (A or B or E, which is equivalent to A or B or [C and D]).

Example: Nested Traffic Class for Maintenance

In the following example, the traffic class class1 has the same characteristics as the traffic class class2, with the exception that the former has added a destination address as a match criterion. Rather than configuring traffic class class1 line by line, you can enter the **match class-map class2** command. This command allows you to include all of the characteristics in the traffic class called class2 in the traffic class class1, and you can add the new destination address match criterion without reconfiguring the entire traffic class.

```
Router(config)# class-map match-any class2
Router(config-cmap)# match protocol ip
Router(config-cmap)# match qos-group 3
Router(config-cmap)# match access-group 2
Router(config-cmap)# exit
Router(config)# class-map match-all class1
Router(config-cmap)# match class-map class2
Router(config-cmap)# match destination-address mac 00.00.00.00.00.00
Router(config-cmap)# exit
```

Example: Nested Traffic Class to Combine match-any and match-all Characteristics in One Traffic Class

The only method of including both match-any and match-all characteristics in a single traffic class is to use the **match class-map** command. To combine match-any and match-all characteristics into a single class, use the match-any instruction to create a traffic class that uses a class configured with the match-all instruction as a match criterion (through the **match class-map** command).

The following example shows how to combine the characteristics of two traffic classes, one with match-any and one with match-all characteristics, into one traffic class with the **match class-map** command. The result requires a packet to match one of the following three match criteria to be considered a member of traffic class class4: IP protocol *and* QoS group 4, destination MAC address 00.00.00.00.00.00, or access group 2.

In this example, only the traffic class called class4 is used with the traffic policy called policy1.

```
Router(config)# class-map match-all class3
Router(config-cmap)# match protocol ip
Router(config-cmap)# match qos-group 4
Router(config-cmap)# exit
Router(config)# class-map match-any class4
Router(config-cmap)# match class-map class3
Router(config-cmap)# match destination-address mac 00.00.00.00.00.00
Router(config-cmap)# match access-group 2
Router(config-cmap)# exit
Router(config)# policy-map policy1
Router(config-pmap)# class class4
Router(config-pmap-c)# police 8100 1500 2504 conform-action transmit exceed-action
```

```
set-qos-transmit 4
Router(config-pmap-c)# end
```

Example: Traffic Policy as a QoS Policy (Hierarchical Traffic Policies)

A traffic policy can be included in a QoS policy when the **service-policy** command is used in QoS policy-map class configuration mode. A traffic policy that contains a traffic policy is called a hierarchical traffic policy.

A hierarchical traffic policy contains a child policy and a parent policy. The child policy is the previously defined traffic policy that is being associated with the new traffic policy through the use of the **service-policy** command. The new traffic policy using the preexisting traffic policy is the parent policy. In the example in this section, the traffic policy called child is the child policy and traffic policy called parent is the parent policy.

Hierarchical traffic policies can be attached to subinterfaces and ATM PVCs. When hierarchical traffic policies are used, a single traffic policy (with a child and a parent policy) can be used to shape and prioritize permanent virtual connection (PVC) traffic. In the following example, the child policy is responsible for prioritizing traffic and the parent policy is responsible for shaping traffic. In this configuration, the parent policy allows packets to be sent from the interface, and the child policy determines the order in which the packets are sent.

```
Router(config)# policy-map child
Router(config-pmap)# class voice
Router(config-pmap-c)# priority ?
384-100000000 Kilo Bits per second
level Multi-Level Priority Queue
percent % of total bandwidth
Router(config-pmap-c)# priority 50
Router(config)# policy-map parent
Router(config-pmap)# class class-default
Router(config-pmap-c)# shape average 10000000
Router(config-pmap-c)# service-policy child
```

The value used with the **shape** command is provisioned from the committed information rate (CIR) value from the service provider.

Additional References

Related Documents

Related Topic	Document Title
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Quality of Service Solutions Command Reference*
Packet classification	“Classifying Network Traffic” module
Frame Relay Fragmentation (FRF) PVCs	“FRF .20 Support” module
Selective Packet Discard	“IPv6 Selective Packet Discard” module

Related Topic	Document Title
Scaling and performance information	"Broadband Scalability and Performance" module of the Cisco ASR 1000 Series Aggregation Services Routers Software Configuration Guide .

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Applying QoS Features Using the MQC

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Table 3: Feature Information for Applying QoS Features Using the MQC

Feature Name	Releases	Feature Information
Class-Based Weighted Fair Queueing (CBWFQ)	Cisco IOS XE Release 2.1 Cisco IOS XE Release 3.5S	This feature was introduced on Cisco ASR 1000 Series Aggregation Services Routers. In Cisco IOS XE Release 3.5S, support was added for the Cisco ASR 903 Router.

Feature Name	Releases	Feature Information
Modular QoS CLI (MQC)	Cisco IOS XE Release 2.1 Cisco IOS XE Release 3.5S	This module describes how to apply and configure quality of service (QoS) features using the modular QoS CLI (MQC). The MQC allows you to define a traffic class, create a traffic policy (policy map), and attach the traffic policy to an interface. The traffic policy contains the QoS feature that will be applied to the traffic class. This feature was introduced on Cisco ASR 1000 Series Aggregation Services Routers. This feature was enhanced to provide infrastructure support for additional features included with Cisco IOS XE Release 2.3. In Cisco IOS XE Release 3.5S, support was added for the Cisco ASR 903 router.
MQC Hierarchical Class Map	Cisco IOS XE Release 3.2	MQC allows multiple traffic classes (nested traffic classes, which are also called nested class maps or MQC hierarchical class maps) to be configured as a single traffic class. This feature was introduced on Cisco ASR 1000 Series Aggregation Services Routers.
Priority Queueing	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Aggregation Services Routers.
Weighted Random Early Detection	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Aggregation Services Routers.