



Applying QoS Features Using the MQC

This module contains the concepts about applying QoS features using the Modular Quality of Service (QoS) Command-Line Interface (CLI) (MQC) and the tasks for configuring the 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.

- [Finding Feature Information, page 1](#)
- [Restrictions for Applying QoS Features Using the MQC, page 1](#)
- [Information About Applying QoS Features Using the MQC, page 3](#)
- [How to Apply QoS Features Using the MQC, page 8](#)
- [Configuration Examples for Applying QoS Features Using the MQC, page 14](#)
- [Additional References, page 17](#)
- [Feature Information for Applying QoS Features Using the MQC, page 17](#)

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see [Bug Search Tool](#) and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

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

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. See the following tables for the supported platform information.

**Note**

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

The following restrictions apply to Cisco IOS XE Release 3.5S for the Cisco ASR 903 router:

- QoS policy maps are not supported in sessions.
- Nested traffic maps are not supported.

For more information on restrictions for Cisco ASR 903 router, refer the [Quality of Service Configuration Guidelines for Cisco ASR 903 Router](#).

Table 1: Cisco ASR 903 Policy and Class Map Support

	Cisco IOS XE 3.5S, 3.6S, 3.7S, 3.8S	Cisco IOS XE 3.9S and higher
Number of unique policy-maps	1024	1024
Number of unique class-maps	4096	4096
Number of classes per policy-map	512	4096
Number of filters per class-map	16	16

Table 2: Cisco ASR 1000 Series Policy and Class Map Support

	Cisco IOS XE 2.0S, 2.1S, 2.2S	Cisco IOS XE 2.3S	Cisco IOS XE 3.5S, 3.6S, 3.7S, 3.8S, 3.9S	Cisco IOS XE 3.10S and higher
Number of unique policy-maps	1024	4096	4096	4096 16000 (RP2, ESP40, ESP100, ESP200 models only)
Number of unique class-maps	4096	4096	4096	4096
Number of classes per policy-map	8	256	1000	1000
Number of filters per class-map	16	16	32	32

Table 3: Cisco CSR 1000V Policy and Class Map Support

	Cisco IOS XE 3.9S and higher
Number of unique policy-maps	30
Number of unique class-maps	256
Number of classes per policy-map	32
Number of filters per class-map	8

Table 4: Cisco ISR 4451-X Policy and Class Map Support

	Cisco IOS XE 3.9S and higher
Number of unique policy-maps	4000
Number of unique class-maps	4096
Number of classes per policy-map	256
Number of filters per class-map	32

Information About Applying QoS Features Using the MQC

The MQC Structure

The MQC structure allows you to define a traffic class, create a traffic policy, and attach the traffic policy to an interface.

The MQC structure consists of the following three high-level steps:

- 1 Define a traffic class by using the **class-map** command. A traffic class is used to classify traffic.
- 2 Create a traffic policy by using the **policy-map** command. (The terms *traffic policy* and *policy map* are often synonymous.) A traffic policy (policy map) contains a traffic class and one or more QoS features that will be applied to the traffic class. The QoS features in the traffic policy determine how to treat the classified traffic.
- 3 Attach the traffic policy (policy map) to the interface by using the **service-policy** command.

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 5: 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 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 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.
match-all	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
match-any	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
match [ip] precedence	Identifies IP precedence values as match criteria.
match qos-group	Identifies a specific QoS group value as a match criterion.
match service-instance ethernet id	Matches using service instance ID.

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 6: 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.
drop	Discards the packets in the specified traffic class.
police	Configures traffic policing.

Command	Purpose
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).
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 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 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.

Command	Purpose
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. When packets meet more than one match criterion, multiple traffic classes can be associated with a single traffic policy.

Similarly, the 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 nesting can be achieved with the use of the **match class-map** command. The only method of combining match-any and match-all characteristics within a single traffic class is with the **match class-map** command.

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:



Note

match-all class-map is supported for cos and inner-cos and (vlan and inner-vlan) on the Cisco ASR 900 Series router.

- 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 IOS XE Release 2.1 and later 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 command, 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	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: <pre>Router> enable</pre>	Enter your password if prompted.
Step 2	configure terminal Example: <pre>Router# configure terminal</pre>	Enters global configuration mode.
Step 3	class-map [match-all match-any] <i>class-map-name</i> Example: <pre>Router(config)# class-map match-any class1</pre>	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. - 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 <i>cos-number</i> Example: <pre>Router(config-cmap)# match cos 2</pre>	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: <pre>Router(config-cmap)# end</pre>	(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 <i>policy-map-name</i> 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 { <i>class-name</i> 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 { <i>bandwidth-kbps</i> 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.

	Command or Action	Purpose
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



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.

	Command or Action	Purpose
Step 2	configure terminal Example: Router# configure terminal	Enters global configuration mode.
Step 3	interface type number Example: Router(config)# interface gigabit 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	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: Router> enable	• 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 <i>policy-map-name</i> class <i>class-name</i> 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 <i>type number</i> Example: Router# show policy-map interface gigabit 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

Example: Creating a Traffic Class

In the following example, two traffic classes are created and their match criteria are defined. For the first traffic class called class1, access control list (ACL) 101 is used as the match criterion. For the second traffic class called class2, ACL 102 is used as the match criterion. Packets are checked against the contents of these ACLs to determine if they belong to the class.

```
Router(config)# class-map class1
Router(config-cmap)# match access-group 101
Router(config-cmap)# exit
Router(config)# class-map class2
Router(config-cmap)# match access-group 102
Router(config-cmap)# end
```

Example Creating a Traffic Policy

In the following example, a traffic policy called policy1 is defined. The traffic policy contains the QoS features to be applied to two classes--class1 and class2. The match criteria for these classes were previously defined (as described in the Example Creating a Traffic Class).

For class1, the policy includes a bandwidth allocation request and a maximum packet count limit for the queue reserved for the class. For class2, the policy specifies only a bandwidth allocation request.

```
Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth 3000
Router(config-pmap-c)# queue-limit 30
Router(config-pmap-c)# exit
Router(config-pmap)# class class2
Router(config-pmap-c)# bandwidth 2000
Router(config-pmap-c)# 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 gigabitethernet0/3/6
Router(config-if)# service instance 1 ethernet
Router(config-if-srv)# service-policy input col
Router(config-if-srv)# encapsulation dot1q 1
Router(config-if-srv)# bridge-domain 1
Router(config-if)# service-policy output policy1
Router(config-if)# end
```

Example: Default Traffic Class Configuration

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

If you do not configure a default class, packets are still treated as members of the default class. However, by default, the default class has no QoS features enabled. Therefore, packets belonging to a default class have no QoS functionality. These packets are placed into a first-in, first-out (FIFO) queue managed by tail drop. Tail drop 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 in effect, packets are dropped until the congestion is eliminated and the queue is no longer full.

The following example configures a traffic policy for the default class of the traffic policy called policy1. The default class (which is always called class-default) has these characteristics: 10 queues for traffic that does not meet the match criteria of other classes whose policy is defined by the traffic policy policy1.

```
Router(config)# policy-map policy1
Router(config-pmap)# class class-default
Router(config-pmap-c0)#shape average 100m
```

Example: class-map match-any and class-map match-all Commands

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

The following example shows a traffic class configured with the **class-map match-all** command:

```
Router(config)# class-map match-all vlan
Router(config-cmap)#match vlan 1
Router(config-cmap)#match vlan inner 1
```

If a packet arrives on a router with the traffic class called cisco1 configured on the interface, the packet is evaluated to determine if it matches the IP protocol, QoS group 4, *and* access group 101. If all three of these match criteria are met, the packet is classified as a member of the traffic class cisco1.

The following example shows a traffic class that is configured with the **class-map match-any** command:

```
Router(config)# class-map match-any col
Router(config-cmap)#match dscp 1
Router(config-cmap)# match prec 1
Router(config-cmap)# match cos 1
```

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

Note that the **class-map match-all** command requires that *all* of the match criteria be met in order for the packet to be considered a member of the specified traffic class (a logical AND operator). In the first example, protocol IP AND QoS group 4 AND access group 101 must be successful match criteria. However, only one match criterion must be met in order for the packet in the **class-map match-any** command to be classified

as a member of the traffic class (a logical OR operator). In the second example, protocol IP OR QoS group 4 OR access group 101 must be successful match criterion.

Example: 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 more common reason for the **match class-map** command is to allow users to use match-any and match-all statements in the same traffic class. If you want to combine match-all and match-any characteristics in a traffic policy, create a traffic class using one match criterion evaluation instruction (either match-any or match-all) and then use this traffic class as a match criterion in a traffic class that uses a different match criterion type.

Here is a possible scenario: Suppose A, B, C, and D were all separate match criterion, and you wanted traffic matching A, B, or C and D (A or B or [C and D]) to be classified as belonging to the 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, and you would therefore 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 would also be A or B or [C and D]). The desired traffic class configuration has been achieved.

The only method of mixing match-all and match-any statements in a traffic class is through the use of the traffic class match criterion.

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 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
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Cisco IOS Quality of Service Solutions Command Reference</i>
Packet classification	“Classifying Network Traffic” module
Frame Relay Fragmentation (FRF) PVCs	“FRF .20 Support” module
Selective Packet Discard	“IPv6 Selective Packet Discard” module
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 www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 7: 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.
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.

