QoS: Modular QoS Command-Line Interface Configuration Guide, Cisco IOS XE Release 2

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Applying QoS Features Using the MQC

This module describes how to apply and configure QoS features using the modular quality of service (QoS) 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

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see 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 document.

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 does not support Internetwork Packet Exchange (IPX) packets.

The number of QoS class maps supported in a single policy map varies by release, as follows:

- For Cisco IOS XE Release 2.1 and 2.2, the MQC supports a maximum of eight class maps in a single policy map.
- For Cisco IOS XE Release 2.3, the MQC supports a maximum of 256 class maps in a single policy map.

The number of QoS class maps supported on a router varies by release, as follows:

- For Cisco IOS XE Release 2.1 and 2.2, the MQC supports no more than 1000 policy maps in the incoming (ingress) direction, outgoing (egress) direction, or a combination of both directions on a router.

- For Cisco IOS XE Release 2.3, the MQC supports no more than 4000 policy maps in the incoming (ingress) direction, outgoing (egress) direction, or a combination of both directions on a router.

When sessions are created and QoS policy maps are attached in both the ingress and egress directions, only 2000 sessions are supported. Sessions exceeding this limit can still be created, but the QoS policy maps will not be applied to the session.

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.

Information About Applying QoS Features Using the MQC

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 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.
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.

Command	Purpose
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 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.

Command	Purpose
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. 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:

- 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 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.

	Command or Action	Purpose
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. • 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.

	Command or Action	Purpose
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 A traffic policy containing a queueing mechanism or feature cannot be attached to a physical interface or subinterface.

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

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 “Elements of a Traffic Policy”).

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)# exit
```

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
```

Example: match not Command

The **match not**command is used to specify a specific QoS policy value that is not used as a match criterion. If the **match not** command is issued, all other values of that QoS policy become successful match criteria. For instance, if the **match not qos-group 4** command is issued 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:

```
Router(config)# class-map noip
Router(config-cmap)# match not protocol ip
Router(config-cmap)# 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, and a maximum of 20 packets per queue before tail drop is enacted to handle additional queued packets.

```
Router(config)# policy-map policy1
Router(config-pmap)# class class-default
Router(config-pmap-c)# fair-queue
Router(config-pmap-c)# queue-limit 20
```

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**c ommand:

```
Router(config)# class-map match-all cisco1
Router(config-cmap)# match protocol ip
Router(config-cmap)# match qos-group 4
Router(config-cmap)# match access-group 101
```

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 cisco2
Router(config-cmap)# match protocol ip
```

```
Router(config-cmap)# match qos-group 4
Router(config-cmap)# match access-group 101
```

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: Nested Traffic Class for Maintenance, page 16
- Example: Nested Traffic Class to Combine match-any and match-all Characteristics in One Traffic Class, page 17

Example: Nested Traffic Class for Maintenance

In the following example, the traffic class called class1 has the same characteristics as the traffic class called class2, with the exception that traffic class class1 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 all of the characteristics in the traffic class called class2 to be

included in the traffic class called 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 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	*Cisco IOS Quality of Service Solutions Command Reference*
Packet classification	"Classifying Network Traffic" module
Frame Relay Fragmentation (FRF) PVCs	"FRF .20 Support" module

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 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.
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.

Feature Name	Releases	Feature Information
Weighted Random Early Detection	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Aggregation Services Routers.

QoS Policies Aggregation

The QoS: Policies Aggregation (QoS: Policies Aggregation) feature for the Cisco ASR 1000 Series Aggregation Services Routers supports Modular Quality of Service Command-Line Interface (MQC) configuration of default traffic classes in policy maps on different subinterfaces to be queued as a single, user-defined traffic class at the main interface policy map. It is most useful in QoS configurations where you have several subinterface policy maps on the same physical interface and you want identical treatment of the default traffic classes on those subinterfaces.

Beginning in Cisco IOS XE Release 2.6, the QoS: Policies Aggregation feature is enhanced to support queueing aggregation at the primary interface for other traffic classes, including Differentiated Services Code Point (DSCP) traffic classes such as the expedited forwarding (EF), Assured Forwarding 1 (AF1), and AF4 traffic classes. With this enhancement, any traffic classes from VLAN subinterfaces can share a common queue for that traffic class at the main interface policy map. Other enhancements include the ability to configure and show drop statistics that occur at the aggregate level for these classes.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see 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 document.

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.

Prerequisites for QoS Policies Aggregation

- This feature is configured using the Modular Quality of Service (QoS) Command-Line Interface (CLI) (MQC).
- All traffic over the main interface should come through one or more subinterfaces.

Restrictions for QoS Policies Aggregation

- Applies only when multiple subinterfaces with policy maps are attached to the same physical interface. This feature cannot be used to collectively classify default traffic classes or other traffic classes of policy maps on different physical interfaces.
- Certain traffic class configuration prior to Cisco IOS XE Release 2.6 at the subinterface policy-map and main-interface policy-map will have different behavior and queueing results. See the Understanding the QoS Policies Aggregation MQC, page 23 and Differences Between the Original Feature and the MQC Support for Multiple Queue Aggregation, page 24.

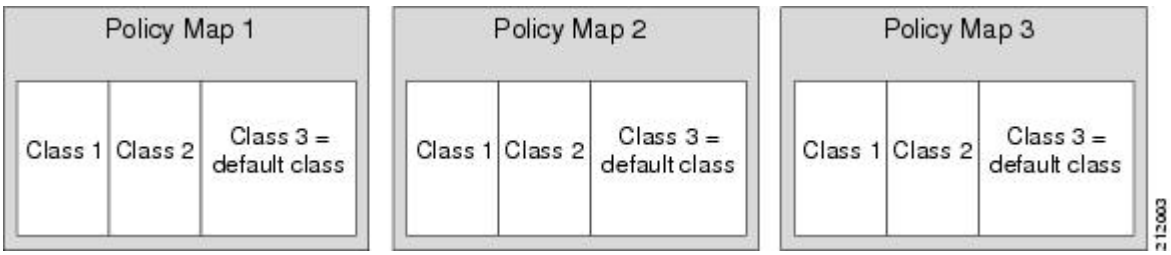
Information About QoS Policies Aggregation

Understanding Fragments in Class Definition Statements

QoS: Policies Aggregation introduces the idea of fragments in class definition statements. A default traffic class definition statement can be marked as a fragment within a policy map. Other policy maps on the same interface can also define their default traffic class statements as fragments, if desired. A separate policy map can then be created with a service fragment class definition statement that will be used to apply QoS to all of the fragments as a single group.

The figure below provides an example of one physical interface with three attached policy maps that is not using fragments. Note that each policy map has a default traffic class that can only classify traffic for the default traffic within its own policy map.

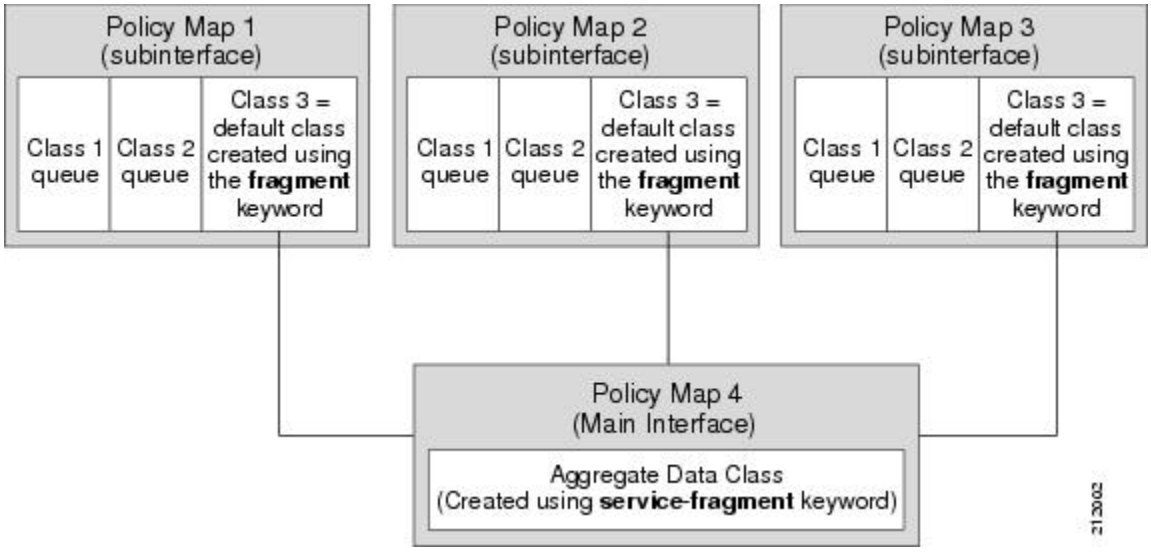
Figure 1 Three Policy Maps Configured Without Fragments



The figure below shows the same configuration configured with fragments and adds a fourth policy map with a class definition statement that classifies the fragments collectively. The default traffic classes are

now classified as one service fragment group rather than three separate default traffic classes within the individual policy maps.

Figure 2 Three Policy Maps Configured Using Fragments



Understanding Fragments for Gigabit Etherchannel Bundles

Fragments can be configured for Gigabit Etherchannels when all of the member links of the Gigabit Etherchannel (GEC) bundle are on the same physical interface. Notably, if VLANs on the same physical interface are bundled, fragments can be used to define the collective treatment of all default traffic for the GEC bundle of VLAN subinterface member links.

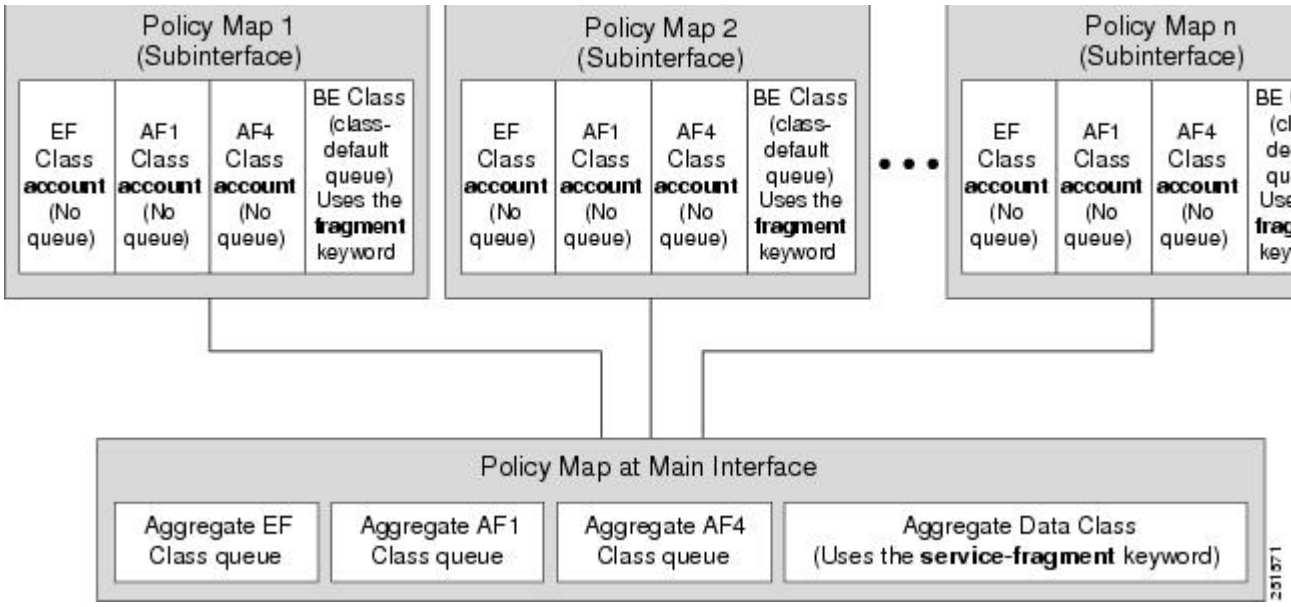
When fragments are configured for Gigabit Etherchannel bundles, the policy maps that have a default traffic class configured using the **fragment** keyword are attached to the member subinterface links, and the policy maps that have a traffic class configured with the service-fragment keyword to collectively classify the fragments is attached to the physical interface.

Understanding the QoS Policies Aggregation MQC

The QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature extends the previous support of aggregation of class-default traffic using the **fragment** and **service-fragment** configurations, to other user-defined traffic classes in a subinterface policy-map, such as DSCP-based traffic classes, that are aggregated at the main interface policy-map as shown in the figure below.

When no queueing is configured on a traffic class in the subinterface policy map, the **account** command can be used to track queueing drops that occur at the aggregate level for these classes, and can be displayed using the **show policy-map interface** command.

Figure 3 Policy Map Overview for the MQC Support for Multiple Queue Aggregation at Main Interface Feature



- Differences Between the Original Feature and the MQC Support for Multiple Queue Aggregation, page 24
- Changes in Queue Limit and WRED Thresholds, page 25

Differences Between the Original Feature and the MQC Support for Multiple Queue Aggregation

Although some of the configuration between the original QoS policies aggregation feature and enhancements in the MQC Support for Multiple Queue Aggregation at Main Interface feature appears similar, there are some important differences in the queueing behavior and the internal data handling.

For example, both configurations share and require the use of the **fragment** keyword for the **class class-default** command in the subscriber policy-map, as well as configuration of the **service-fragment**keyword for a user-defined class in the main interface policy-map to achieve common policy treatment for aggregate traffic. However, the use of this configuration results in different behavior between the original and enhanced QoS policies aggregation implementation:

- In the original implementation using the fragment and service-fragment architecture, all default class traffic and any traffic for classes without defined queueing features at the subinterface goes to the class-default queue and is aggregated into a common user-defined queue and policy defined at the main policy-map. Subinterface traffic aggregation (for example, from multiple subscribers on the same physical interface) ultimately occurs only for a single class, which is the default class.
- In the enhanced implementation of the MQC Support for Multiple Queue Aggregation at Main Interface feature also using the fragment and service-fragment architecture, all default class traffic also goes to the class-default queue and is aggregated into a common user-defined queue and policy defined at the main policy-map. However, other classes, such as DSCP-based subscriber traffic

classes, are also supported for an aggregate policy. These traffic classes do not support any queues or queueing features other than **account** at the subscriber policy-map. The use of the fragment and service-fragment architecture enables these other subscriber traffic classes (from multiple subscribers on the same physical interface) to achieve common policy treatment for aggregate traffic that is defined for those same classes at the main policy-map.

The following sections summarize the key behavioral differences between the original QoS: Policies Aggregation feature and the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature.

Qos: Policies Aggregation Feature Prior to Cisco IOS XE Release 2.6

- All subinterface traffic classes have queues. However, when a traffic class in the subinterface policy-map is not configured with any queueing feature (commands such as **priority**, **shape**, **bandwidth**, **queue-limit**, **fair-queue**, **random-detect**, and so on, are not configured), the traffic is assigned to the class-default queue.
- Default class traffic from multiple subinterfaces can be aggregated into a common policy-map at the main interface when you use the **fragment** keyword at the subinterface **class class-default** configuration, and **service-fragment** configuration at the main interface class.
- No classification occurs or is supported at the main interface policy-map for any subinterface traffic classes that do not use the **fragment** and **service-fragment** configuration.
- Queueing occurs at the subinterface for other traffic classes defined with queueing features in the subinterface policy-map.

QoS: Policies Aggregation - MQC Support for Multiple Queue Aggregation at Main Interface Feature Beginning in Cisco IOS XE Release 2.6

- Subinterface traffic classes without configured queueing features do not have queues at the subscriber level.
- Default class traffic from multiple subinterfaces can be aggregated into a common policy-map at the main interface when you use the **fragment** keyword at the subinterface **class class-default** configuration, and **service-fragment** configuration at the main interface class. This configuration additionally enables support for other subinterface traffic classes (such as DSCP-based classes) to be aggregated into a common policy-map at the main interface.
- Other class traffic from multiple subinterfaces can be aggregated into a common policy-map at the main interface, according to the following configuration requirements:
- You enable this behavior by using the **fragment** keyword at the subinterface **class class-default** configuration, and **service-fragment** configuration at the main interface class (this also enables aggregation of the default class).
- You do not configure any queueing features at the subinterface policy-map for the other traffic classes.
- Queueing occurs at the main interface policy-map for other subinterface traffic classes as an aggregate.
- Optional tracking of statistics is supported using the **account** command for other traffic classes in the subinterface policy-map.

Changes in Queue Limit and WRED Thresholds

In Cisco IOS XE Release 2.6 the Cisco ASR 1000 Series Aggregation Services Routers support the addition of bytes as a unit of configuration for both queue limits and WRED thresholds. Therefore, as of this release, packet-based and byte-based limits are configurable, with some restrictions.

How to Configure QoS Policies Aggregation

Configuring QoS Policies Aggregation for an Interface

Configuring a Fragment Traffic Class in a Policy Map

This procedure only shows how to configure the default traffic class as a fragment within a policy map. It does not include steps on configuring other classes within the policy map, or other policy maps on the router.

Like any policy map, the configuration is not managing network traffic until it has been attached to an interface. This procedure does not cover the process of attaching a policy map to an interface.

Note the following points about attaching and removing a policy map:

- To configure QoS: Policies Aggregation, you must attach the policy map that contains the **service-fragment** keyword to the main interface first, and then you must attach the policy map that contains the **fragment** keyword to the subinterface.
- To disable QoS: Policies Aggregation, you must remove the policy map that contains the **fragment** keyword from the subinterface first, and then you must remove the policy map that contains the **service-fragmen**t keyword from the main interface.

Note Only the default class statement in a policy map can be configured as a fragment.

Fragments only work when multiple policy maps are attached to the same physical interface. This process cannot be used to classify default traffic classes as fragments on policy maps on different physical interfaces.

Only queueing features are allowed in classes where the **fragment** keyword is entered, and at least one queueing feature must be entered in classes where the **fragment** keyword is used.

A policy map with a class using the **fragment** keyword can only be applied to traffic leaving the interface (policy maps attached to interfaces using the **service-policy output** command).

The **fragment** keyword cannot be entered in a child policy map.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class class-default fragment** *fragment-class-name*
5. *qos-queueing-feature*

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 subscriber1`	Specifies the name of the traffic policy to configure and enters policy map configuration mode.

Step 4	**class class-default fragment** *fragment-class-name*
	
	Example:
	`Router(config-pmap)#`
	`class class-default fragment BestEffort`
		Specifies the default traffic class as a fragment, and names the fragment traffic class.

Step 5	*qos-queueing-feature*	Enters a QoS configuration command. Only queueing features are supported in default traffic classes configured as fragments. The queueing features that are currently supported are **bandwidth**, **shape**, and **random-detect exponential-weighting-constant**. Multiple QoS queueing commands can be entered.

Example

Releases Prior to Cisco IOS XE Release 2.6

Cisco IOS XE Release 2.6 and Later Releases

In the following example, a fragment named BestEffort is created in policy map subscriber1 and policy map subscriber 2. In this example, queueing features for other traffic classes are supported at the subinterface policy map.

```
policy-map subscriber1
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map subscriber 2
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
```

The following example also shows how to configure a fragment named BestEffort for the default class in a policy map on a subinterface using the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface implementation. In this example, notice that queueing features are not supported for the other classes in the policy map:

```
policy-map subscriber1
 class voice
  set cos 5
  account
 class video
  set cos 4
  account
 class AF1
  account
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
```

- What to Do Next, page 28

What to Do Next

After configuring default class statements as fragments in multiple subinterface policy maps, a separate policy map with a class statement using the **service-fragment** keyword must be configured to apply QoS to the class statements configured as fragments.

This process is documented in the Configuring a Service Fragment Traffic Class, page 28.

Configuring a Service Fragment Traffic Class

This procedure assumes that fragment default traffic classes were already created. The procedure for creating fragment default traffic classes is documented in the Configuring a Fragment Traffic Class in a Policy Map, page 26.

Like any policy map, the configuration is not managing network traffic until it has been attached to an interface. This procedure does not cover the process of attaching a policy map to an interface.

Note A service fragment can be used to collectively classify fragments only from the same physical interface. Fragments from different interfaces cannot be classified using the same service fragment.

Only queueing features are allowed in classes where the **service-fragment** keyword is entered, and at least one queueing feature must be entered in classes when the **service-fragment** keyword is used.

A policy map with a class using the **service-fragment** keyword can only be applied to traffic leaving the interface (policy maps attached to interfaces using the **service-policy output** command).

A class configured using the **service-fragment** keyword cannot be removed when it is being used to collectively apply QoS to fragments that are still configured on the interface. If you wish to remove a class configured using the **service-fragment** keyword, remove the fragment traffic classes before removing the service fragment.

The **service-fragment** keyword cannot be entered in a child policy map.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class** *class-name* **service-fragment** *fragment-class-name*
5. *qos-queueing-feature*

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.

Command or Action	Purpose
Step 3 **policy-map** *policy-map-name* **Example:** `Router(config)# policy-map BestEffortFragments`	Specifies the name of the traffic policy to configure and enters policy map configuration mode.
Step 4 **class** *class-name* **service-fragment** *fragment-class-name* **Example:** `Router(config-pmap)# class data service-fragment BestEffort`	Specifies a class of traffic that is the composite of all fragments matching the *fragment-class-name*. The *fragment-class-name* when defining the fragments in other policy maps must match the *fragment-class-name* in this command line to properly configure the service fragment class.
Step 5 *qos-queueing-feature*	Enters a QoS configuration command. Only queueing features are supported in default traffic classes configured as fragments. The queueing features that are currently supported are **bandwidth**, **shape**, and **random-detect exponential-weighting-constant**. Multiple QoS queueing commands can be entered.

Examples

Releases Prior to Cisco IOS XE Release 2.6

Cisco IOS XE Release 2.6 and Later Releases

In the following example, a policy map is created to apply QoS to all fragments named BestEffort.

```
policy-map main-interface
class data service-fragment BestEffort
shape average 400000000
```

In the following example, two fragments are created and then classified collectively using a service fragment.

```
policy-map subscriber1
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map subscriber 2
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
```

```
  shape average 200000000
  bandwidth remaining ratio 10
```

The following example shows the creation of two fragments called BestEffort in the subinterface policy maps, followed by a sample configuration for the **service-fragment** called BestEffort to aggregate the queues at the main interface policy map:

```
policy-map subscriber1
 class voice
  set cos 5
  account
 class video
  set cos 4
  account
 class AF1
  account
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map subscriber2
 class voice
  set cos 5
  account
 class video
  set cos 4
  account
 class AF1
  account
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map main-interface
 class voice
  priority level 1
 class video
  priority level 2
 class AF1
  bandwidth remaining ratio 90
 class data service-fragment BestEffort
  shape average 400000000
  bandwidth remaining ratio 1
```


- Troubleshooting Tips, page 31
- What to Do Next, page 31

Troubleshooting Tips

Ensure that all class statements that are supposed to be part of the same service fragment share the same *fragment-class-name*.

What to Do Next

The policy map must be attached to an interface.

Configuring QoS Policies Aggregation on Gigabit Etherchannels

To properly configure QoS: Policies Aggregation on a Gigabit Etherchannel bundle, the following actions must be completed:

- Service fragment traffic classes must be configured and attached to the main physical interfaces.
- Fragment traffic classes must be configured and attached to the member link subinterfaces.

- Configuring Service Fragments on Physical Interface Supporting a Gigabit Etherchannel Bundle, page 32

- Configuring Fragments on Gigabit Etherchannel Member Link Subinterfaces, page 34

Configuring Service Fragments on Physical Interface Supporting a Gigabit Etherchannel Bundle

This procedure assumes that a service fragment traffic class has already been created. A service fragment traffic class cannot be configured without configuring a fragment class. The procedure for creating a fragment class is documented in the Configuring a Fragment Traffic Class in a Policy Map, page 26. The procedure for creating a service fragment traffic classes is documented in the Configuring a Service Fragment Traffic Class, page 28.

These instructions do not provide any details about the options that can be configured for Gigabit Etherchannel member link subinterfaces. These instructions only document the procedure for attaching a policy map that already has a fragment traffic class to a member link subinterface.

Note This process works only if all of the links of the GEC bundle are on the same physical interface.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet** *interface-number*
4. **service-policy output** *service-fragment-class-name*

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 gigabitethernet** *interface-number* **Example:** `Router(config)# interface gigabitethernet 1/1/1`	Enters Gigabit Ethernet interface mode.

Command or Action	Purpose
Step 4 **service-policy output** *service-fragment-class-name* **Example:** `Router(config-subif)# service-policy output aggregate-member-link`	Attaches a service policy that contains a service fragment default traffic class to the physical Gigabit Ethernet interface.

Examples

Note This example shows a sample configuration that is supported for the original QoS: Policies Aggregation feature in releases prior to Cisco IOS XE Release 2.6. By following the newer policy-map configuration guidelines for the updates in Cisco IOS XE Release 2.6, it can be adapted to the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature.

In the following example, policy map subscriber is configured with a fragment class named BE. The fragment is then configured as part of a policy map named aggregate-member-link. Policy map subscriber is then attached to the bundle subinterfaces while policy map aggregate-member-link is attached to the physical interface.

```
port-channel load-balancing vlan-manual
class-map match any data
!
class-map match-all BestEffort
!
class-map match-all video
!
class-map match-all voice
!
policy-map subscriber
 class voice
  priority level 1
 class video
  priority level 2
 class class-default fragment BE
  shape average 100000000
  bandwidth remaining ratios 80
policy-map aggregate-member-link
 class BestEffort service-fragment BE
  shape average 100000000
!
interface Port-channel1
 ip address 10.0.0.0 255.255.0.0
!
interface Port-channel1.100
 encapsulation dot1Q 100
 ip address 10.0.0.1 255.255.255.0
 service-policy output subscriber
!
interface Port-channel1.200
 encapsulation dot1Q 200
 ip address 10.0.0.2 255.255.255.0
 service-policy output subscriber
!
interface Port-channel1.300
 encapsulation dot1Q 300
 ip address 10.0.0.4 255.255.255.0
 service-policy output subscriber
!
interface GigabitEthernet1/1/1
```

```
 no ip address
 channel-group 1 mode on
 service-policy output aggregate-member-link
!
interface GigabitEthernet1/1/2
 no ip address
 channel-group 1 mode on
 service-policy output aggregate-member-link
```


- Troubleshooting Tips, page 34
- What to Do Next, page 34

Troubleshooting Tips

Ensure that the *fragment-class-name* is consistent across service-fragment and fragment class definitions.

What to Do Next

Attach the fragment service policy on the Gigabit Etherchannel member link subinterfaces.

Configuring Fragments on Gigabit Etherchannel Member Link Subinterfaces

This procedure assumes that a service fragment traffic class has already been created. A service fragment traffic class cannot be configured without configuring a fragment class. The procedure for creating a fragment class is documented in the Configuring a Fragment Traffic Class in a Policy Map, page 26. The procedure for creating a service fragment traffic classes is documented in the Configuring a Service Fragment Traffic Class, page 28.

These instructions do not provide any details about the options that can be configured for Gigabit Etherchannel member link subinterfaces. These instructions only document the procedure for attaching a policy map that already has a fragment traffic class to a member link subinterface.

Note Fragments cannot be used for traffic on two or more physical interfaces. The GEC must all be on the same physical interface for this configuration to work properly.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *port-channel-interface-number.port-channel- subinterface-number*
4. **service-policy output** *fragment-class-name*

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 port-channel** *port-channel-interface-number.port-channel- subinterface-number* **Example:** `Router(config)# interface port-channel 1.100`	Enters subinterface configuration mode to configure a Etherchannel member link subinterface.
Step 4	**service-policy output** *fragment-class-name* **Example:** `Router(config-subif)# service-policy output subscriber`	Attaches a service policy that contains a fragment default traffic class to the Etherchannel member link subinterface.

Example

Note This example shows a sample configuration that is supported for the original QoS: Policies Aggregation feature in releases prior to Cisco IOS XE Release 2.6. By following the newer policy-map configuration guidelines for the updates in Cisco IOS XE Release 2.6, it can be adapted to the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature.

In the following example, the service policy named subscriber has a fragment default traffic class and is attached to the member link subinterface of a Gigabit Etherchannel bundle.

Note This example only shows how to attach a fragment default traffic class to the member link subinterface of a Gigabit Etherchannel bundle. This configuration is incomplete and would not classify default traffic appropriately until the physical interface was configured to support a service fragment traffic class.

```
policy-map subscriber
 class voice
  priority level 1
 class video
  priority level 2
```

```
 class class-default fragment BE
  shape average 100000000
  bandwidth remaining ratios 80
policy-map aggregate-member-link
 class BestEffort service-fragment BE
  shape average 100000000
!
interface Port-channel1
 ip address 172.16.2.3 255.255.0.0
!
interface Port-channel1.100
 encapsulation dot1Q 100
 ip address 192.168.2.100 255.255.255.0
 service-policy output subscriber
!
```


- Troubleshooting Tips, page 36
- What to Do Next, page 36

Troubleshooting Tips

This configuration will not work until a service fragment default traffic class is created to classify the default traffic classes marked as fragments. This service fragment traffic class must be configured for this configuration to have any affect on network traffic.

What to Do Next

This is the final configuration step for configuring the QoS: Policies Aggregation feature on a Gigabit Etherchannel (GEC) bundle.

How to Configure QoS Policies Aggregation MQC

Some backward-compatibility exists between support of policies aggregation feature configuration in Cisco IOS XE Release 2.6 and prior Cisco IOS XE software releases. However, we recommend that you follow these upgrade guidelines for any physical interface where you want to move to the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature configuration.

For best results, you should upgrade any service policies configuration that you implemented prior to Cisco IOS XE Release 2.6, to the latest supported configuration.

The original and enhanced QoS: Policies Aggregation feature configuration can only reside on the same Cisco ASR 1000 Series Aggregation Services Router if the mixed configuration does not reside on the same physical interface. In other words, you can support the original configuration for one physical interface, and the enhanced configuration on a different physical interface.

The QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature requires the same configuration of a fragment traffic class as the original feature, using the **class class-default fragment** command to enable and then define all subinterface policies aggregation, both for the default traffic class and the other traffic classes.

In the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature, the queueing features for the aggregate class queues (with traffic from the corresponding classes identified at the subinterfaces), are configured at the main interface policy-map.

- Upgrading Your Service Policies fo QoS Policies Aggregation - MQC, page 37
- Configuring QoS Policies Aggregation MQC Traffic Classes, page 38
- Configuring QoS Policies Aggregation MQC Support, page 41
- Verifying the Traffic Policy Class Policy Information and Drop Statistics, page 41

Upgrading Your Service Policies fo QoS Policies Aggregation - MQC

Prerequisites

Upgrading your service policies to support the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature assumes the following network conditions:

- The corresponding class-map statements appropriate for your network traffic are already configured.
- QoS service policies aggregation has been previously configured and applied for the main interface policy-map for a given physical interface and its corresponding subinterfaces, or subscriber interfaces, prior to Cisco IOS XE Release 2.6 for the default traffic class.
- A port on the same physical interface where you have previously configured the service policies aggregation feature prior to Cisco IOS XE Release 2.6 needs to support the configuration for the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface.

Upgrade Tasks

SUMMARY STEPS

1. Configure the service policies for the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature.
2. Remove any service policies configured prior to Cisco IOS XE Release 2.6 for any prior configured policies aggregation features using the **no service-policy** and **no policy-map** commands as follows:
3. Apply the new service policies for the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature at the appropriate interfaces using the **service-policy output** command as follows:

DETAILED STEPS

Step 1 Configure the service policies for the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature.
See the tasks described in the Configuring QoS Policies Aggregation MQC Traffic Classes, page 38.

Step 2 Remove any service policies configured prior to Cisco IOS XE Release 2.6 for any prior configured policies aggregation features using the **no service-policy** and **no policy-map** commands as follows:

a) At each of the subinterfaces, configure the **no service-policy** command. Be sure to remove the policies at the subinterfaces first.

b) At the physical interface, configure the **no service-policy**command.

Step 3 Apply the new service policies for the QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature at the appropriate interfaces using the **service-policy output** command as follows:

a) At the physical interface, configure the **service-policy output** command.

b) At each of the subinterfaces, configure the **service-policy output** command.

Configuring QoS Policies Aggregation MQC Traffic Classes

Configuring Traffic Classes on the Subscriber Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class** *class-name*
5. **account** [**drop**]

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 subscriber1`	Specifies the name of the traffic policy to configure and enters policy map configuration mode.

	Command or Action	Purpose
Step 4	**class** *class-name* **Example:** `Router(config-pmap)# class EF`	Specifies the name of the traffic class to be aggregated at the main interface policy-map, and enters policy-map class configuration mode. **Note** Do not configure any queueing features for this class. Queueing is configured and aggregated at the main interface policy-map for all subinterfaces associated with this class and physical interface.
Step 5	**account** [**drop**] **Example:** `Router(config-pmap-c)# account`	(Optional) Enables collection of statistics for packets matching the traffic class where this command is configured, where the **drop** keyword collects all packet drop statistics. Collection of drop statistics is the default.

Example

The following example configures the EF traffic class for policies aggregation at the subscriber subinterface with collection of drop statistics:

```
policy-map subscriber1
 class EF
 account
```

- What to Do Next, page 39

What to Do Next

Follow this procedure for all traffic classes that you want to aggregate. Then, follow the instructions in the Configuring the Fragment Traffic Class on a Subinterface, page 39.

Configuring the Fragment Traffic Class on a Subinterface

- What to Do Next, page 39

What to Do Next

If you are upgrading your subinterface policy-map configuration from an earlier implementation of the QoS: Policies Aggregation feature, then remove the current service-policy from the subinterface using the **no service-policy** command.

Apply the new policy-map to outbound traffic on the subinterface using the **service-policy output** command.

Configuring Traffic Classes at the Main Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class** *class-name*
5. *qos-queueing-feature*

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 main-interface`	Specifies the name of the traffic policy to configure and enters policy map configuration mode.

Step 4	**class** *class-name*
	
	Example:
	`Router(config-pmap)#`
	`class EF`
		Specifies the name of the traffic class to be aggregated at the main interface policy-map, and enters policy-map class configuration mode.

Step 5	*qos-queueing-feature* **Example:** **Example:** `Router(config-pmap-c)# priority level 1`	Enters a QoS configuration command. The queueing features that are currently supported are **bandwidth**, **priority**, **shape**, and **random-detect exponential-weighting-constant**. Multiple QoS queueing commands can be entered.

Example

The following example configures three traffic classes at the main interface policy-map, along with the aggregate service-fragment data class:

```
policy-map main-interface
 class voice
  priority level 1
 class video
  priority level 2
 class AF1
  bandwidth remaining ratio 90
 class data service-fragment BestEffort
  shape average 400000000
  bandwidth remaining ratio 1
```

- What to Do Next, page 41

What to Do Next

Follow this procedure to define queueing features for all traffic classes that you want to aggregate. Then, follow the instructions in the Configuring the Service Fragment Traffic Class at the Main Interface, page 41.

Configuring the Service Fragment Traffic Class at the Main Interface

- What to Do Next, page 41

What to Do Next

If you are upgrading your main interface policy-map configuration from an earlier implementation of the QoS: Policies Aggregation feature, then remove the current service policy from the main interface using the **no service-policy** command.

Apply the new policy-map to outbound traffic on the main interface using the **service-policy output** command.

Configuring QoS Policies Aggregation MQC Support

The QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature also supports configuration of the enhanced service policies on Gigabit Etherchannels according to the subscriber and main interface configuration guidelines described for this enhancement.

For more information, see the following sections:

Verifying the Traffic Policy Class Policy Information and Drop Statistics

To display information about policy-map configuration and subscriber drop statistics enabled using the account command, use the **show policy-map interface** command:

```
Router# show policy-map interface port-channel 1.1
Port-channel1.1
   Service-policy input: input_policy
     Class-map: class-default (match-any)
       0 packets, 0 bytes
       5 minute offered rate 0000 bps, drop rate 0000 bps
       Match: any
       QoS Set
```

```
    dscp default
    No packet marking statistics available
Service-policy output: Port-channel_1_subscriber
  Class-map: EF (match-any)
    105233 packets, 6734912 bytes
    5 minute offered rate 134000 bps, drop rate 0000 bps
    Match: dscp ef (46)
    Match: access-group name VLAN_REMARK_EF
    Match: qos-group 3
    Account QoS statistics
      Queueing
        Packets dropped 0 packets/0 bytes
    QoS Set
    cos 5
    No packet marking statistics available
    dscp ef
    No packet marking statistics available
  Class-map: AF4 (match-all)
    105234 packets, 6734976 bytes
    5 minute offered rate 134000 bps, drop rate 0000 bps
    Match: dscp cs4 (32)
    Account QoS statistics
      Queueing
        Packets dropped 0 packets/0 bytes
    QoS Set
    cos 4
    No packet marking statistics available
  Class-map: AF1 (match-any)
    315690 packets, 20204160 bytes
    5 minute offered rate 402000 bps, drop rate 0000 bps
    Match: dscp cs1 (8)
    Match: dscp af11 (10)
    Match: dscp af12 (12)
    Account QoS statistics
      Queueing
        Packets dropped 0 packets/0 bytes
    QoS Set
    cos 1
    No packet marking statistics available
  Class-map: class-default (match-any) fragment Port-channel_BE
    315677 packets, 20203328 bytes
    5 minute offered rate 402000 bps, drop rate 0000 bps
    Match: any
    Queueing
      queue limit 31250 bytes
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 315679/20203482
      bandwidth remaining ratio 1
```

Configuration Examples for QoS Policies Aggregation

Example QoS Policies Aggregation

Note This example shows a sample configuration that is supported in the original QoS: Policies Aggregation feature prior to Cisco IOS XE Release 2.6.

In the following example, QoS: Policies Aggregation is used to define a fragment class of traffic to classify default traffic using the default traffic class named BestEffort. All default traffic from the policy maps

named subscriber1 and subscriber2 is part of the fragment default traffic class named BestEffort. This default traffic is then shaped collectively by creating a class called data that uses the **service-fragment** keyword and the **shape** command.

Note the following about this example:

- The *class-name* for each fragment default traffic class is "BestEffort."
- The *class-name* of "BestEffort" is also used to define the class where the **service-fragment** keyword is entered. This class applies a shaping policy to all traffic forwarded using the fragment default traffic classes named "BestEffort."

```
policy-map subscriber1
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map subscriber 2
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map input_policy
 class class-default
  set dscp default
policy-map main-interface
 class data service-fragment BestEffort
  shape average 400000000
interface portchannel1.1001
 encapsulation dot1q 1001
 service-policy output subscriber1
 service-policy input input_policy
interface portchannel1.1002
 encapsulation dot1q 1002
 service-policy output subscriber2
 service-policy input input_policy
interface gigabitethernet 0/1
 description member-link1
 port channel 1
 service-policy output main-interface
interface gigabitethernet 0/2
 description member-link2
 port channel 1
```

service-policy output main-interface

Example Gigabit Etherchannel QoS Policies Aggregation

Note This example shows a sample configuration that is supported in the original QoS: Policies Aggregation feature prior to Cisco IOS XE Release 2.6.

In the following example, policy map subscriber is configured with a fragment class named BE. The fragment is then configured as part of a policy map named aggregate-member-link. Policy map subscriber

is then attached to the bundle subinterfaces while policy map aggregate-member-link is attached to the physical interface.

```
port-channel load-balancing vlan-manual
class-map match-all BestEffort
!
class-map match-all video
!
class-map match-all voice
!
policy-map subscriber
 class voice
  priority level 1
 class video
  priority level 2
 class class-default fragment BE
  shape average 100000000
  bandwidth remaining ratios 80
policy-map aggregate-member-link
 class BestEffort service-fragment BE
  shape average 100000000
!
interface Port-channel1
 ip address 10.1.1.3 255.255.0.0
!
interface Port-channel1.100
 encapsulation dot1Q 100
 ip address 10.1.2.1 255.255.255.0
 service-policy output subscriber
!
interface Port-channel1.200
 encapsulation dot1Q 200
 ip address 10.1.2.2 255.255.255.0
 service-policy output subscriber
!
interface Port-channel1.300
 encapsulation dot1Q 300
 ip address 10.1.2.3 255.255.255.0
 service-policy output subscriber
!
interface GigabitEthernet1/1/1
 no ip address
 channel-group 1 mode on
 service-policy output aggregate-member-link
!
interface GigabitEthernet1/1/2
 no ip address
 channel-group 1 mode on
 service-policy output aggregate-member-link
```

Example QoS Policies Aggregation MQC Support at Main Interface

Note This example shows a sample configuration that is supported beginning in Cisco IOS XE Release 2.6.

At the main interface policy map called Port-channel_1_main_policy, the queueing features for the DSCP-based subscriber traffic classes are configured. You can also see the use of byte-based queue limits and random-detect thresholds implemented at the main interface queues.

The service-fragment called Port-channel_BE is also configured to aggregate the traffic from the subscriber class-default fragment class.

```
policy-map Port-channel_1_main_policy
 class EF
  priority level 1
  queue-limit 547500 bytes
```

```
 class AF4
  priority level 2
  queue-limit 4037500 bytes
 class AF1
  bandwidth remaining ratio 90
  queue-limit 750000 bytes
  random-detect dscp-based
  random-detect dscp 8 750000 bytes 750000 bytes
  random-detect dscp 10 750000 bytes 750000 bytes
  random-detect dscp 12 600000 bytes 675000 bytes
 class data service-fragment Port-channel_BE
  shape average 250000000
  bandwidth remaining ratio 1
!
```

In this example, the policy map Port-channel_1_subscriber is configured with a fragment class named Port-channel_BE. (For simplicity, only a single subinterface policy is shown.) This enable queueing and policies aggregation for the subscriber traffic classes at the main interface policy map.

The Port-channel_1_subscriber policy map identifies the DSCP-based traffic classes of EF, AF4, and AF1 and enables collection of drop statistics for those classes.

```
policy-map Port-channel_1_subscriber
 class EF
  account
  set cos 5
  set dscp ef
 class AF4
  account
  set cos 4
 class AF1
  account
  set cos 1
 class class-default fragment Port-channel_BE
  bandwidth remaining ratio 1
  queue-limit 31250 bytes
!
port-channel load-balancing vlan-manual
!
interface Port-channel1
 no ip address
 no negotiation auto
!
```

The service policies are applied first to the physical interface, and then to the subinterfaces as shown:

```
interface GigabitEthernet1/2/0
 no ip address
 negotiation auto
 no cdp enable
 service-policy output Port-channel_1_main_policy
 channel-group 1
!
interface GigabitEthernet2/2/0
 no ip address
 negotiation auto
 service-policy output Port-channel_1_main_policy
 channel-group 1
!
interface Port-channel1.1
 encapsulation dot1Q 2 primary GigabitEthernet1/2/0 secondary GigabitEthernet2/2/0
 ip address 10.0.0.2 255.255.255.0
 service-policy output Port-channel_1_subscriber
```

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	*Cisco IOS Quality of Service Solutions Command Reference*
Modular Quality of Service Command-Line Interface	"Applying QoS Features Using the MQC" module
Distribution of Remaining Bandwidth Using Ratio	"Distribution of Remaining Bandwidth Using Ratio" module
Class-Based Shaping	"Regulating Packet Flow-- Using Class-Based Traffic Shaping" module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
CISCO-CLASS-BASED-QOS-MIB	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported, and support for existing RFCs has not been modified.	--

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 QoS Policies Aggregation

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 4 Feature Information for QoS: Policies Aggregation

Feature Name	Releases	Feature Information
QoS: Policies Aggregation	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers. The following command was modified: **class (policy-map)**.

Feature Name	Releases	Feature Information
QoS: QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface	Cisco IOS XE Release 2.6	This feature was enhanced to support queueing aggregation at the primary interface for other traffic classes, including DSCP-based classes such as EF, AF1, and AF4 traffic classes. With this enhancement, other traffic classes from different subinterfaces share a common queue for that traffic class. Other enhancements include the ability to configure and show per-subscriber drop statistics on the aggregate queues and byte-based queue limits and WRED thresholds. In Cisco IOS XE Release 2.6, support for the CISCO-CLASS-BASED-QOS-MIB was added. The following commands are new or modified: **account**, **show policy-map interface**.

Legacy QoS Command Deprecation

In Cisco IOS XE Release 2.6, to streamline Cisco IOS XE quality of service (QoS), certain commands have been hidden. Although these commands are available, the command-line interface (CLI) interactive help does not display them. This means that if you attempt to view a hidden command by entering a question mark (**?**) at the command line, the command does not appear. However, if you know the command syntax, you can enter it (the system accepts the command and returns a message stating that it is deprecated).

The functionality provided by these hidden commands has been replaced by similar functionality provided via the modular QoS CLI (MQC). The MQC is a set of a platform-independent commands for configuring QoS on Cisco platforms. This means that you should now provision QoS by defining traffic classes, creating traffic policies containing those classes, and attaching those policies to the desired interfaces.

In Cisco IOS XE Release 3.2S, these commands have been removed. This means that you must use the appropriate replacement MQC commands.

This document lists the hidden or removed commands and their replacement commands.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see 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 document.

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.

Information About Legacy QoS Command Deprecation

QoS Features Applied Using the MQC

The MQC structure lets you define a traffic class (also called a class map), create a traffic policy (also called a policy map), and attach the traffic policy to an interface. This comprises 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. A traffic policy 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 to the interface by using the **service-policy** command.

Steps 1 and 3 do not involve legacy QoS hidden or removed commands, which means that they are not within the scope of this document. For more information about these two steps, see the " Applying QoS Features Using the MQC " module in the **Quality of Service Solutions Configuration Guide**.

Legacy Commands Being Hidden or Removed

The table below lists the commands that have been hidden or removed. The table also lists their replacement commands (or sequence of commands).

Table 5 Map of Hidden or Removed Commands to Their Replacement Commands

Hidden or Removed Commands	Replacement MQC Command Sequence
Configuring Bandwidth Allocation	

Commands
• max-reserved-bandwidth
Command Usage
`Router(config)# interface`
`type`
`number`
`Router(config-if)# max-reserved-bandwidth percentage`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# bandwidth {`*bandwidth-in-kbps* `| remaining percent` *percentage* `| percent` *percentage*`}`

Configuring Custom Queueing	

Hidden or Removed Commands	Replacement MQC Command Sequence

Commands
• custom-queue-list
Command Usage
`Router(config)# interface`
`type`
`number`
`Router(config-if)# custom-queue-list`
`[`
`list-number`
`]`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# bandwidth`
	{*bandwidth-in-kbps* | **remaining percent** *percentage* | **percent** *percentage*}

Configuring Priority Queueing	

Commands
• ip rtp priority
Command Usage
`Router(config)# interface`
`type`
`number`
`Router(config-if)# ip rtp priority`
`starting-port-number port-range bandwidth`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class`
	`class-name`
	`Router(config-pmap-c)# priority`

Configuring Weighted Fair Queueing	

Commands
• fair-queue (WFQ)
Command Usage
`Router(config)# interface`
`type`
`number`
`Router(config-if)# fair-queue`
[`congestive-`*discard-threshold* [**dynamic-queue-count** [*reserved-queue-count*]]]
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# fair-queue`
	`Router(config-pmap-c)# fair-queue`
	`dynamic-queues`
	`Router(config-pmap-c)# fair-queue queue-limit`
	`packets`

Configuring the Threshold for Discarding DE Packets from a Switched PVC Traffic Shaping Queue	

Hidden or Removed Commands	Replacement MQC Command Sequence

Commands
• frame-relay congestion threshold de
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config-map-class)# frame-relay congestion threshold de`
`percentage`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name1`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# random-detect discard-class-based`
	`Router(config-pmap-c)# random-detect discard-class`
	`discard-class`
	`min-threshold`
	`max-threshold`
	`Router(config-pmap-c)# exit`
	`Router(config-pmap)# exit`
	`Router(config)# policy-map shape`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# shape average`
	`rate`
	`Router(config-pmap-c)# service-policy`
	`policy-map-name1`
	`Router(config-pmap-c)# exit`
	`Router(config-pmap)# exit`
	`Router(config)# policy-map`
	`policy-map-name2`
	`Router(config-pmap)# class`
	`class-name`
	`Router(config-pmap-c)# set discard-class`
	`discard-class`

Configuring Frame Relay Custom Queueing for Virtual Circuits	

Commands
• frame-relay custom-queue-list
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config-map-class)# frame-relay custom-queue-list`
`list-number`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# bandwidth {`*bandwidth-in-kbps* `| remaining percent` *percentage* `| percent` *percentage*`}`

Configuring Frame Relay ECN Bits Threshold	

Commands
• frame-relay congestion threshold ecn
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config-map-class)# frame-relay congestion threshold ecn`
`percentage`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# shape average`
	`rate`
	`Router(config-pmap-c)# set fr-fecn-becn`
	`percent`

Hidden or Removed Commands	Replacement MQC Command Sequence
Configuring Frame Relay Weighted Fair Queueing	

Commands
• frame-relay fair-queue
Command Usage
`Router(config)# map-class frame-relay`
` map-class-name`
`Router(config-map-class)# frame-relay fair-queue`
` [`
`discard-threshold`
`[`
`dynamic-queue-count`
` [`
`reserved-queue-count`
` [`
`buffer-limit`
`]]]]`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# fair-queue`
	`Router(config-pmap-c)# fair-queue`
	` dynamic-queues`
	`Router(config-pmap-c)# fair-queue queue-limit`
	`packets`

Configuring Frame Relay Priority Queueing on a PVC	

Commands
• frame-relay ip rtp priority
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config-map-class)# frame-relay ip rtp priority`
`starting-port-number`
`port-range`
`bandwidth`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)#`
	`class`
	`class-name`
	`Router(config-pmap-c)#`
	`priority`
	`bandwidth-in-kbps`
	` [`
	`burst-in-bytes`
	`]`

Assigning a Priority Queue to Virtual Circuits Associated with a Map Class	

Hidden or Removed Commands	Replacement MQC Command Sequence

Commands
• frame-relay priority-group
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config-map-class)# frame-relay priority-group`
`group-number`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# priority`
	`Router(config-pmap-c)# priority`
	`bandwidth-in-kbps`
	`[`
	`burst-in-bytes`
	`]`
	`Router(config-pmap-c)# priority percent`
	`percentage`
	`[`
	`burst-in-bytes`
	`]`
	`Router(config-pmap-c)# priority level`
	`level`
	`[percent`
	`percentage`
	`[`
	`burst-in-bytes`
	`]]`

Configuring the Frame Relay Rate Adjustment to BECN	

Commands
• frame-relay adaptive-shaping (becn keyword)
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config-map-class)# frame-relay adaptive-shaping becn`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# shape average`
	`rate`
	`Router(config-pmap-c)# shape adaptive`
	`rate`

Configuring the Frame Relay Rate Adjustment to ForeSight Messages	

Commands
• frame-relay adaptive-shaping (foresight keyword)
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config)# frame-relay adaptive-shaping foresight`
	Command Usage
	None (this functionality no longer exists).

Enabling Frame Relay Traffic-Shaping FECNs as BECNs	

Hidden or Removed Commands	Replacement MQC Command Sequence

Commands
• frame-relay fecn-adapt
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config-map-class)# frame-relay fecn-adapt`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# shape average`
	rate`Router(config-pmap-c)# shape fecn-adapt`

Configuring Frame Relay Traffic Shaping	

Commands
• frame-relay bc
• frame-relay be
• frame-relay cir
Command Usage
`Router(config)# map-class frame-relay`
`map-class-name`
`Router(config-map-class)# frame-relay bc`
` {in`
`| out`
`}`
`committed-burst-size-in-bits`
`Router(config-map-class)# frame-relay be`
`{in`
`| out`
`} excess-`
`burst-size-in-bits`
`Router(config-map-class)# frame-relay cir`
`{in`
`| out`
`}`
`bits-per-second`
	Command Usage
	`Router(config)# policy-map`
	`policy-map-name`
	`Router(config-pmap)# class class-default`
	`Router(config-pmap-c)# shape average`
	`rate`

Configuring the Frame Relay Enhanced Local Management Interface	

Commands
• frame-relay qos-autosense
Command Usage
`Router(config)# interface` *type number*`Router(config-if)# no ip address`
`Router(config-if)# encapsulation frame-relay`
`Router(config-if)# frame-relay lmi-type`
` ansi`
`Router(config-if)# frame-relay traffic-shaping`
`Router(config-if)# frame-relay qos-autosense`
	Command Usage
	None (this functionality no longer exists).

Displaying the Contents of Packets Inside a Queue for an Interface or VC	

Hidden or Removed Commands	Replacement MQC Command Sequence
Commands • show queue **Command Usage** `Router# show queue interface`	**Command Usage** `Router# show policy-map interface`
Displaying Queueing Strategies	
Commands • show queueing **Command Usage** `Router# show queueing`	**Command Usage** `Router# show policy-map interface`
Displaying Weighted Random Early Detection (WRED) Information	
Commands • show interfaces random-detect **Command Usage** `Router# show interfaces [ type number ] random-detect`	**Command Usage** `Router# show policy-map interface`
Displaying the Traffic-Shaping Configuration, Queueing, and Statistics	

Hidden or Removed Commands	Replacement MQC Command Sequence

Commands
• show traffic-shape
• show traffic-shape queue
• show traffic-shape statistics
Command Usage
`Router# show traffic-shape [ interface-type interface-number ]`
`Router# show traffic-shape queue [ interface-number [dlci dlci-number ]]`
`Router# show traffic-shape statistics [ interface-type interface-number ]`
	Command Usage
	`Router# show policy-map interface`

Displaying Weighted Fair Queueing Information	
Commands • show interfaces fair-queue **Command Usage** `Router# show interfaces [ interface-type interface-number ] fair-queue`	**Command Usage** `Router# show policy-map interface`

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Defining traffic classes; attaching traffic policies to interfaces	" Applying QoS Features Using the MQC " module in the *Quality of Service Solutions Configuration Guide*

Related Topic	Document Title
Reference pages for QoS commands	*Cisco IOS Quality of Service Solutions Command Reference*
Reference pages for wide-area networking commands	*Cisco IOS Wide-Area Networking Command Reference*

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 Legacy QoS Command Deprecation

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 6 Feature Information for Legacy QoS Command Deprecation

Feature Name	Releases	Feature Information
Legacy QoS Command Deprecation: Hidden Commands	Cisco IOS XE Release 2.6	To streamline Cisco IOS XE QoS, certain commands have been hidden, which means that if you try to view a hidden command by entering a question mark (**?**) at the command line, the command does not appear. However, if you know the command syntax, you can enter it. These commands will be removed in a future release. The functionality provided by these hidden commands is replaced by similar functionality from the modular QoS CLI (MQC), which is a set of a platform-independent commands for configuring QoS. The following commands were modified: **custom-queue-list**, **fair-queue (WFQ)**, **frame-relay adaptive-shaping** (**becn** keyword), **frame-relay adaptive-shaping** (**foresight** keyword), **frame-relay bc**, **frame-relay be**, **frame-relay cir**, **frame-relay congestion threshold de**, **frame-relay congestion threshold ecn**, **frame-relay custom-queue-list**, **frame-relay fair-queue**, **frame-relay fecn-adapt**, **frame-relay ip rtp priority**, **frame-relay priority-group**, **frame-relay qos-autosense**, **ip rtp priority**, **max-reserved-bandwidth**, **show interfaces fair-queue**, **show interfaces random-detect**, **show queue**, **show queueing**, **show traffic-shape**, **show traffic-shape queue**, **show traffic-shape statistics**.

Feature Name	Releases	Feature Information
Legacy QoS Command Deprecation: Removed Commands	Cisco IOS XE Release 3.2S	The legacy QoS commands were removed. This means that you must use the appropriate replacement MQC commands. The following commands were removed: **custom-queue-list**, **fair-queue (WFQ)**, **frame-relay adaptive-shaping** (**becn** keyword), **frame-relay adaptive-shaping** (**foresight** keyword), **frame-relay bc**, **frame-relay be**, **frame-relay cir**, **frame-relay congestion threshold de**, **frame-relay congestion threshold ecn**, **frame-relay custom-queue-list**, **frame-relay fair-queue**, **frame-relay fecn-adapt**, **frame-relay ip rtp priority**, **frame-relay priority-group**, **frame-relay qos-autosense**, **ip rtp priority**, **max-reserved-bandwidth**, **show interfaces fair-queue**, **show interfaces random-detect**, **show queue**, **show queueing**, **show traffic-shape**, **show traffic-shape queue**, **show traffic-shape statistics**.