MQC-Based Frame Relay Traffic Shaping

First Published: November 25, 2002
Last Updated: June 11, 2009

The MQC-Based Frame Relay Traffic Shaping feature provides users with the ability to configure Frame Relay traffic shaping (FRTS) using Modular Quality of Service (QoS) Command Line Interface (CLI) commands. Modular QoS CLI is known as MQC.

This feature was already available for routers in the Cisco 7500 and above product range; MQC FRTS is now available for the Cisco routers specified in the "Supported Platforms" section.

Feature Specifications for MQC-Based Frame Relay Traffic Shaping

Feature History	
Release	**Modification**
12.2(13)T	This feature was introduced.
Supported Platforms	
Cisco 1700 series, Cisco 2500 series, Cisco 2600 series, Cisco 3620 router, Cisco 3631 router, Cisco 3640 router, Cisco 3660 router, Cisco 3725 router, Cisco 3745 router, Cisco 7200 series, Cisco 7400 series	

Determining Platform Support Through Cisco Feature Navigator

Cisco IOS software is packaged in feature sets that are supported on specific platforms. To get updated information regarding platform support for this feature, access Cisco Feature Navigator. Cisco Feature Navigator dynamically updates the list of supported platforms as new platform support is added for the feature.

Cisco Feature Navigator is a web-based tool that enables you to determine which Cisco IOS software images support a specific set of features and which features are supported in a specific Cisco IOS image. You can search by feature or release. Under the release section, you can compare releases side by side to display both the features unique to each software release and the features in common.

To access Cisco Feature Navigator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions found at this URL:

http://www.cisco.com/register

Cisco Feature Navigator is updated regularly when major Cisco IOS software releases and technology releases occur. For the most current information, go to the Cisco Feature Navigator home page at the following URL:

http://www.cisco.com/go/fn

Availability of Cisco IOS Software Images

Platform support for particular Cisco IOS software releases is dependent on the availability of the software images for those platforms. Software images for some platforms may be deferred, delayed, or changed without prior notice. For updated information about platform support and availability of software images for each Cisco IOS software release, refer to the online release notes or, if supported, Cisco Feature Navigator.

Contents

Prerequisites for MQC-Based Frame Relay Traffic Shaping

Every permanent virtual circuit (PVC) to which FRTS using MQC is applied must have Frame Relay encapsulation enabled on the associated interface with the **encapsulation frame-relay** command.

Note For FRTS using MQC for the routers specified in the "Supported Platforms" section, distributed Cisco Express Forwarding (dCEF) is not required. For FRTS using MQC for routers in the Cisco 7500 and above product range, dCEF is required.

Restrictions for MQC-Based Frame Relay Traffic Shaping

You must create a default class for the service policy as specified with the **class class-default** command to configure FRTS using MQC on a Frame Relay PVC. The default class will have all the FRTS parameters applied to it. Refer to the "Creating a Policy Map and Entering CBWFQ Parameters for the Class Map" section for more information on creating a default class structure for a service policy.

If FRTS and fragmentation are applied to a PVC using MQC, the interface queue will change to dual FIFO queueing. The two queues will consist of a high-priority queue to carry Voice over IP (VoIP) and certain control packets, and a low-priority queue to carry all other packets.

In configurations created by using traditional FRTS commands, the minimum acceptable outgoing committed information rate (minCIR) will be used as the total available bandwidth for a policy map that has class-based weighted fair queueing (CBWFQ) attached to the map class for the PVC.

If the MQC-Based Frame Relay Traffic Shaping feature is used to configure FRTS, the shaping rate that was configured in the parent policy map using MQC will be used as the total available bandwidth for the child policy map, if CBWFQ is configured. If both the **shape average** and **shape adaptive** commands are used for traffic-shaping, the available bandwidth will be based on the parameters specified by the **shape adaptive** command

Information About MQC-Based Frame Relay Traffic Shaping

To configure MQC-Based Frame Relay Traffic Shaping, you must understand the following concepts:

- MQC Overview, page 3
- FRTS Overview, page 3
- Feature Design of MQC-Based Frame Relay Traffic Shaping, page 4
- Supported MQC Frame Relay Traffic Shaping Commands, page 5
- Benefits of MQC-Based Frame Relay Traffic Shaping, page 5

MQC Overview

MQC is used to configure FRTS. MQC is a framework that provides a clear separation between a classification policy and the specification of other parameters that act on the results of that applied classification policy.

Previously, FRTS for the Cisco routers specified in the "Supported Platforms" section could be configured only by using traditional FRTS commands (refer to the "How to Configure MQC-Based Frame Relay Traffic Shaping" section for more information). With the addition of the MQC-Based Frame Relay Traffic Shaping feature, FRTS can be configured on routers throughout the Cisco router product line by using MQC.

Broadly, MQC is configured and implemented as follows:

- Define a traffic class with the **class-map** command.
- Create a service policy by associating the traffic class with one or more QoS features (using the **policy-map** command).
- Attach the service policy to the interface with the **service-policy** command.

For more detailed information on MQC, refer to the document "Modular Quality of Service Command-Line Interface." MQC commands used for FRTS are further explained in the "How to Configure MQC-Based Frame Relay Traffic Shaping" section of this document.

FRTS Overview

FRTS allows you to control the traffic going out through a PVC in order to match its flow to the speed of the remote target interface and to ensure that the traffic conforms to the parameters that have been set for it. Traffic that matches a particular profile can be shaped to meet downstream requirements, thereby eliminating the bottlenecks that occur in topologies that have data-rate mismatches.

The primary reasons you would use FRTS are the following:

- To allow high-priority packets to take precedence over other packets as they are encapsulated and forwarded over the Frame Relay network. FRTS is useful for applications such as VoIP and streaming video, which require a low latency to be effective.
- To control access to available bandwidth.
- To ensure that traffic conforms to the parameters established for it.
- To regulate the flow of traffic in order to avoid congestion that can occur when the sent traffic exceeds the access speed of its remote target interface.
- To eliminate bottlenecks in Frame Relay networks that have high-speed connections at the central site and low-speed connections at branch sites by configuring rate enforcement to limit the rate at which data is sent on the virtual circuit (VC) at the central site. Rate enforcement is a peak rate configured to limit outbound traffic.

Feature Design of MQC-Based Frame Relay Traffic Shaping

The MQC-Based Frame Relay Traffic Shaping feature allows the Cisco routers specified in the "Supported Platforms" section to have FRTS configured using MQC instead of traditional FRTS commands.

Before this feature was introduced, FRTS for the Cisco routers specified in the "Supported Platforms" section could be configured only by using traditional FRTS commands (for example, the **frame-relay traffic-shaping** command). For traditional FRTS, all traffic shaping and fragmentation values are entered under the map class. Traffic shaping is defined by entering the **map-class frame-relay** command, then entering the traffic shaping and, optionally, fragmentation values.

For routers in the Cisco 7500 and above product range, Distributed Traffic Shaping (DTS) is used for traffic shaping. With DTS, the traffic-shaping values are configured by entering the **policy-map** command, then entering the traffic-shaping values. However, fragmentation values are still entered under the map class—the **map-class frame-relay** command is still used before any fragmentation values are entered.

The traffic-shaping commands supported by the MQC-Based Frame Relay Traffic Shaping feature are listed in the "Supported MQC Frame Relay Traffic Shaping Commands" section.

CBWFQ can also be configured under the policy map by entering the **policy-map** command and then entering the traffic-shaping CBWFQ values.

Note Configuring traffic shaping using MQC and configuring traffic shaping using traditional FRTS commands are mutually exclusive. Traffic shaping cannot be configured on the same interface using both methods.

Supported MQC Frame Relay Traffic Shaping Commands

The following MQC traffic-shaping commands are supported by the MQC-Based Frame Relay Traffic Shaping feature:

- **shape** {**average** | **peak**}
- **shape adaptive**
- **shape fecn-adapt**
- **shape max-buffers**

The **frame-relay ip rtp priority** command is not supported.

Note Previously, FRTS for the Cisco routers specified in the "Supported Platforms" section could be configured only by using traditional FRTS commands (refer to the "How to Configure MQC-Based Frame Relay Traffic Shaping" section for more information).

Benefits of MQC-Based Frame Relay Traffic Shaping

MQC allows users to specify a traffic class independently of QoS parameters.

This feature allows users to apply FRTS parameters using MQC across the entire Cisco router product line.

The MQC-Based Frame Relay Traffic Shaping feature ensures that FRTS is defined in the same manner for routers across the Cisco router product line, rather than only for routers in the Cisco 7500 and above product range.

Previously, FRTS for the Cisco routers specified in the "Supported Platforms" section could only be defined by using traditional FRTS commands. Using different methods to define FRTS for different routers can introduce inconsistency and complexity when FRTS is being implemented on different router platforms.

How to Configure MQC-Based Frame Relay Traffic Shaping

This section contains the following procedures. Each procedure is identified as either required or optional.

- Creating a Class Map and Specifying Match Criteria for CBWFQ (optional)
- Creating a Policy Map and Entering CBWFQ Parameters for the Class Map (optional)
- Creating a Shaping Policy Map and Entering FRTS Parameters for the Default Class Map (required)
- Attaching the Class-Based Weighted Fair Queueing Policy Map to the Shaping Policy Map (optional)
- Specifying a Map Class and Attaching a Service Policy for the PVC (required)
- Configuring an Interface or Subinterface for Frame Relay and Associating a Map Class with a PVC (required)
- Defining Fragmentation Parameters for the PVC (optional)
- Adding the Policy Map to the Map Class (required)

Creating a Class Map and Specifying Match Criteria for CBWFQ

To create a class map and specify match criteria for CBWFQ, use the following commands:

SUMMARY STEPS

1. **class-map** *class-map-name*
2. **match** *match-criteria*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**class-map** *class-map-name* **Example:** `Router(config)# class-map voice`	Creates a class map to be used for matching packets to a specified class. • The example command creates a class map named "voice".
Step 2	**match** *match-criteria* **Example:** `Router(config-cmap)# match ip dscp ef`	Identifies packets that will belong to the class map. • The example command identifies an IP differentiated service code point (DSCP) value of EF (101110) as a match criterion for the class map named "voice".

Creating a Policy Map and Entering CBWFQ Parameters for the Class Map

To create a policy map and enter CBWFQ parameters for the class map, use the following commands:

SUMMARY STEPS

1. **policy-map** *policy-map-name*
2. **class** *name*
3. **priority** {*bandwidth-kbps* | **percent** *percentage*} [*burst*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	`policy-map` *policy-map-name* **Example:** `Router(config)# policy-map llq`	Creates a policy map that can be attached to one or more interfaces to specify a service policy. • The example command creates a policy map named "llq". This policy map will be used for low latency queueing (LLQ).
Step 2	`class` *name* **Example:** `Router(config-pmap)# class voice`	Specifies the name of the class whose policy you want to create. • The example command creates a traffic class named "voice".
Step 3	`priority` {*bandwidth-kbps* | `percent` *percentage*} [*burst*] **Example:** `Router(config-pmap-c)# priority 32`	(Optional) Gives priority to a class of traffic belonging to a policy map. • The example command provides a guaranteed allowed bandwidth of 32 kbps and a guaranteed low latency for up to 32 kbps to the traffic class "voice" that was created in Step 2.

Creating a Shaping Policy Map and Entering FRTS Parameters for the Default Class Map

To create a shaping policy map and enter FRTS values for the default class map, use the following commands:

SUMMARY STEPS

1. **policy-map** *policy-map-name*
2. **class class-default**
3. **shape** [**average** | **peak**] *mean-rate* [[*burst-size*] [*excess-burst-size*]]
4. **shape adaptive** *mean-rate-lower-bound*
5. **shape fecn-adapt**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**policy-map** *policy-map-name* **Example:** `Router(config-pmap-c)# policy-map shape_policy_map`	Creates a policy map that can be attached to one or more interfaces to specify a service policy. • The example command creates a policy map named shape_policy_map.
Step 2	**class class-default** **Example:** `Router(config-pmap)# class class-default`	Specifies the default class (commonly known as the class-default class) before you configure its policy. The class-default class is the class to which traffic is directed if that traffic does not satisfy the match criteria of other classes whose policy is defined in the policy map. • There is only one class defined with this command (the class-default class); therefore, it will match all traffic.
Step 3	**shape** [**average** | **peak**] *mean-rate* [[*burst-size*] [*excess-burst-size*]] **Example:** `Router(config-pmap-c)# shape average 64000`	Shapes traffic to the indicated bit rate according to the algorithm specified. • The example command configures a shape entity with a committed information rate (CIR) of 64 kbps. • Any FRTS commands supported by this feature can be used for traffic shaping. Refer to the "Restrictions for MQC-Based Frame Relay Traffic Shaping" section for a list of supported FRTS commands.
Step 4	**shape adaptive** *mean-rate-lower-bound* **Example:** `Router(config-pmap-c)# shape adaptive 32000`	(Optional) Configures a Frame Relay PVC to estimate the available bandwidth by backward explicit congestion notification (BECN) integration while traffic shaping is enabled. • The example command sets the lower bound CIR to 32 kbps when BECNs are received.
Step 5	**shape fecn-adapt** **Example:** `Router(config-pmap-c)# shape fecn-adapt`	(Optional) Configure a Frame Relay interface to reflect received forward explicit congestion notification (FECN) bits as BECN bits in Q.922 "Test Response" messages. • The example command adapts the Frame Relay message with FECN to BECN.

Attaching the Class-Based Weighted Fair Queueing Policy Map to the Shaping Policy Map

To attach the CBWFQ policy map to the shaping policy map, use the following commands:

SUMMARY STEPS

1. **service-policy** *policy-map-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	`service-policy` *policy-map-name* **Example:** `Router(config-pmap-c)# service-policy llq`	Attaches a service policy as a traffic shaping policy within a policy map. This parent-child service policy association is called a hierarchical service policy. • The example command creates a hierarchical service policy with the parent policy map being "shape", and the child policy map being "llq".

Specifying a Map Class and Attaching a Service Policy for the PVC

To specify a map class and attach a service policy for the PVC, use the following commands:

SUMMARY STEPS

1. **map-class frame-relay** *map-class-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**map-class frame-relay** *map-class-name* **Example:** `Router(config)# map-class frame-relay shape_map_class`	Specifies a map class to define fragmentation parameters for both a PVC and a MQC policy map attachment. • The example command specifies a map class named shape_map_class.

Configuring an Interface or Subinterface for Frame Relay and Associating a Map Class with a PVC

To configure an interface or subinterface for Frame Relay and associate a map class with a PVC, use the following commands:

SUMMARY STEPS

1. **interface** *type number* [*name-tag*]
2. **encapsulation frame-relay** [**cisco** | **ietf**]
3. **interface** *type slot*/*port*.*subinterface-number* [**multipoint** | **point-to-point**]
4. **ip address** *ip-address mask* [**secondary**]
5. **frame-relay interface-dlci** *dlci* [**ietf** | **cisco**]
6. **class** *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface serial0/0`	Configures an interface type and enters interface configuration mode.
Step 2	**encapsulation frame-relay** [**cisco** | **ietf**] **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
Step 3	**interface** *type slot***/***port***.***subinterface-number* [**multipoint** | **point-to-point**] **Example:** `Router(config-if)# interface serial0/0.1 point-to-point`	Configures a subinterface.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-subif)# ip address 192.168.1.1 255.255.255.0`	Sets the primary IP address and mask for the subinterface.
Step 5	**frame-relay interface-dlci** *dlci* [**ietf** | **cisco**] **Example:** `Router(config-subif)# frame-relay interface-dlci 100`	Assigns a data-link connection identifier (DLCI) to a specified Frame Relay subinterface on the router. • The example creates a PVC with a DLCI number of 100 under subinterface serial0/0.1.
Step 6	**class** *name* **Example:** `Router(config-fr-dlci)# class shape_map_class`	Associates a map class with the subinterface. • The example command associates the map class named shape_map_class with the PVC created in Step 5.

Defining Fragmentation Parameters for the PVC

To define fragmentation parameters for the PVC, use the following commands.

SUMMARY STEPS

1. **frame-relay fragment** *fragment_size* **[switched]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`frame-relay fragment`** *`fragment_size`* [**`switched`**] **Example:** `Router(config-map-class)# frame-relay fragment 80`	(Optional) Enables fragmentation of Frame Relay frames for a Frame Relay map class. • The example command specifies that 80 payload bytes from the original Frame Relay frame will go into each fragment.

Adding the Policy Map to the Map Class

To add the policy map to the map class, use the following commands:

SUMMARY STEPS

1. **service-policy** *policy-map*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`service-policy`** *`policy-map`* **Example:** `Router(config-map-class)# service-policy output shape_policy_map`	Adds a policy map to a map class. • The example command attaches the policy map named shape_policy_map and its child policy map named llq to the PVC with the DLCI number 100. This command will also attach the policy map shape to any other PVCs that are using the map class.

Configuration Examples for MQC-Based Frame Relay Traffic Shaping

This section provides configuration examples to match the identified configuration tasks in the previous section.

- Configuring Class-Based Weighted Fair Queueing Example, page 12
- Configuring Class-Based Weighted Fair Queueing with Fragmentation Example, page 13

Configuring Class-Based Weighted Fair Queueing Example

The following example provides a sample configuration for Class-Based Weighted Fair Queueing (CBWFQ) with FRTS.

```
class-map voice
 match ip dscp ef
policy-map llq
 class voice
  priority 32
policy-map shape_policy_map
 class class-default
  shape average 64000
  shape adaptive 32000
  service-policy llq

map-class frame-relay shape_map_class

interface serial0/0
 encapsulation frame-relay

interface serial0/0.1 point-to-point
 ip address 192.168.1.1 255.255.255.0
 frame-relay interface-dlci 100
  class shape_map_class

service-policy output shape_policy_map
```

Configuring Class-Based Weighted Fair Queueing with Fragmentation Example

The following example provides a sample configuration for CBWFQ and fragmentation with FRTS. This configuration example is exactly the same as the example shown in the section "Configuring Class-Based Weighted Fair Queueing Example", with the addition of the **frame-relay fragment** command to configure fragmentation.

```
class-map voice
 match ip dscp ef

policy-map llq
 class voice
  priority 32
policy-map shape_policy_map
 class class-default
  shape average 64000
  shape adaptive 32000
  service-policy llq

map-class frame-relay shape_map_class
 frame-relay fragment 80
 service-policy output shape_policy_map

interface serial0/0
 encapsulation frame-relay

interface serial0/0.1 point-to-point
 ip address 192.168.1.1 255.255.255.0
 frame-relay interface-dlci 100
  class shape_map_class
```

Command Reference

This section documents modified commands. All other commands used with this feature are documented in the Cisco IOS Release 12.2 command reference publications.

Modified Commands

- **shape adaptive**
- **shape fecn-adapt**

shape adaptive

To configure a Frame Relay interface or a point-to-point subinterface to estimate the available bandwidth by backward explicit congestion notification (BECN) integration while traffic shaping is enabled, use the **shape adaptive** command in policy-map class configuration mode. If traffic shaping is not enabled, this command has no effect. To leave the available bandwidth unestimated, use the **no** form of this command.

shape adaptive *mean-rate-lower-bound*

no shape adaptive

Syntax Description

mean-rate-lower-bound	Specifies the lower bound of the range of permitted bit rates.

Defaults

No default behavior or values.

Command Modes

Policy-map class configuration

Command History

Release	Modification
12.0(5)XE	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.
12.2(13)T	Support for this command was implemented on the Cisco 1700 series, Cisco 2500 series, Cisco 2600 series, Cisco 3620 router, Cisco 3631 router, Cisco 3640 router, Cisco 3660 router, Cisco 3725 router, Cisco 3745 router, Cisco 7200 series, Cisco 7400 series routers.

Usage Guidelines

When continuous BECN messages are received, the shape entity immediately decreases its maximum shape rate by one-fourth for each BECN message received until it reaches the lower bound committed information rate (CIR). If, after several intervals, the interface has not received another BECN and traffic is waiting in the shape queue, the shape entity increases the shape rate back to the maximum rate by 1/16 for each interval. A shape entity configured with the **shape adaptive** *mean-rate-lower-bound* command will always be shaped between the mean rate upper bound and the mean rate lower bound.

Examples

The following example configures a shape entity with CIR of 128 kbps and sets the lower bound CIR to 64 kbps when BECNs are received:

```
policy-map dts-p2p-all-action
  class class-p2p-all
    shape average 128000
    shape adaptive 64000
```

shape fecn-adapt

To configure a Frame Relay interface to reflect received forward explicit congestion notification (FECN) bits as backward explicit congestion notification (BECN) bits in Q.922 “Test Response” messages, use the **shape fecn-adapt** command in policy-map class configuration mode. To configure the Frame Relay interface not to reflect FECN as BECN, use the **no** form of this command.

shape fecn-adapt

no shape fecn-adapt

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Policy-map class configuration

Command History

Release	Modification
12.0(5)XE	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.
12.2(13)T	Support for this command was implemented on the Cisco 1700 series, Cisco 2500 series, Cisco 2600 series, Cisco 3620 router, Cisco 3631 router, Cisco 3640 router, Cisco 3660 router, Cisco 3725 router, Cisco 3745 router, Cisco 7200 series, Cisco 7400 series routers.

Usage Guidelines

When the downstream Frame Relay switch is congested, a Frame Relay interface or point-to-point interface receives a Frame Relay message with the FECN bit on. This message may be an indication that no traffic is waiting to carry a BECN to the far end (voice and multimedia traffic is one-way). When the **shape fecn-adapt** command is configured, a small buffer is allocated and a Frame Relay TEST RESPONSE is built on behalf of the Frame Relay switch. The Frame Relay “Test Response” is equipped with the triggering data-link connection identifier (DLCI) of the triggering mechanism. It also sets the BECN bit and sends it out to the wire.

Examples

The following example configures a shape entity with a committed information rate (CIR) of 1 Mbps and adapts the Frame Relay message with FECN to BECN:

```
policy-map dts-p2p-all-action
  class class-p2p-all
    shape average 1000000
    shape fecn-adapt
```

Related Commands

Command	Description
shape (policy-map class)	Configures an interface to shape traffic to an indicated bit rate.
shape adaptive	Configures a Frame Relay interface or a point-to-point subinterface to estimate the available bandwidth by BECN integration while traffic shaping is enabled.