Configuring Weighted Fair Queueing

This chapter describes the tasks for configuring weighted fair queueing (WFQ), class-based WFQ (CBWFQ), and low latency queueing (LLQ).

For complete conceptual information, see the "Congestion Management Overview" module.

Finding Feature Information

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 for Configuring Weighted Fair Queueing" section on page 15.

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

Weighted Fair Queueing Configuration Task List

WFQ provides traffic priority management that automatically sorts among individual traffic streams without requiring that you first define access lists. WFQ can also manage duplex data streams such as those between pairs of applications, and simplex data streams such as voice or video. There are two categories of WFQ sessions: high bandwidth and low bandwidth. Low-bandwidth traffic has effective priority over high-bandwidth traffic, and high-bandwidth traffic shares the transmission service proportionally according to assigned weights.

When WFQ is enabled for an interface, new messages for high-bandwidth traffic streams are discarded after the configured or default congestive messages threshold has been met. However, low-bandwidth conversations, which include control message conversations, continue to enqueue data. As a result, the fair queue may occasionally contain more messages than its configured threshold number specifies.

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

With standard WFQ, packets are classified by flow. Packets with the same source IP address, destination IP address, source TCP or User Datagram Protocol (UDP) port, or destination TCP or UDP port belong to the same flow. WFQ allocates an equal share of the bandwidth to each flow. WFQ is also called fair queueing because all flows are equally weighted.

Configuring WFQ

To configure WFQ on an interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# fair-queue [congestive-discard-threshold [dynamic-queues [reservable-queues]]]`	Configures an interface to use WFQ.

WFQ uses a traffic data stream discrimination registry service to determine to which traffic stream a message belongs. Refer to the table accompanying the description of the **fair-queue** (WFQ) command in the *Cisco IOS Quality of Service Solutions Command Reference* for the attributes of a message that are used to classify traffic into data streams.

Defaults are provided for the congestion threshold after which messages for high-bandwidth conversations are dropped, and for the number of dynamic and reservable queues; however, you can fine-tune your network operation by changing these defaults. Refer to the tables accompanying the description of the **fair-queue** (WFQ) command in the *Cisco IOS Quality of Service Solutions Command Reference* for the default number of dynamic queues that WFQ and CBWFQ use when they are enabled on an interface.

Monitoring WFQ

To monitor WFQ services in your network, use the following commands in EXEC mode, as needed:

Command	Purpose
`Router# show interfaces [interface]`	Displays statistical information specific to an interface.
`Router# show queue interface-type interface-number`	Displays the contents of packets inside a queue for a particular interface or virtual circuit (VC).
`Router# show queueing fair`	Displays status of the fair queueing configuration.

Class-Based Weighted Fair Queueing Configuration Task List

To configure CBWFQ, perform the tasks described in the following sections. The tasks in the first three sections are required; the tasks in the remaining sections are optional.

- Defining Class Maps, page 3 (Required)
- Configuring Class Policy in the Policy Map, page 3 (Required)
- Attaching the Service Policy and Enabling CBWFQ, page 5 (Required)
- Modifying the Bandwidth for an Existing Policy Map Class, page 6 (Optional)
- Modifying the Queue Limit for an Existing Policy Map Class, page 6 (Optional)

- Deleting Classes, page 6 (Optional)
- Deleting Policy Maps, page 7 (Optional)
- Verifying Configuration of Policy Maps and Their Classes, page 7 (Optional)

See the end of this chapter for the section "CBWFQ Configuration Examples."

Defining Class Maps

To create a class map containing match criteria against which a packet is checked to determine if it belongs to a class—and to effectively create the class whose policy can be specified in one or more policy maps—use the first command in global configuration mode to specify the class map name, then use one of the following commands in class-map configuration mode, as needed:

	Command	Purpose
Step 1	`Router(config)# class-map class-map-name`	Specifies the name of the class map to be created.
Step 2	`Router(config-cmap)# match access-group {access-group | name access-group-name}`	Specifies the name of the access control list (ACL) against whose contents packets are checked to determine if they belong to the class. CBWFQ supports numbered and named ACLs.
	or	
	`Router(config-cmap)# match input-interface interface-name`	Specifies the name of the input interface used as a match criterion against which packets are checked to determine if they belong to the class.
	or	
	`Router(config-cmap)# match protocol protocol`	Specifies the name of the protocol used as a match criterion against which packets are checked to determine if they belong to the class.
	or	
	`Router(config-cmap)# match mpls experimental number`	Specifies the value of the EXP field to be used as a match criterion against which packets are checked to determine if they belong to the class.

Other match criteria can be used when defining class maps. For additional match criteria, see "Applying QoS Features Using the MQC" module.

Configuring Class Policy in the Policy Map

To configure a policy map and create class policies that make up the service policy, use the **policy-map** command to specify the policy map name, then use one or more of the following commands to configure policy for a standard class or the default class:

- **class**
- **bandwidth** (policy-map class)
- **fair-queue**
- **queue-limit** or **random-detect**

For each class that you define, you can use one or more of the listed commands to configure class policy. For example, you might specify bandwidth for one class and both bandwidth and queue limit for another class.

The default class of the policy map (commonly known as 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.

You can configure class policies for as many classes as are defined on the router, as follows:

- For Cisco IOS XE Release 2.1 and 2.2, you can configure 8 classes for each policy map.
- For Cisco IOS XE Release 2.3 and higher, you can configure 256 classes for each policy map.

However, the total amount of bandwidth allocated for all classes included in a policy map must not exceed 99 percent of the available bandwidth on the interface. The other 1 percent is used to control and route traffic.

To configure class policies in a policy map, perform the optional tasks described in the following sections. If you do not perform the steps in these sections, the default actions are used.

- Configuring Class Policy Using WRED Packet Drop, page 4 (Optional)
- Configuring the Class-Default Class Policy, page 5 (Optional)

Configuring Class Policy Using WRED Packet Drop

To configure a policy map and create class policies comprising the service policy, use the first command in global configuration mode, as needed, to specify the policy map name, then use the following commands in policy-map class configuration mode, as needed, to configure policy for a standard class. To configure policy for the default class, see the section "Configuring the Class-Default Class Policy" in this chapter.

	Command	Purpose
Step 1	`Router(config)# policy-map policy-map`	Specifies the name of the policy map to be created or modified.
Step 2	`Router(config-pmap)# class class-name`	Specifies the name of a class to be created and included in the service policy.
Step 3	`Router(config-pmap-c)# bandwidth {bandwidth-kbps | percent percent}`	Specifies the amount of bandwidth, in kbps, or percentage of available bandwidth to be assigned to the class. The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.
Step 4	`Router(config-pmap-c)# random-detect`	Enables WRED.

Step 5	`Router(config-pmap-c)# random-detect exponential-weighting-constant exponent`
	
	or
	
	`Router(config-pmap-c)# random-detect precedence precedence min-threshold max-threshold mark-prob-denominator`
		Configures the exponential weight factor used in calculating the average queue length.
		
		Configures WRED parameters for packets with a specific IP precedence. Repeat this command for each precedence.

To configure policy for more than one class in the same policy map, repeat Step 2 through Step 5.

Configuring the Class-Default Class Policy

The class-default class is used to classify traffic that does not fall into one of the defined classes. Once a packet is classified, all of the standard mechanisms that can be used to differentiate service among the classes apply. The class-default class was predefined when you created the policy map, but you must configure it. If no default class is configured, then by default the traffic that does not match any of the configured classes is flow classified and given best-effort treatment.

By default, the class-default class is defined as WFQ. However, configuring the default class with the **bandwidth** policy-map class configuration command disqualifies the default class as WFQ.

To configure a policy map and configure the class-default class to use WRED packet drop, use the first command in global configuration mode to specify the policy map name, then to configure policy for the default class use the following commands in policy-map class configuration mode:

	Command	Purpose
Step 1	`Router(config)# policy-map` *policy-map*	Specifies the name of the policy map to be created or modified.
Step 2	`Router(config-pmap)# class class-default` *default-class-name*	Specifies the default class so that you can configure or modify its policy.
Step 3	`Router(config-pmap-c)# bandwidth` {*bandwidth-kbps* | `percent` *percent*} or `Router(config-pmap-c)# fair-queue` [*number-of-dynamic-queues*]	Specifies the amount of bandwidth, in kbps, or percentage of available bandwidth to be assigned to the class. The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead. Specifies the number of dynamic queues to be reserved for use by WFQ running on the default class The number of dynamic queues is derived from the bandwidth of the interface. Refer to the tables accompanying the description of the **fair-queue** (WFQ) command in the *Cisco IOS Quality of Service Solutions Command Reference* for the default number of dynamic queues that WFQ and CBWFQ use when they are enabled on an interface.
Step 4	`Router(config-pmap-c)# random-detect`	Enables WRED.
Step 5	`Router(config-pmap-c)# random-detect exponential-weighting-constant` *exponent* or `Router(config-pmap-c)# random-detect precedence` *precedence min-threshold max-threshold mark-prob-denominator*	Configures the exponential weight factor used in calculating the average queue length. Configures WRED parameters for packets with a specific IP precedence. Repeat this command for each precedence.

Attaching the Service Policy and Enabling CBWFQ

To attach a service policy to the output interface and enable CBWFQ on the interface, use the following command in interface configuration mode. When CBWFQ is enabled, all classes configured as part of the service policy map are installed in the fair queueing system.

Command	Purpose
`Router(config-if)# service-policy output policy-map`	Enables CBWFQ and attaches the specified service policy map to the output interface.

Modifying the Bandwidth for an Existing Policy Map Class

To change the amount of bandwidth allocated for an existing class, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# policy-map policy-map`	Specifies the name of the policy map containing the class to be modified.
Step 2	`Router(config-pmap)# class class-name`	Specifies the name of a class whose bandwidth you want to modify.
Step 3	`Router(config-pmap-c)# bandwidth {bandwidth-kbps | percent percent}`	Specifies the new amount of bandwidth, in kbps, or percentage of available bandwidth to be used to reconfigure the class. The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.

Modifying the Queue Limit for an Existing Policy Map Class

To change the maximum number of packets that can accrue in a queue reserved for an existing class, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# policy-map policy-map`	Specifies the name of the policy map containing the class to be modified.
Step 2	`Router(config-pmap)# class class-name`	Specifies the name of a class whose queue limit you want to modify.
Step 3	`Router(config-pmap-c)# queue-limit number-of-packets`	Specifies the new maximum number of packets that can be queued for the class to be reconfigured. The default and maximum number of packets is 64.

Deleting Classes

To delete one or more class maps from a service policy map, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# policy-map policy-map`	Specifies the name of the policy map containing the classes to be deleted.
Step 2	`Router(config-pmap)# no class class-name`	Specifies the name of the classes to be deleted.
Step 3	`Router(config-pmap-c)# no class class-default`	Deletes the default class.

Deleting Policy Maps

To delete a policy map, use the following command in global configuration mode:

Command	Purpose
`Router(config)# no policy-map policy-map`	Specifies the name of the policy map to be deleted.

Verifying Configuration of Policy Maps and Their Classes

To display the contents of a specific policy map, a specific class from a specific policy map, or all policy maps configured on an interface, use the following commands in EXEC mode, as needed:

Command	Purpose
`Router# show policy-map policy-map`	Displays the configuration of all classes that make up the specified policy map.
`Router# show policy-map policy-map class class-name`	Displays the configuration of the specified class of the specified policy map.
`Router# show policy-map interface interface-name`	Displays the configuration of all classes configured for all policy maps on the specified interface.

Low Latency Queueing Configuration Task List

To configure LLQ, perform the tasks described in the following sections. The task in the first section is required; the tasks in the remaining sections are optional.

- Configuring LLQ, page 8 (Required)
- Verifying LLQ, page 8 (Optional)
- Verifying LLQ, page 8 (Optional)
- To verify the LLQ configuration, use the following command in EXEC mode:, page 8 (Optional)

See the end of this chapter for the section "LLQ Configuration Examples."

Configuring LLQ

To give priority to a class within a policy map, use the following command in policy-map class configuration mode:

Command	Purpose
`Router(config-pmap-c)# priority bandwidth`	Reserves a strict priority queue for this class of traffic.

Verifying LLQ

To verify the LLQ configuration, use the following command in EXEC mode:

Command	Purpose
`Router# show policy-map interface interface-name`	Displays the configuration of all classes configured for all traffic policies on the specified interface. Displays if packets and bytes were discarded or dropped for the priority class in the traffic policy attached to the interface.

WFQ Configuration Examples

The following example requests a fair queue with a congestive discard threshold of 64 messages, 512 dynamic queues, and 18 RSVP queues:

```
Router(config)# interface Serial 3/0
Router(config-if)# ip unnumbered Ethernet 0/0
Router(config-if)# fair-queue 64 512 18
```

For information on how to configure WFQ, see the section "Weighted Fair Queueing Configuration Task List" in this chapter.

CBWFQ Configuration Examples

The following sections provide CBWFQ configuration examples:

- Class Map Configuration: Example, page 9
- Policy Creation: Example, page 9
- Policy Attachment to Interfaces: Example, page 9
- CBWFQ Using WRED Packet Drop: Example, page 10
- Display Service Policy Map Content: Examples, page 10

For information on how to configure CBWFQ, see the section "Class-Based Weighted Fair Queueing Configuration Task List" in this chapter.

Class Map Configuration: Example

In the following example, ACLs 101 and 102 are created. Next, two class maps are created and their match criteria are defined. For the first map class, called class1, the numbered ACL 101 is used as the match criterion. For the second map class, called class2, the numbered 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)# access-list 101 permit udp host 10.10.10.10 host 10.10.10.20 range 16384
20000
Router(config# access-list 102 permit udp host 10.10.10.10 host 10.10.10.20 range 53000
56000

Router(config)# class-map class1
Router(config-cmap)# match access-group 101
Router(config-cmap)# exit

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

Policy Creation: Example

In the following example, a policy map called policy1 is defined to contain policy specification for the two classes, class1 and class2. The match criteria for these classes were defined in the previous "Class Map Configuration: Example" section.

For class1, the policy specifies the bandwidth allocation request and the maximum number of packets that the queue for this class can accumulate. For class2, the policy specifies only the bandwidth allocation request, so the default queue limit of 64 packets is assumed.

```
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
```

Policy Attachment to Interfaces: Example

The following example shows how to attach an existing policy map. After you define a policy map, you can attach it to one or more interfaces to specify the service policy for those interfaces. Although you can assign the same policy map to multiple interfaces, each interface can have only one policy map attached at the input and one policy map attached at the output.

The policy map in this example was defined in the previous section, "Policy Creation: Example."

```
Router(config)# interface e1/1
Router(config-if)# service output policy1
Router(config-if)# exit

Router(config)# interface fa1/0/0
Router(config-if)# service output policy1
Router(config-if)# exit
```

CBWFQ Using WRED Packet Drop: Example

In the following example, the class map called class1 is created and defined to use the input FastEthernet interface 0/1 as a match criterion to determine if packets belong to the class. Next, the policy map policy1 is defined to contain policy specification for class1, which is configured for WRED packet drop.

```
Router(config)# class-map class1
Router(config-cmap)# match input-interface FastEthernet0/1
!
Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth 1000
Router(config-pmap-c)# random-detect
!
Router(config)# interface serial0/0
Router(config-if)# service-policy output policy1
!
```

Display Service Policy Map Content: Examples

The following examples show how to display the contents of service policy maps. Four methods can be used to display the contents.

- Display all classes that make up a specified service policy map
- Display all classes configured for all service policy maps
- Display a specified class of a service policy map
- Display all classes configured for all service policy maps on a specified interface

Displaying All Classes for a Specified Service Policy Map: Example

The following example displays the contents of the service policy map called pol1:

```
Router# show policy-map po1

Policy Map po1
 Weighted Fair Queueing
    Class class1
       Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class2
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class3
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class4
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class5
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class6
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class7
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class8
         Bandwidth 937 (kbps)  Max thresh 64 (packets)
```

Displaying All Classes for All Service Policy Maps: Example

The following example displays the contents of all policy maps on the router:

```
Router# show policy-map

Policy Map poH1
 Weighted Fair Queueing
    Class class1
       Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class2
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class3
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class4
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class5
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class6
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class7
        Bandwidth 937 (kbps)  Max thresh 64 (packets)
    Class class8
         Bandwidth 937 (kbps)  Max thresh 64 (packets)
Policy Map policy2
 Weighted Fair Queueing
    Class class1
       Bandwidth 300 (kbps) Max thresh 64 (packets)
    Class class2
        Bandwidth 300  (kbps)  Max thresh 64 (packets)
    Class class3
        Bandwidth 300 (kbps)  Max thresh 64 (packets)
    Class class4
        Bandwidth 300 (kbps)  Max thresh 64 (packets)
    Class class5
        Bandwidth 300 (kbps)  Max thresh 64 (packets)
    Class class6
        Bandwidth 300 (kbps)  Max thresh 64 (packets)
```

Displaying Specified Class for a Service Policy Map: Example

The following example displays configurations for the class called class7 that belongs to the policy map called po1:

```
Router# show policy-map po1 class class7

Class class7
 Bandwidth 937 (kbps) Max Thresh 64 (packets)
```

Displaying All Classes for All Service Policy Maps on a Specified Interface: Example

The following example displays configurations for classes on the output Ethernet interface 2/0. The numbers shown in parentheses are for use with the Management Information Base (MIB).

```
Router# show policy-map interface e2/0

Ethernet2/0

  Service-policy output:p1 (1057)

    Class-map:c1 (match-all) (1059/2)
      19 packets, 1140 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match:ip precedence 0  (1063)
      Weighted Fair Queueing
```

```
      Output Queue:Conversation 265
      Bandwidth 10 (%) Max Threshold 64 (packets)
      (pkts matched/bytes matched) 0/0
      (depth/total drops/no-buffer drops) 0/0/0

 Class-map:c2 (match-all) (1067/3)
   0 packets, 0 bytes
   5 minute offered rate 0 bps, drop rate 0 bps
   Match:ip precedence 1  (1071)
   Weighted Fair Queueing
     Output Queue:Conversation 266
     Bandwidth 10 (%) Max Threshold 64 (packets)
     (pkts matched/bytes matched) 0/0
     (depth/total drops/no-buffer drops) 0/0/0

Class-map:class-default (match-any) (1075/0)
   8 packets, 2620 bytes
   5 minute offered rate 0 bps, drop rate 0 bps
   Match:any  (1079)
```

LLQ Configuration Examples

Question for Reviewers------Are these examples still valid?

The following sections provide LLQ configuration examples:

- LLQ Configuration: Example, page 12
- Virtual Template Configuration: Example, page 13
- Multilink Bundle Configuration: Example, page 13

For information on how to configure LLQ, see the section "Low Latency Queueing Configuration Task List" in this chapter.

LLQ Configuration: Example

In the following example, a strict priority queue with a guaranteed allowed bandwidth of 50 kbps is reserved for traffic that is sent from the source address 10.10.10.10 to the destination address 10.10.10.20, in the range of ports 16384 through 20000 and 53000 through 56000.

First, the following commands configure access list 102 to match the desired voice traffic:

```
Router(config)# access-list 102 permit udp host 10.10.10.10 host 10.10.10.20 range 16384
20000
Router(config)# access-list 102 permit udp host 10.10.10.10 host 10.10.10.20 range 53000
56000
```

Next, the class map voice is defined, and the policy map called policy1 is created; a strict priority queue for the class voice is reserved, a bandwidth of 20 kbps is configured for the class bar, and the default class is configured for WFQ. The **service-policy** command then attaches the policy map to the PVC interface 0/102 on the subinterface serial1/0.2.

```
Router(config)# class-map voice
Router(config-cmap)# match access-group 102

Router(config)# policy-map policy1
Router(config-pmap)# class voice
Router(config-pmap-c)# priority 50
Router(config-pmap)# class bar
```

```
Router(config-pmap-c)# bandwidth 20
Router(config-pmap)# class class-default
Router(config-pmap-c)# fair-queue

Router(config)# interface serial1/0.2
Router(config-subif)# pvc 0/102
Router(config-subif-vc)# service-policy output policy1
```

Virtual Template Configuration: Example

The following example configures a strict priority queue in a virtual template configuration with CBWFQ. Traffic on virtual template 1 that is matched by access list 102 will be directed to the strict priority queue.

First, the class map voice is defined, and the policy map called policy1 is created. A strict priority queue (with a guaranteed allowed bandwidth of 50 kbps) is reserved for the class called voice.

```
Router(config)# class-map voice
Router(config-cmap)# match access-group 102
Router(config)# policy-map policy1
Router(config-pmap)# class voice
Router(config-pmap-c)# priority 50
```

Next, the **service-policy** command attaches the policy map called policy1 to virtual template 1.

```
Router(config)# multilink virtual-template 1
Router(config)# interface virtual-template 1
Router(config-if)# ip address 172.16.1.1 255.255.255.0
Router(config-if)# no ip directed-broadcast
Router(config-if)# service-policy output policy1
Router(config-if)# ppp multilink
Router(config-if)# ppp multilink fragment-delay 20
Router(config-if)# ppp multilink interleave
Router(config-if)# end

Router(config)# interface serial 2/0
Router(config-if)# bandwidth 256
Router(config-if)# no ip address
Router(config-if)# no ip directed-broadcast
Router(config-if)# encapsulation ppp
Router(config-if)# no fair-queue
Router(config-if)# clockrate 256000
Router(config-if)# ppp multilink
```

Multilink Bundle Configuration: Example

The following example configures a strict priority queue in a multilink bundle configuration with CBWFQ. Traffic on serial interface 2/0 that is matched by access list 102 will be directed to the strict priority queue. The advantage to using multilink bundles is that you can specify different **priority** parameters on different interfaces. To specify different **priority** parameters, you would configure two multilink bundles with different parameters.

First, the class map voice is defined, and the policy map called policy1 is created. A strict priority queue (with a guaranteed allowed bandwidth of 50 kbps) is reserved for the class called voice.

```
Router(config)# class-map voice
Router(config-cmap)# match access-group 102
Router(config)# policy-map policy1
Router(config-pmap)# class voice
```

```
Router(config-pmap-c)# priority 50
```

The following commands create multilink bundle 1. The policy map called policy1 is attached to the bundle by the **service-policy** command.

```
Router(config)# interface multilink 1
Router(config-if)# ip address 172.17.254.161 255.255.255.248
Router(config-if)# no ip directed-broadcast
Router(config-if)# no ip mroute-cache
Router(config-if)# service-policy output policy1
Router(config-if)# ppp multilink
Router(config-if)# ppp multilink fragment-delay 20
Router(config-if)# ppp multilink interleave
```

In the next part of the example, the **multilink-group** command configures serial interface 2/0 to be part of multilink bundle 1, which effectively directs traffic on serial interface 2/0 that is matched by access list 102 to the strict priority queue:

```
Router(config)# interface serial 2/0
Router(config-if)# bandwidth 256
Router(config-if)# no ip address
Router(config-if)# no ip directed-broadcast
Router(config-if)# encapsulation ppp
Router(config-if)# no fair-queue
Router(config-if)# clockrate 256000
Router(config-if)# ppp multilink
Router(config-if)# multilink-group 1
```

Feature Information for Configuring Weighted Fair Queueing

Table 1 lists the features in this module and provides links to specific configuration information.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS XE Software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS XE Software release that introduced support for a given feature in a given Cisco IOS XE Software release train. Unless noted otherwise, subsequent releases of that Cisco IOS XE software release train also support that feature.

Table 1 Feature Information for Configuring Weighted Fair Queueing

Feature Name	Releases	Feature Information
Class-Based Weighted Fair Queueing	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers. The following section provides information about this feature: • Class-Based Weighted Fair Queueing Configuration Task List, page 2.
Low Latency Queueing	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers. The following section provides information about this feature: • Low Latency Queueing Configuration Task List, page 7.
Weighted Fair Queueing	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers. The following section provides information about this feature: • Weighted Fair Queueing Configuration Task List, page 1.