



# Configuring Modular QoS Congestion Avoidance

---

This chapter covers the following topics:

- [Modular QoS Congestion Avoidance](#) , on page 1
- [Tail Drop and the FIFO Queue](#), on page 2
- [Random Early Detection and TCP](#), on page 4
- [Weighted Random Early Detection](#), on page 6
- [Explicit Congestion Notification](#) , on page 13

## Modular QoS Congestion Avoidance

Congestion avoidance techniques monitor traffic flow to anticipate and avoid congestion at common network bottlenecks. Avoidance techniques are implemented before congestion occurs as compared with congestion management techniques that control congestion after it has occurred.



---

**Note** From Cisco IOS XR Release 7.3.1 onwards, systems with Cisco NC57 line cards running in compatibility mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

Starting with Cisco IOS XR Release 7.4.1 systems with Cisco NC57 line cards running in native mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
  - L2 VPN – VPWS
-



**Note** For traffic requiring header decapsulation, the size of the header that is being removed is still included for the egress queuing actions. To offset this header size (required to achieve line rate for small frame sizes), configure an egress user policy with user overhead accounting on the egress interface. This policy can be a dummy policy configuration as well (allowing full traffic rate), if a policy isn't already in use or required on the egress interface.

You can enable user overhead accounting using the optional configuration of **accounting user-defined** *<overhead size in bytes>* while attaching the service policy on the egress interface.

Congestion avoidance is achieved through packet dropping. The router supports these QoS congestion avoidance techniques:

- [Tail Drop and the FIFO Queue, on page 2](#)
- [Random Early Detection and TCP, on page 4](#)
- [Weighted Random Early Detection, on page 6](#)

## Tail Drop and the FIFO Queue

Tail drop is a congestion avoidance technique that drops packets when an output queue is full until congestion is eliminated. Tail drop treats all traffic flow equally and does not differentiate between classes of service. It manages the packets that are unclassified, placed into a first-in, first-out (FIFO) queue, and forwarded at a rate determined by the available underlying link bandwidth.

### Configure Tail Drop

Packets satisfying the match criteria for a class accumulate in the queue reserved for the class until they are serviced. The **queue-limit** command is used to define the maximum threshold for a class. When the maximum threshold is reached, the enqueued packets to the class queue result in tail drop (packet drop).

#### Restrictions

- When configuring the **queue-limit** command, you must configure one of the following commands: **priority**, **shape average**, **bandwidth** or **bandwidth remaining**, except for the default class.

#### Configuration Example

You have to accomplish the following to complete the tail drop configuration:

1. Creating (or modifying) a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Specifying the maximum limit the queue can hold for a class policy configured in a policy map.
4. Specifying priority to a class of traffic belonging to a policy map.

5. (Optional) Specifying the bandwidth allocated for a class belonging to a policy map or specifying how to allocate leftover bandwidth to various classes.
6. Attaching a policy map to an output interface to be used as the service policy for that interface.

```

Router# configure
Router(config)# class-map qos-1
Router(config-cmap)# match traffic-class 1
Router(config-cmap)# commit
Router(config-pmap)# exit

Router(config)# policy-map test-qlimit-1
Router(config-pmap)# class qos-1
Router(config-pmap-c)# queue-limit 100 us
Router(config-pmap-c)# priority level 7
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy output test-qlimit-1
Router(config-if)# commit

```

### Running Configuration

```

class-map qos-1
  match traffic-class 1
  commit

policy-map test-qlimit-1
  class qos-1
    queue-limit 100 us
    priority level 7
  !
  class class-default
  !
end-policy-map
!

```

### Verification

```

Router# show qos int hundredGigE 0/6/0/18 output

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220 -- output policy
NPU Id: 3
Total number of classes: 2
Interface Bandwidth: 100000000 kbps
VOQ Base: 11176
VOQ Stats Handle: 0x88550ea0
Accounting Type: Layer1 (Include Layer 1 encapsulation and above)
-----
Level1 Class (HP7) = qos-1
Egressq Queue ID = 11177 (HP7 queue)
TailDrop Threshold = 1253376 bytes / 100 us (100 us)
WRED not configured for this class

Level1 Class = class-default

```

```

Egressq Queue ID           = 11176 (Default LP queue)
Queue Max. BW.             = 101803495 kbps (default)
Queue Min. BW.             = 0 kbps (default)
Inverse Weight / Weight    = 1 (BWR not configured)
TailDrop Threshold         = 1253376 bytes / 10 ms (default)
WRED not configured for this class

```

### Related Topics

- [Tail Drop and the FIFO Queue, on page 2](#)

### Associated Commands

- [queue-limit](#)

## Random Early Detection and TCP

The Random Early Detection (RED) congestion avoidance technique takes advantage of the congestion control mechanism of TCP. By randomly dropping packets prior to periods of high congestion, RED tells the packet source to decrease its transmission rate. Assuming the packet source is using TCP, it decreases its transmission rate until all packets reach their destination, indicating that the congestion is cleared. You can use RED as a way to cause TCP to slow transmission of packets. TCP not only pauses, but it also restarts quickly and adapts its transmission rate to the rate that the network can support.

RED distributes losses in time and maintains normally low queue depth while absorbing traffic bursts. When enabled on an interface, RED begins dropping packets when congestion occurs at a rate you select during configuration.

## Configure Random Early Detection

The **random-detect** command with the **default** keyword must be used to enable random early detection (RED).

### Guidelines

If you configure the **random-detect default** command on any class including class-default, you must configure one of the following commands: **shape average**, **bandwidth**, and **bandwidth remaining**.

### Configuration Example

You have to accomplish the following to complete the random early detection configuration:

1. Creating (or modifying) a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Enabling RED with default minimum and maximum thresholds.
4. (Optional) Specifying the bandwidth allocated for a class belonging to a policy map or specifying how to allocate leftover bandwidth to various classes.

5. (Optional) Shaping traffic to the specified bit rate or a percentage of the available bandwidth.
6. Attaching a policy map to an output interface to be used as the service policy for that interface.

```

Router# configure
Router(config)# class-map qos-1
Router(config-cmap)# match traffic-class 1
Router(config-cmap)# commit
Router(config-pmap)# exit

Router# configure
Router(config)# policy-map test-wred-2
Router(config-pmap)# class qos-1
Router(config-pmap-c)# random-detect default
Router(config-pmap-c)# shape average percent 10
Router(config-pmap-c)# end-policy-map
Router(config)# commit
Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy output test-wred-2
Router(config-if)# commit

```

### Running Configuration

```

class-map qos-1
  match traffic-class 1
  commit

policy-map test-wred-2
  class qos-1
    random-detect default
    shape average percent 10
  !
  class class-default
  !
  end-policy-map
!

interface HundredGigE 0/6/0/18
  service-policy output test-wred-2
!

```

### Verification

```
Router# show qos int hundredGigE 0/6/0/18 output
```

```

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220 -- output policy
NPU Id:                               3
Total number of classes:                2
Interface Bandwidth:                    100000000 kbps
VOQ Base:                               11176
VOQ Stats Handle:                       0x88550ea0
Accounting Type:                         Layer1 (Include Layer 1 encapsulation and above)
-----
Level1 Class                            = qos-1
Egressq Queue ID                         = 11177 (LP queue)
Queue Max. BW.                           = 10082461 kbps (10 %)

```

```

Queue Min. BW.                = 0 kbps (default)
Inverse Weight / Weight       = 1 (BWR not configured)
Guaranteed service rate       = 10000000 kbps
TailDrop Threshold            = 12517376 bytes / 10 ms (default)

Default RED profile
WRED Min. Threshold           = 12517376 bytes (10 ms)
WRED Max. Threshold           = 12517376 bytes (10 ms)

Level1 Class                   = class-default
Egressq Queue ID              = 11176 (Default LP queue)
Queue Max. BW.                = 101803495 kbps (default)
Queue Min. BW.                = 0 kbps (default)
Inverse Weight / Weight       = 1 (BWR not configured)
Guaranteed service rate       = 50000000 kbps
TailDrop Threshold            = 62652416 bytes / 10 ms (default)
WRED not configured for this class

```

### Related Topics

- [Random Early Detection and TCP, on page 4](#)

### Associated Commands

- [random-detect](#)

## Weighted Random Early Detection

The Weighted Random Early Detection (WRED) drops packets selectively based on any specified criteria, like discard-class. WRED uses this matching criteria to determine how to treat different types of traffic.

You can configure WRED using the **random-detect** command and different discard-class values. The value can be range or a list of values that are valid for that field. You can also use minimum and maximum queue thresholds to determine the dropping point. Ensure that the WRED maximum threshold value is close to the queue limit. When the maximum threshold value is reached, packets start to get dropped.

When a packet arrives, the following actions occur:

- The average queue size is calculated.
- If the average queue size is less than the minimum queue threshold, the arriving packet is queued.
- If the average queue size is between the minimum queue threshold for that type of traffic and the maximum threshold for the interface, the packet is either dropped or queued, depending on the packet drop probability for that type of traffic.
- If the average queue size is greater than the maximum threshold, the packet is dropped.

## Average Queue Size for WRED

The router automatically determines the parameters to use in the WRED calculations. The average queue size is based on the previous average and current size of the queue. The formula is:

$$\text{average} = (\text{old\_average} * (1-2-x)) + (\text{current\_queue\_size} * 2-x)$$

where  $x$  is the exponential weight factor.

For high values of  $x$ , the previous average becomes more important. A large factor smooths out the peaks and lows in queue length. The average queue size is unlikely to change very quickly, avoiding a drastic change in size. The WRED process is slow to start dropping packets, but it may continue dropping packets for a time after the actual queue size has fallen below the minimum threshold. The slow-moving average accommodates temporary bursts in traffic.

**Note**

- The exponential weight factor,  $x$ , is fixed and is not user configurable.
- If the value of  $x$  gets too high, WRED does not react to congestion. Packets are sent or dropped as if WRED were not in effect.
- If the value of  $x$  gets too low, WRED overreacts to temporary traffic bursts and drops traffic unnecessarily.

For low values of  $x$ , the average queue size closely tracks the current queue size. The resulting average may fluctuate with changes in the traffic levels. In this case, the WRED process responds quickly to long queues. Once the queue falls below the minimum threshold, the process stops dropping packets.

## Configure Weighted Random Early Detection

This configuration task is similar to that used for RED except that the **random-detect** command is not configured in RED.

**Restrictions**

- You cannot use the **random-detect** command in a class configured with the **priority** command, because WRED cannot be configured in a class that has been set for priority queueing (PQ).
- When configuring the **random-detect** command, you must configure one of the following commands: **shape average**, **bandwidth**, and **bandwidth remaining**.

**Configuration Example**

You have to accomplish the following to complete the random early detection configuration:

1. Creating (or modifying) a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Enabling WRED by specifying the match criteria (discard-class).
4. (Optional) Specifying the bandwidth allocated for a class belonging to a policy map or specifying how to allocate leftover bandwidth to various classes.
5. (Optional) Shaping traffic to the specified bit rate or a percentage of the available bandwidth.
6. (Optional) Changing queue limit to fine-tune the amount of buffers available for each queue.
7. Attaching a policy map to an output interface to be used as the service policy for that interface.

```

Router# configure
Router(config)# class-map qos-1
Router(config-cmap)# match traffic-class 1
Router(config-cmap)# commit
Router(config-pmap)# exit

Router# configure
Router(config)# policy-map test-wred-1
Router(config-pmap)# class qos-1
Router(config-pmap-c)# random-detect default
Router(config-pmap-c)# random-detect discard-class 0 10 ms 500 ms
Router(config-pmap-c)# shape average percent 10
Router(config-pmap-c)# commit

Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy output policy1
Router(config-if)# commit

```

### Running Configuration

```

class-map qos-1
  match traffic-class 1
commit

policy-map test-wred-1
  class qos-1
    random-detect default
    random-detect discard-class 0 10 ms 500 ms
    shape average percent 10
  !
  class class-default
  !
end-policy-map
!

interface HundredGigE 0/6/0/18
  service-policy output test-wred-1
!

```

### Verification

```
Router# show qos int hundredGigE 0/0/0/20 output
```

```

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/20 ifh 0x38 -- output policy
NPU Id:                                0
Total number of classes:                2
Interface Bandwidth:                    100000000 kbps
Policy Name:                             test-wred-1
VOQ Base:                                1184
Accounting Type:                         Layer1 (Include Layer 1 encapsulation and above)
-----
Level1 Class                            = qos-1
Egressq Queue ID                         = 1185 (IP queue)
Queue Max. BW.                           = 10000152 kbps (10 %)
Queue Min. BW.                           = 0 kbps (default)
Inverse Weight / Weight                   = 1 / (BWR not configured)
Guaranteed service rate                   = 10000000 kbps

```



```

Peak burst = 36864 bytes (default)
TailDrop Threshold = 1250000896 bytes / 1000 ms (default)

WRED profile for Discard_Class 0
WRED Min. Threshold = 12499968 bytes (10 ms)
WRED Max. Threshold = 624999936 bytes (500 ms)

Default RED profile
WRED Min. Threshold = 7499776 bytes (6 ms)
WRED Max. Threshold = 12499968 bytes (10 ms)

WRED ECN = Disabled

Levell Class = class-default
Egressq Queue ID = 1184 (Default LP queue)
Queue Max. BW. = no max (default)
Queue Min. BW. = 0 kbps (default)
Inverse Weight / Weight = 1 / (BWR not configured)
Guaranteed service rate = 50000000 kbps
Peak burst = 36864 bytes (default)
TailDrop Threshold = 62499840 bytes / 10 ms (default)
WRED not configured for this class
    
```

**Related Topics**

- [Weighted Random Early Detection, on page 6](#)
- [Configure Random Early Detection, on page 4](#)

**Associated Commands**

- [random-detect](#)

## Configure WRED Counters by Class

*Table 1: Feature History Table*

Feature Name	Release Information	Feature Description
Configure WRED Counters by Class	Release 7.4.1	<p>This feature enables the display of WRED statistics per class, thus providing a more accurate and granular statistics profile for packet drops. Such insight allows you to monitor, anticipate, and avoid congestion at common bottlenecks on your network.</p> <p>This functionality introduces the <a href="#">hw-module profile qos wred-stats-enable</a> command and modifies the output of the <b>show policy-map interface</b> command.</p>

- **Statistics profile**—Up until now, running the **show policy-map interface** command displayed the drop statistics per class under tail drop. Which meant that no matter what drop mechanism you deployed ([Tail Drop and the FIFO Queue, on page 2](#), [Random Early Detection and TCP, on page 4](#), or [Weighted Random Early Detection, on page 6](#)), the statistics profile that displayed was a number aggregated under the tail drop entry. Given that WRED provides a far more granular control over packet drops than the other mechanisms, the generic statistics profile didn't provide you with the validation that you required for your WRED actions.
- **WRED counters by class**—This functionality introduces a new command, **hw-module profile qos wred-stats-enable**. Configuring this command enables the display of WRED and RED statistics per class, thus providing a more accurate and granular statistic for packet drops.
- **Why WRED counters by class**—Prior to this functionality, you did not have visibility into the WRED drop counters and had no way to validate WRED actions. With this improvement, the **show policy-map interface** command provides the much-needed visibility by clearly depicting the drop counters per class for WRED. This visibility gives you the confidence that WRED is indeed in action and is providing the requisite congestion avoidance.
- **Guidelines and limitations**
  - WRED max-drop threshold statistics are unavailable.
  - WRED statistics are collected separately for each drop precedence value.
- **WRED counters support cheat-sheet**—Here's a quick look at some key support areas and their details for WRED counters by class.

Support	Details
Line card	Supported on Cisco NC57 line cards with external Ternary Content-Addressable Memory (TCAM) operating in native mode.
Statistics	WRED drop count is available for all the supported discard-values (from 0 through 2).

- **Configure WRED counters by class**—To configure WRED counters by class you must:
  - Configure the **hw-module profile** for WRED statistics. Manually reload the chassis to activate the profile.
  - Configure a class map.
  - Create a service policy map.
  - Attach the service policy to an interface.

```
/*Configure hw-module profile for wred stats*/
Router#conf
Router(config)#hw-module profile qos wred-stats-enable
Router(config)#commit
Router#reload

/*Configure a class map*/
Router(config)#class-map TC3
Router(config-cmap)#match traffic-class 3
Router(config-cmap)#commit
```

```

Router(config-cmap) #end-class-map

/*Create a service policy map*/
Router(config) #policy-map EGRESS_WRED
Router(config-pmap) #class TC3
Router(config-pmap-c) #shape average 1 gbps
Router(config-pmap-c) #random-detect discard-class 0 1 ms 2 ms
Router(config-pmap-c) #random-detect discard-class 1 3 ms 5 ms
Router(config-pmap-c) #random-detect discard-class 2 10 ms 20 ms
Router(config-pmap-c) #commit
Router(config-pmap) end-policy-map

/*Attach the service policy to an interface*/
Router(config) #int hundredGigE 0/7/0/2
Router(config-if) #service-policy output EGRESS_WRED
Router(config-if) #commit

```

### • Running Configuration

```

class-map match-any TC3
  match traffic-class 3
end-class-map
!
policy-map EGRESS_WRED
  class TC3
    shape average 1 gbps
    random-detect discard-class 0 1 ms 2 ms
    random-detect discard-class 1 3 ms 5 ms
    random-detect discard-class 2 10 ms 20 ms
  !
  class class-default
  !
end-policy-map
!
interface HundredGigE0/7/0/2
  service-policy output EGRESS_WRED
  ipv4 address 192.168.0.1 255.255.255.0
!

```

### • Verification

```

Router#show qos int hundredGigE 0/7/0/2 output
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/7/0/2 ifh 0xe000088 -- output policy
NPU Id: 0
Total number of classes: 2
Interface Bandwidth: 100000000 kbps
Policy Name: EGRESS_WRED
SPI Id: 0x0
VOQ Base: 1160
PFC enabled: 0
Accounting Type: Layer1 (Include Layer 1 encapsulation and above)
-----
Level1 Class = TC3
Egressq Queue ID = 1163 (LP queue)
Queue Max. BW. = 1000000 kbps (1 gbits/sec)
Queue Min. BW. = 0 kbps (default)
Inverse Weight / Weight = 1 / (BWR not configured)
Guaranteed service rate = 1000000 kbps
Peak burst = 36864 bytes (default)
TailDrop Threshold = 4999168 bytes / 40 ms (default)

```

```

WRED profile for Discard_Class 0
WRED Min. Threshold           = 124928 bytes (1 ms)
WRED Max. Threshold           = 249856 bytes (2 ms)

WRED profile for Discard_Class 1
WRED Min. Threshold           = 374784 bytes (3 ms)
WRED Max. Threshold           = 624896 bytes (5 ms)

WRED profile for Discard_Class 2
WRED Min. Threshold           = 1249792 bytes (10 ms)
WRED Max. Threshold           = 2499840 bytes (20 ms)

Default RED profile
WRED Min. Threshold           = 256 bytes (0 ms)
WRED Max. Threshold           = 256 bytes (0 ms)

WRED ECN                       = Disabled

Levell Class                   = class-default
Egressq Queue ID               = 1160 (Default LP queue)
Queue Max. BW.                 = no max (default)
Queue Min. BW.                 = 0 kbps (default)
Inverse Weight / Weight        = 1 / (BWR not configured)
Guaranteed service rate        = 50000000 kbps
Peak burst                     = 36864 bytes (default)
TailDrop Threshold             = 62499840 bytes / 10 ms (default)
WRED not configured for this class

```

- **show policy-map interface output for statistics verification**

```
Router#show policy-map int hundredGigE 0/7/0/2 output
```

```
HundredGigE0/7/0/2 output: EGRESS_WRED
```

```

Class TC3
  Classification statistics      (packets/bytes)      (rate - kbps)
  Matched                       : 1151236/1151236000      0
  Transmitted                   : 578807/578807000       0
  Total Dropped                 : 572429/572429000     0
  Queueing statistics
  Queue ID                      : 1163
  Taildropped(packets/bytes)    : 572429/572429000

  WRED profile for Discard Class 0
  RED Transmitted (packets/bytes) : 0/0
  RED random drops (packets/bytes) : 0/0
  RED maxthreshold drops (packets/bytes) : N/A
  RED ecn marked & transmitted (packets/bytes) : N/A
  WRED profile for Discard Class 1
  RED Transmitted (packets/bytes) : 0/0
  RED random drops (packets/bytes) : 0/0
  RED maxthreshold drops (packets/bytes) : N/A
  RED ecn marked & transmitted (packets/bytes) : N/A
  WRED profile for Discard Class 2
  RED Transmitted (packets/bytes) : 578807/578807000
  RED random drops (packets/bytes) : 572429/572429000
  RED maxthreshold drops (packets/bytes) : N/A
  RED ecn marked & transmitted (packets/bytes) : N/A
  RED profile (default)
  RED Transmitted (packets/bytes) : 0/0
  RED random drops (packets/bytes) : 0/0
  RED maxthreshold drops (packets/bytes) : N/A
  RED ecn marked & transmitted (packets/bytes) : N/A

```

```

Class class-default
  Classification statistics          (packets/bytes)    (rate - kbps)
  Matched                          :                0/0                0
  Transmitted                       :                0/0                0
  Total Dropped                     :                0/0                0
  Queueing statistics
  Queue ID                          :             1160
  Taildropped (packets/bytes)       :             0/0

```

- **Clear WRED statistics on an interface**

Run the following command to clear WRED statistics on an interface.

```
Router#clear qos counters int hundredGigE 0/7/0/2 output
```

### Associated Commands

[hw-module profile qos wred-stats-enable](#)

## Explicit Congestion Notification

Weighted Random Early Detection (WRED) is implemented at the core routers of a network. Edge routers assign IP precedences to packets, as the packets enter the network. With WRED, core routers then use these precedences to determine how to treat different types of traffic. WRED provides separate thresholds and weights for different IP precedences, enabling the network to provide different qualities of service, in regard to packet dropping, for different types of traffic. Standard traffic may be dropped more frequently than premium traffic during periods of congestion.

ECN is an extension to WRED. ECN marks packets instead of dropping them when the average queue length exceeds a specific threshold value. When configured, ECN helps routers and end hosts to understand that the network is congested and slow down sending packets. However If the number of packets in the queue is above the maximum threshold, packets are dropped based on the drop probability. This is the identical treatment that a packet receives when WRED is enabled without ECN configured on the router.

RFC 3168, *The Addition of Explicit Congestion Notification (ECN) to IP*, states that with the addition of active queue management (for example, WRED) to the Internet infrastructure, routers are no longer limited to packet loss as an indication of congestion.



**Note** You cannot use this feature when you have set qos-group or mpls experimental along with a traffic class in the ingress policy.

### Implementing ECN

Implementing ECN requires an ECN-specific field that has 2 bits—the ECN-capable Transport (ECT) bit and the CE (Congestion Experienced) bit—in the IP header. The ECT bit and the CE bit can be used to make four ECN field combinations of 00 to 11. The first number is the ECT bit and the second number is the CE bit.

**Table 2: ECN Bit Setting**

ECT Bit	CE Bit	Combination Indicates
0	0	Not-ECN-capable.

ECT Bit	CE Bit	Combination Indicates
0	1	Endpoints of the transport protocol are ECN-capable.
1	0	Endpoints of the transport protocol are ECN-capable.
1	1	Congestion experienced.

The ECN field combination 00 indicates that a packet is not using ECN. The ECN field combinations 01 and 10—Called ECT(1) and ECT(0), respectively—are set by the data sender to indicate that the endpoints of the transport protocol are ECN-capable. Routers treat these two field combinations identically. Data senders can use either one or both of these two combinations. The ECN field combination 11 indicates congestion to the endpoints. Packets arriving a full queue of a router will be dropped.

### Packet Handling When ECN Is Enabled

When the number of packets in the queue is below the minimum threshold, packets are transmitted. This happens whether ECN is enabled or not, and this treatment is identical to the treatment a packet receives when WRED only is being used on the network.

If the number of packets in the queue is above the maximum threshold:

- all incoming packets are dropped when you enable only WRED.
- all incoming packets are marked as CE when you configure ECN on the router.



**Note** When the number of packets reaches the queue limit, all packets are dropped. This is the identical treatment that a packet receives when you enable WRED without ECN configured on the router.

Three different scenarios arise if the number of packets in the queue is between the minimum threshold and the maximum threshold:

- If the ECN field on the packet indicates that the endpoints are ECN-capable (that is, the ECT bit is set to 1 and the CE bit is set to 0, or the ECT bit is set to 0 and the CE bit is set to 1)—and the WRED algorithm determines that the packet should have been dropped based on the drop probability—the ECT and CE bits for the packet are changed to 1, and the packet is transmitted. This happens because ECN is enabled and the packet gets marked instead of dropped.
- If the ECN field on the packet indicates that neither endpoint is ECN-capable (that is, the ECT bit is set to 0 and the CE bit is set to 0), packet is dropped once the queue limit is reached.
- If the ECN field on the packet indicates that the network is experiencing congestion (that is, both the ECT bit and the CE bit are set to 1), the packet is transmitted. No further marking is required.

### Configuration Example

```
Router# configure
Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth percent 50
Router(config-pmap-c)# random-detect 1000 packets 2000 packets
```

```

Router(config-pmap-c) # random-detect ecn
Router(config-pmap-c) # exit
Router(config-pmap) # exit
Router(config) # commit

```

### Verification

Use the **show policy-map interface** to verify the configuration.

```

Router# show policy-map interface tenGigE 0/0/0/6 output
TenGigE0/0/0/6 output: pm-out-queue

Class cm-tc-1
  Classification statistics          (packets/bytes)    (rate - kbps)
  Matched                          : 85528554/87581239296  4830672
  Transmitted                       : 16240891/16630672384  966585
  Total Dropped                     : 69287663/70950566912  3864087
  Queueing statistics
  Queue ID                          : 1113
  Taildropped(packets/bytes)        : 69287663/70950566912

  WRED profile for
  RED Transmitted (packets/bytes)    : N/A
  RED random drops (packets/bytes)   : N/A
  RED maxthreshold drops (packets/bytes) : N/A
  RED ecn marked & transmitted (packets/bytes) : N/A
Class class-default
  Classification statistics          (packets/bytes)    (rate - kbps)
  Matched                          : 0/0                0
  Transmitted                       : 0/0                0
  Total Dropped                     : 0/0                0
  Queueing statistics
  Queue ID                          : 1112
  Taildropped(packets/bytes)        : 0/0

```




---

**Note** No ECN-specific statistics are displayed in the show output for this command. ECN is enabled if all rows display N/A, as highlighted in the example.

---

