Extreme OS ONE Switching and Routing v22.2.3.0 QoS Configuration Guide

Traffic Management

9041120-00 Rev AA
August 2026

Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

Trademarks

Extreme Networks® and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners.

For additional information on Extreme Networks trademarks, see: https://www.extremenetworks.com/about-extreme-networks/company/legal/trademarks

Open Source Declarations

Some software files have been licensed under certain open source or third-party licenses. End-user license agreements and open source declarations can be found at: https://www.extremenetworks.com/support/policies/open-source-declaration/

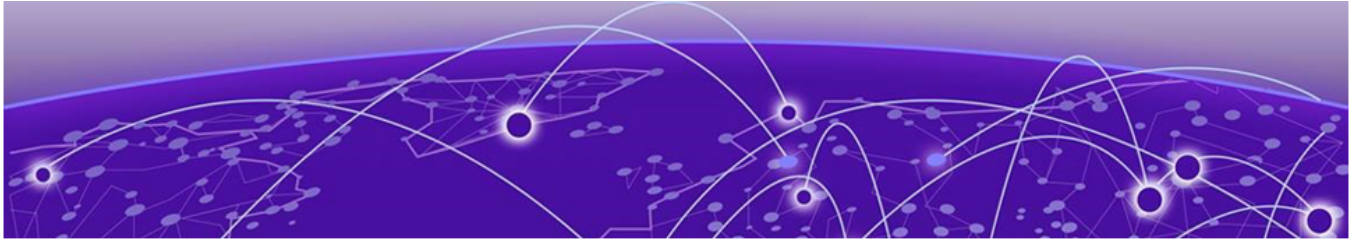
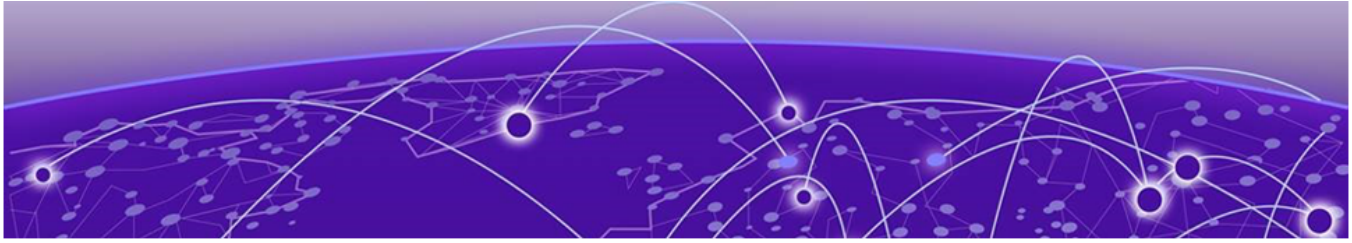


Table of Contents



Abstract

The *Extreme OS ONE Switching and Routing QoS Configuration Guide* version 22.2.3.0 provides technical procedures for implementing Quality of Service (QoS) policies. Designed for IT professionals and network engineers, the guide addresses deployment and management of QoS in complex, multi-platform networks.

Preface

Read the following topics to learn about:

- The meanings of text formats used in this document.
- Where you can find additional information and help.
- How to reach us with questions and comments.

Text Conventions

Unless otherwise noted, information in this document applies to all supported environments for the products in question. Exceptions, like command keywords associated with a specific software version, are identified in the text.

When a feature, function, or operation pertains to a specific hardware product, the product name is used. When features, functions, and operations are the same across an entire product family, such as Extreme Networks switches, the product is referred to as *the switch*.

Table 1: Notes and warnings

Icon	Notice type	Alerts you to...
	Tip	Helpful tips and notices for using the product
	Note	Useful information or instructions
	Important	Important features or instructions
	Caution	Risk of personal injury, system damage, or loss of data
	Warning	Risk of severe personal injury

Table 2: Text

Convention	Description
`screen displays`	This typeface indicates command syntax, or represents information as it is displayed on the screen.
The words *enter* and *type*	When you see the word *enter* in this guide, you must type something, and then press the Return or Enter key. Do not press the Return or Enter key when an instruction simply says *type*.
Key names	Key names are written in boldface, for example **Ctrl** or **Esc**. If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press **Ctrl**+**Alt**+**Del**
Words in italicized type	Italics emphasize a point or denote new terms at the place where they are defined in the text. Italics are also used when referring to publication titles.
NEW!	New information. In a PDF, this is searchable text.

Table 3: Command syntax

Convention	Description
bold text	Bold text indicates command names, keywords, and command options.
italic text	Italic text indicates variable content.
`[ ]`	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.
`{ x | y | z }`	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x | **y**	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, such as passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, *member*[*member*...].
\	In command examples, the backslash indicates a “soft” line break. When a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Documentation and Training

Find Extreme Networks product information at the following locations:

Current Product Documentation
Release Notes
Hardware and Software Compatibility for Extreme Networks products
Extreme Optics Compatibility
Other Resources such as articles, white papers, and case studies

Open Source Declarations

Some software files have been licensed under certain open source licenses. Information is available on the Open Source Declaration page.

Training

Extreme Networks offers product training courses, both online and in person, as well as specialized certifications. For details, visit the Extreme Networks Training page.

Help and Support

If you require assistance, contact Extreme Networks using one of the following methods:

Extreme Portal

Search the GTAC (Global Technical Assistance Center) knowledge base; manage support cases and service contracts; download software; and obtain product licensing, training, and certifications.

The Hub

A forum for Extreme Networks customers to connect with one another, answer questions, and share ideas and feedback. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.

Call GTAC

For immediate support: (800) 998 2408 (toll-free in U.S. and Canada) or 1 (408) 579 2800. For the support phone number in your country, visit www.extremenetworks.com/support/contact.

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number, or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any actions already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

Subscribe to Product Announcements

You can subscribe to email notifications for product and software release announcements, Field Notices, and Vulnerability Notices.

1. Go to The Hub.
2. In the list of categories, expand the **Product Announcements** list.
3. Select a product for which you would like to receive notifications.
4. Select **Subscribe**.
5. To select additional products, return to the **Product Announcements** list and repeat steps 3 and 4.

You can modify your product selections or unsubscribe at any time.

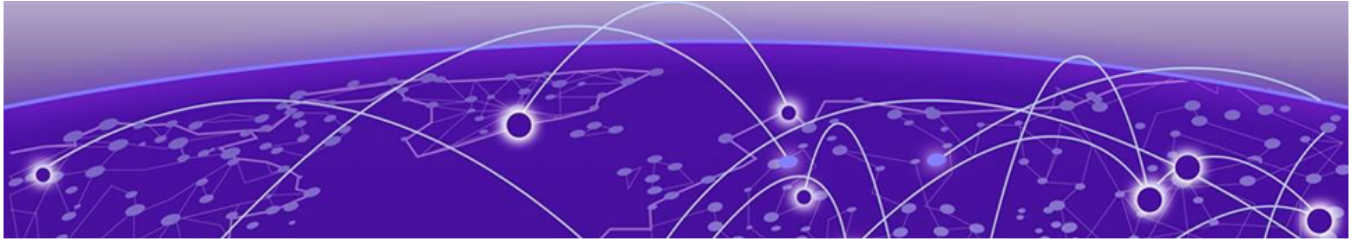
Send Feedback

The User Enablement team at Extreme Networks has made every effort to ensure that this document is accurate, complete, and easy to use. We strive to improve our documentation to help you in your work, so we want to hear from you. We welcome all feedback, but we especially want to know about:

- Content errors, or confusing or conflicting information.
- Improvements that would help you find relevant information.
- Broken links or usability issues.

To send feedback, email us at Product-Documentation@extremenetworks.com.

Provide as much detail as possible including the publication title, topic heading, and page number (if applicable), along with your comments and suggestions for improvement.



About This Document

What's New in This Document

The following table describes information added to this guide for Extreme OS ONE Switching and Routing, release 22.2.3.0.

Feature	Description	Link
Data Center Bridging (DCB)	DCB is a collection of IEEE standards designed to make Ethernet suitable for lossless, low-latency communication in data center environments. DCB mitigates packet loss that occurs in Ethernet during congestion, while data center applications such as Remote Direct Memory Access (RDMA) and storage require guaranteed delivery.	Data Center Bridging on page 40
Priority-based Flow Control (PFC)	Priority Flow Control (PFC) enables per-priority lossless transport by pausing only congested traffic classes while allowing other classes to continue forwarding, unlike standard link-level PAUSE. The feature provides independent enable/disable controls for each of the 8 priority levels, directional settings (TX/RX), per-priority headroom management, and watchdog-based deadlock detection.	Monitor QoS Queue Performance on page 37

For additional information, see the *Extreme OS ONE Switching and Routing Release Notes*.

Supported Platforms

Extreme OS ONE Switching and Routing 22.2.3.0 and later releases support Extreme 8520, Extreme 8720, Extreme 8730, and Extreme 8820 hardware platforms.

Note
Although many software and hardware configurations are tested and supported for this release, all possible configurations and scenarios are beyond this document's scope.

For information about other releases, see the documentation for those releases.

Fabric QoS

This chapter explains how to configure and manage Quality of Service (QoS) for both access and fabric traffic in your Extreme Networks environment. The Fabric QoS feature has comprehensive traffic classification, marking, and prioritization capabilities for both traditional access traffic and Virtual Extensible LAN (VxLAN) fabric traffic.

This feature supports QoS configuration for Multi-Chassis Link Aggregation Group (MLAG) tunnels using the same NVO level configuration. MLAG tunnel QoS follows the same uniform and pipe mode behaviors as standard VxLAN tunnels.

Fabric QoS Overview

Use this feature to perform the following tasks:

- Control traffic prioritization based on packet content.
- Manage VxLAN tunnel QoS behavior.
- Ensure proper traffic class derivation and packet remarking.

Note
QoS actions such as traffic class, DSCP, PCP, and drop precedence modification can also be applied through Access Control Lists (ACLs). See the *Extreme OS ONE Switching and Routing Security Configuration Guide* for details.

Key Capabilities

The Fabric QoS implementation delivers three core functionalities.

Internal Traffic Class Derivation

The internal traffic class (TC) derivation feature delivers the following core functionality:

- Derives internal traffic classes and Drop Precedence (DP) from packet headers automatically.
- Supports both Layer 2 (CoS or PCP) and Layer 3 (DSCP) classification.
- Has flexible mapping options for different traffic types.

Egress Packet Remarking

The egress packet remarking feature delivers the following core functionality:

- Modifies VLAN Priority Code Point (PCP) values on outgoing packets.
- Updates IP Differentiated Services Code Point (DSCP) values.
- Maintains QoS markings consistent with your network policies.

VxLAN Tunnel QoS

The VxLAN tunnel QoS feature delivers the following core functionality:

- Controls QoS behavior during VxLAN encapsulation and decapsulation.
- Supports both uniform and pipe tunnel modes.
- Manages outer header DSCP marking for fabric traffic.

Access QoS and Fabric QoS Limitations

This feature has the following platform related limitations, feature related limitations, and scope.

Access QoS and Fabric QoS Platform Limitations

Access QoS (which is the foundation for fabric QoS) and fabric QoS have the following platform related limitations.

Extreme 8520 and Extreme 8720

The Extreme 8520 and Extreme 8720 platforms have the following limitations:

- Egress Layer 2 remarking is always enabled in hardware.
- DSCP remarking (in egress) in switched IP packets must be configured only at the interface level, not at the logical interface (subinterface) level.
- Trust DSCP without Trust CoS on a Layer 2 logical interface use the hardware default map (traffic class = PCP) for Layer 2 tagged packets without IP headers.
- Tunnel decapsulation traffic class derivation is applied at the physical interface level, so any routed packets on this interface have traffic classes derived using this map.

- After tunnel decapsulation, if a packet undergoes routing, then traffic class derivation is done from the ingress QoS map applied on the ingress Layer 3 interface on the tunnel terminating node.
- Untagged routed traffic with Trust DSCP on untagged logical interfaces without a user configured map derive the traffic class as 1 instead of basing it on IP packet DSCP. The workaround enables Trust DSCP on a Layer 2 logical interface.

Extreme 8730TD4

The Extreme 8730 platform has the following limitations:

- The following table shows how classes are mapped (because the hardware is limited to only four multicast queues) for unknown unicast and multicast traffic.

Traffic Class	Queue Number
0	0
1	0
2	1
3	1
4	2
5	2
6	3
7	3

- Updating the default traffic class value overwrites the CoS 0 (default or best effort) to traffic class mapping in hardware for default mode bridge domain members.
- Egress Layer 2 remarking is always enabled in hardware.
- For IP packet forwarding in a bridge domain, the Layer 2 logical interface QoS configuration is not applied to packets. Instead, it uses the Virtual Ethernet (VE) interface QoS configuration (if the VE interface is present).

Extreme 8820

The Extreme 8820 platform has the following limitations:

- Broadcast, unknown unicast, and multicast (BUM) traffic shows per queue statistics on source ports.
- Egress packet remarking is not supported because of hardware limitations.
- Shared user QoS maps affect all associated interfaces.
- User configuration of tunnel decapsulation mode is not supported.
- After tunnel decapsulation, if a packet undergoes routing, then the traffic class derivation is done from the ingress QoS map applied on the ingress Layer 3 interface on the tunnel terminating node.

Feature Limitations

Fabric QoS does not support the following features:

- User configurable maps for fabric (VxLAN) QoS.
- Policers, schedulers, and rate limiters.

Other Limitations

Access QoS and fabric QoS have the following additional limitations:

- (Non Extreme 8730 platforms only) Configuration for egress packet remarking is not supported for VLAN mode bridge domains. For logical interfaces (LIFs) associated to a VLAN mode bridge domain, a shared LIF is created internally. So specific LIF level behavior is shared across such LIFs on a port and cannot be controlled individually per LIF.
- To check the functionality of ingress maps on a VLAN mode bridge domain, configure QoS maps under the Ethernet or port channel interface (this loops through all LIFs under this interface and apply the same QoS configuration as that of the Ethernet or port channel interface to the LIFs).
- Any QoS configuration on VLAN mode bridge domain LIFs is ignored and is replayed if the same LIF is moved to a default mode bridge domain.
- Trust and remark DSCP settings on a Layer 2 LIF is not effective until a user map is configured and attached to the Layer 2 LIF.
- If any specific QoS configuration exists under a specific LIF that belongs to an Ethernet or port channel interface, then the QoS map applied on the Ethernet or port channel interface is not applied to this LIF until the specific configuration is deleted from the LIF.
- When QoS configuration is deleted on a LIF, configurations on Ethernet or port channel interfaces are applied to the LIF.
- (Extreme 8730 and Extreme 8820 platforms only) User configuration of tunnel decapsulation mode is not supported.

Scope

The fabric QoS implementation in this release focuses on per hop behavior (PHB) including:

- Packet classification and marking.
- Traffic class and drop precedence derivation.
- Basic remarking functionality.

Traffic Flow Processing

The system processes packets through these stages:

1. Ingress classification: Analyzes incoming packets and assigns internal traffic classes and drop precedence.
2. Internal processing: Routes or switches packets while preserving QoS information.
3. Egress processing: Applies remarking policies and queue assignments before transmission.
4. Tunnel processing: Handles VxLAN encapsulation and decapsulation with appropriate QoS treatment.

Prerequisites

Before configuring fabric QoS, make sure that you have:

- Administrative access to the switch management interface.
- An understanding of your network's QoS requirements.
- Existing VLAN and interface configurations (for access QoS).
- VxLAN tunnel configurations (for fabric QoS).

Planning Considerations

Before configuring fabric QoS, make sure to:

- Identify traffic types requiring specific QoS treatment.
- Determine appropriate traffic class mappings for your environment.
- Plan DSCP and CoS marking strategies.
- Consider bandwidth and latency requirements for different traffic classes.

Best Practices

Configuration Management

Use the following practices to manage your fabric QoS configuration:

- Document your QoS policies and mappings.
- Test configurations in a lab environment before deployment.
- Use consistent naming conventions for maps and policies.
- Regularly review and validate QoS behavior.

Performance Optimization

Use the following practices to optimize the performance of your fabric QoS configuration:

- Monitor traffic patterns and adjust classifications as needed.
- Consider platform specific queue limitations.
- Balance QoS granularity with operational simplicity.
- Plan for growth in traffic volumes and types.

Troubleshooting Preparation

Use the following practices to prepare for troubleshooting your fabric QoS configuration:

- Establish baseline QoS statistics before making changes.
- Keep configuration backups for quick restoration.
- Document known platform limitations for your environment.
- Train operations staff on QoS verification procedures.

CLI Commands

To enable fabric QoS, use the CLI to configure access QoS and then to configure fabric QoS. Access QoS and fabric QoS are different but related levels of QoS enforcement.

Access QoS is the foundation for fabric QoS. Access QoS sets the traffic classification and markings, and fabric QoS depends on those markings to enforce the correct forwarding behavior.

If traffic is not classified or marked properly at the access level, the fabric cannot give it the intended priority. Access QoS defines and enforces the contract at the edge, and fabric QoS honors that contract across the network.

Configure Access QoS

Access QoS controls how the system classifies and marks traffic on traditional Layer 2 and Layer 3 interfaces.

Extreme OS ONE Switching and Routing supports ACL-based QoS configuration. For detailed information, see the *Extreme OS ONE Switching and Routing Security Configuration Guide* and the *Extreme OS ONE Switching and Routing Command Reference Guide*.

Create a Traffic Class Map

Traffic class maps define how incoming packet markings translate to internal traffic classes.

You can define a nondefault drop precedence in a CoS-to-traffic-class map and in a DSCP-to-traffic-class map. This defines the likelihood of the device dropping a packet during congestion to discard less critical traffic selectively.

Drop precedence is a subclassification in a traffic class to manage queue congestion. When buffers are full, the device drops packets with higher precedence first, even if they occupy the same traffic class or queue as others.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Enter QoS configuration mode.

```
device(config)# qos
```

3. Configure a QoS map for packet classification.

```
device(config-qos)# traffic-class-map tcmap1
```

4. Define the CoS to traffic class entries in the QoS traffic class map.

```
device(config-qos-traffic-class-map-tcmap1)# cos 2 traffic-class 6 drop-precedence 2
```

   Specifying a drop precedence value is optional. Use an integer of 0 (lowest drop probability) to 2 (highest). The default value is 0 if you do not specify this keyword.

5. Define the DSCP to traffic class entries in the QoS traffic class map.

```
device(config-qos-traffic-class-map-tcmap1)# dscp 24 traffic-class 6 drop-precedence 2
```

   Specifying a drop precedence value is optional. Use an integer of 0 (lowest drop probability) to 2 (highest). The default value is 0 if you do not specify this keyword.

6. (Optional) Verify the traffic class map configuration.

```
device(config-qos-traffic-class-map-tcmap1)# do show running-config qos

qos
  traffic-class-map tcmap1
    cos 2 traffic-class 6 drop-precedence 2
    dscp 24 traffic-class 6 drop-precedence 2
  !
!
device#
```

- The following example creates a traffic class map named tcmap1:

```
device# configure terminal
device(config)# qos
device(config-qos)# traffic-class-map tcmap1
device(config-qos-traffic-class-map-tcmap1)# cos 2 traffic-class 6 drop-precedence 2
device(config-qos-traffic-class-map-tcmap1)# dscp 24 traffic-class 6 drop-precedence 2
device(config-qos-traffic-class-map-tcmap1)#
```

- The following example shows how to display a QoS configuration. In this example, two traffic class maps (tcmap1 and tcl3map) are configured:

```
device(config-qos-traffic-class-map-tcmap1)# do show running-config qos

qos
  traffic-class-map tcmap1
    cos 2 traffic-class 6 drop-precedence 2
    dscp 24 traffic-class 6 drop-precedence 2
  !
  traffic-class-map tcl3map
    dscp 10 traffic-class 7
```

```
  !
  interface ethernet 0/3
    input
      trust dscp
      traffic-class-map tcl3map
    !
  !
  interface ethernet 0/1 subinterface vlan 100
    input
      trust cos
      traffic-class-map tcmap1
    !
  !
!
device#
```

- The following example displays a default traffic class map that is configured with drop precedence:

```
device# show qos traffic-class-map default

 Default Traffic Class for map default: 1
 COS-to-Traffic-Class Map: default
   COS            : 0 1 2 3 4 5 6 7
   ---------------------------------------
   Traffic-Class        : 1 0 2 3 4 5 6 7
   Drop-Precedence      : 0 0 0 0 0 0 0 0

DSCP-to-Traffic-Class Map: default (DSCP = d1d2)
  d1: d2 0   1    2    3    4    5    6    7    8    9

-------------------------------------------------------------------------------------------
-----
  0:     1/0 1/0  1/0  1/0  1/0  1/0  1/0  1/0  0/0  0/0
  1:     0/0 0/0  0/0  0/0  0/0  0/0  2/0  2/0  2/0  2/0
  2:     2/0 2/0  2/0  2/0  3/0  3/0  3/0  3/0  3/0  3/0
  3:     3/0 3/0  4/0  4/0  4/0  4/0  4/0  4/0  4/0  4/0
  4:     5/0 5/0  5/0  5/0  5/0  5/0  5/0  5/0  6/0  6/0
  5:     6/0 6/0  6/0  6/0  6/0  6/0  7/0  7/0  7/0  7/0
  6:     7/0 7/0  7/0  7/0
device#
```

Create a Remark Map

Remark maps specify how internal traffic classes map to outgoing packet markings.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Enter QoS configuration mode.

```
device(config)# qos
```

3. Configure a map for remarking the QoS markings on packets.

```
device(config-qos)# remark-map rmrkmap1
```

4. Define the traffic class to CoS entries for remarking.

```
device(config-qos-remark-map-rmrkmap1)# traffic-class 6 drop-precedence 2 cos 2
```

Specifying a drop precedence value is optional. Use an integer of 0 (lowest drop probability) to 2 (highest). The default value is 0 if you do not specify this keyword.

5. Define the traffic class to DSCP entries for remarking

```
device(config-qos-remark-map-rmrkmap1)# traffic-class 6 drop-precedence 2 dscp 1
```

   Specifying a drop precedence value is optional. Use an integer of 0 (lowest drop probability) to 2 (highest). The default value is 0 if you do not specify this keyword.

6. (Optional) Verify the remark map configuration.

```
device(config-qos-remark-map-rmrkmap1)# do show running-config qos

qos
  remark-map rmrkmap1
    traffic-class 6 drop-precedence 2 cos 2
    traffic-class 6 drop-precedence 2 dscp 1
  !
!
device#
```

- The following example creates a remark map named rmrkmap1 with drop-precedence:

```
device# configure terminal
device(config)# qos
device(config-qos)# remark-map rmrkmap1
device(config-qos-remark-map-rmrkmap1)# traffic-class 6 drop-precedence 2 cos 2
device(config-qos-remark-map-rmrkmap1)# traffic-class 6 drop-precedence 2 dscp 1
device(config-qos-remark-map-rmrkmap1)#
```

- The following example shows how to display a QoS configuration. In this example, a remark map named rmrkmap1 is configured with drop-precedence:

```
device(config-qos-remark-map-rmrkmap1)# do show running-config qos

qos
remark-map rmrkmap1
    traffic-class 6 drop-precedence 2 cos 2
    traffic-class 6 drop-precedence 2 dscp 1
  !
  interface ethernet 0/2
    output
      remark cos
      remark-map rmrkmap1
    !
  !
!
device#
```

- The following example displays a default remark map that is configured with drop precedence:

```
device# show qos remark-map

  Traffic-Class-to-COS Map: default
    Traffic-Class : 0 1 2 3 4 5 6 7
    ---------------------------------------------------
    COS(DP=0)     : 1 0 2 3 4 5 6 7
    COS(DP=1)     : 1 0 2 3 4 5 6 7
    COS(DP=2)     : 1 0 2 3 4 5 6 7

  Traffic-Class-to-DSCP Map: default
    Traffic-Class : 0   1    2    3    4    5    6    7
    ---------------------------------------------------
    DSCP(DP=0)    : 8   0    16   24   32   40   48   56
    DSCP(DP=1)    : 8   0    16   24   32   40   48   56
    DSCP(DP=2)    : 8   0    16   24   32   40   48   56
device#
```

Apply a Traffic Class Map to an Interface

You can attach a QoS traffic class map to an interface under the input or output directions.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Enter QoS configuration mode.

```
device(config)# qos
```

3. Access the interface or subinterface to which you are assigning the traffic class map.

```
device(config-qos)# interface ethernet 0/3
```

4. Enter the mode for configuring access QoS in the input direction or the output direction on an interface.

```
device(config-qos-if-eth-0/3)# input
```

5. Enable the interface to trust the QoS CoS or QoS DSCP markings (using the **cos** or **dscp** keywords respectively) on ingress (incoming) packets for traffic class mapping. You can enable the interface to trust both marking types simultaneously.

```
device(config-qos-if-eth-0/3-input)# trust dscp
```

6. Attach the traffic-class map to the interface.

```
device(config-qos-if-eth-0/3-input)# traffic-class-map tcl3map
```

7. (Optional) Verify the interface configuration.

```
device(config-qos-if-eth-0/3-input)# do show running-config qos interface ethernet 0/3

qos
  interface ethernet 0/3
    input
      trust dscp
      traffic-class-map tcl3map
    !
  !
!
device#
```

The following example attaches a traffic class map named tcl3map to Ethernet interface 0/3.

```
device# configure terminal
device(config)# qos
device(config-qos)# interface ethernet 0/3
device(config-qos-if-eth-0/3)# input
device(config-qos-if-eth-0/3-input)# trust dscp
device(config-qos-if-eth-0/3-input)# traffic-class-map tcl3map
device(config-qos-if-eth-0/3-input)#
```

The following example shows how to display a QoS configuration. In this example, a traffic class map named tcl3map is attached to Ethernet interface 0/3 in the input direction.

```
device(config-qos-if-eth-0/3-input)# do show running-config qos

qos
  traffic-class-map tcmap1
    cos 2 traffic-class 6 drop-precedence 2
    dscp 24 traffic-class 6 drop-precedence 2
```

```
  !
  traffic-class-map tcl3map
    dscp 10 traffic-class 7
  !
  remark-map rmrkmap1
    traffic-class 6 drop-precedence 2 cos 2
    traffic-class 6 drop-precedence 2 dscp 1
  !
  interface ethernet 0/2
    output
      remark cos
      remark-map rmrkmap1
    !
  !
  interface ethernet 0/3
    input
      trust dscp
      traffic-class-map tcl3map
    !
  !
  interface ethernet 0/1 subinterface vlan 100
    input
      trust cos
      traffic-class-map tcmap1
    !
  !
!
device#
```

Apply a Remark Map to an Interface

You can attach a QoS map to an interface for remarking the QoS markings on egress (outgoing) packets.

1. From privileged EXEC mode, access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter QoS configuration mode.

   ```
   device(config)# qos
   ```

3. Access the interface (or subinterface) to which you are assigning the remark map.

   ```
   device(config-qos)# interface ethernet 0/2
   ```

4. Enter the mode for configuring access QoS in the output direction or the input direction on an interface.

   ```
   device(config-qos-if-eth-0/2)# output
   ```

5. Enable the interface to remark the packet CoS or DSCP values (using the **cos** or **dscp** keyword respectively) on egress (outgoing) packets based on remark mapping.

   ```
   device(config-qos-if-eth-0/2-output)# remark cos
   ```

6. Attach the remark map to the interface.

   ```
   device(config-qos-if-eth-0/2-output)# remark-map rmrkmap1
   ```

7. (Optional) Verify the interface configuration.

   ```
   device(config-qos-if-eth-0/2-output)# do show running-config qos interface ethernet 0/2

   qos
     interface ethernet 0/2
       output
         remark cos
   ```

```
      remark-map rmrkmap1
    !
  !
!
device#
```

The following example attaches a remark map named rmrkmap1 to Ethernet interface 0/2.

```
device# configure terminal
device(config)# qos
device(config-qos)# interface ethernet 0/2
device(config-qos-if-eth-0/2)# output
device(config-qos-if-eth-0/2-output)# remark cos
device(config-qos-if-eth-0/2-output)# remark-map rmrkmap1
device(config-qos-if-eth-0/2-output)#
```

The following example shows how to display a QoS configuration. In this example, a remark map named rmrkmap1 is attached to Ethernet interface 0/2 in the output direction.

```
device(config-qos-if-eth-0/2-output)# do show running-config qos

qos
  traffic-class-map tcmap1
    cos 2 traffic-class 6 drop-precedence 2
    dscp 24 traffic-class 6 drop-precedence 2
  !
  traffic-class-map tcl3map
    dscp 10 traffic-class 7
  !
  remark-map rmrkmap1
    traffic-class 6 drop-precedence 2 cos 2
    traffic-class 6 drop-precedence 2 dscp 1
  !
  interface ethernet 0/2
    output
      remark cos
      remark-map rmrkmap1
    !
  !
  interface ethernet 0/3
    input
      trust dscp
      traffic-class-map tcl3map
    !
  !
  interface ethernet 0/1 subinterface vlan 100
    input
      trust cos
      traffic-class-map tcmap1
    !
  !
!
device#
```

gNMI Commands

Use the gNMI commands for configuring a traffic class map and a remark map.

Traffic-Class Map Creation

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/maps/traffic-class-maps/
traffic-class-map[name=tcmap1]/config/name:::string:::tcmap1"
```

Traffic-Class Map Deletion

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/maps/traffic-class-maps/
traffic-class-map[name=tcmap1]"
```

Traffic-Class Map Default-Traffic-Class Configuration

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/maps/traffic-class-maps/
traffic-class-map[name=csk]/config/default-traffic-class:::uint:::5"
```

Traffic-Class Map Default-Traffic-Class Deletion

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/maps/traffic-class-maps/
traffic-class-map[name=csk]/config/default-traffic-class"
```

Traffic-Class Map COS-TC Configuration

- Without drop-precedence

  ```
  gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
  os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/maps/traffic-class-
  maps/traffic-class-map[name=tcmap1]/cos-maps/cos-map[cos=4]/config/cos:::uint:::4"
  --update "/qos/maps/traffic-class-maps/traffic-class-map[name=tcmap1]/cos-maps/cos-
  map[cos=4]/config/traffic-class:::uint:::1"
  ```

- With drop-precedence

  ```
  gnmic -a DUT_IP:PORT_NUM -u admin -p rocks  --tls-ca /
  home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set
  --update "/qos/maps/traffic-class-maps/traffic-class-map[name=tcmap1]/cos-maps/cos-
  map[cos=4]/config/cos:::uint:::4" --update "/qos/maps/traffic-class-maps/traffic-
  class-map[name=tcmap1]/cos-maps/cos-map[cos=4]/config/traffic-class:::uint:::1" --
  update "/qos/maps/traffic-class-maps/traffic-class-map[name=tcmap1]/cos-maps/cos-
  map[cos=4]/config/drop-precedence:::uint:::2"
  ```

Traffic-Class Map COS-TC Deletion

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/maps/traffic-class-maps/
traffic-class-map[name=tcmap1]/cos-maps/cos-map[cos=4]"
```

Traffic-Class Map DSCP-TC Configuration

- Without drop-precedence

  ```
  gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
  os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/maps/traffic-class-maps/
  traffic-class-map[name=tcmap1]/dscp-maps/dscp-map[dscp=31]/config/dscp:::uint:::31"
  --update "/qos/maps/traffic-class-maps/traffic-class-map[name=tcmap1]/dscp-maps/dscp-
  map[dscp=31]/config/traffic-class:::uint:::6"
  ```

- With drop-precedence

  ```
  gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /
  home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --
  update "/qos/maps/traffic-class-maps/traffic-class-map[name=tcmap1]/dscp-maps/dscp-
  map[dscp=31]/config/dscp:::uint:::31" --update "/qos/maps/traffic-class-maps/traffic-
  ```

```
class-map[name=tcmap1]/dscp-maps/dscp-map[dscp=31]/config/traffic-class:::uint:::6"
--update "/qos/maps/traffic-class-maps/traffic-class-map[name=tcmap1]/dscp-maps/dscp-
map[dscp=31]/config/drop-precedence:::uint:::1"
```

Traffic-Class Map DSCP-TC Deletion

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/maps/traffic-class-maps/
traffic-class-map[name=tcmap1]/dscp-maps/dscp-map[dscp=31]"
```

Remark Map Creation

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/maps/remark-maps/remark-
map[name=rmrkmap1]/config/name:::string:::rmrkmap1"
```

Remark Map Deletion

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/maps/remark-maps/remark-
map[name=rmrkmap1]"
```

Remark Map TC-COS Configuration

- Without drop-precedence

  Drop-precedence is set to 0 in the key.

  ```
  gnmic -a DUT_IP:PORT_NUM -u admin -p rocks
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
  set --update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/
  traffic-class-map[traffic-class=4][drop-precedence=0]/config/traffic-class:::uint:::4"
  --update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/traffic-
  class-map[traffic-class=4][drop-precedence=0]/config/cos:::uint:::1"
  ```

- With drop-precedence

  ```
  gnmic -a DUT_IP:PORT_NUM -u admin -p rocks
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
  set --update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/
  traffic-class-map[traffic-class=4][drop-precedence=1]/config/traffic-class:::uint:::4"
  --update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/traffic-
  class-map[traffic-class=4][drop-precedence=1]/config/drop-precedence:::uint:::1" --
  update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/traffic-
  class-map[traffic-class=4][drop-precedence=1]/config/cos:::uint:::1"
  ```

Remark Map TC-COS Deletion

Drop-precedence is set to 0 in the key.

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /
home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --
delete "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/traffic-class-
map[traffic-class=4][drop-precedence=0]/config/cos"
```

Remark Map TC-DSCP Configuration

- Without drop-precedence

  Drop-precedence is set to 0 in the key.

  ```
  gnmic -a DUT_IP:PORT_NUM -u admin -p rocks
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
  set --update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/
  ```

```
traffic-class-map[traffic-class=4][drop-precedence=0]/config/traffic-class:::uint:::4"
--update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/traffic-
class-map[traffic-class=4][drop-precedence=0]/config/dscp:::uint:::33"
```

- With drop-precedence

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks
--tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
set --update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/
traffic-class-map[traffic-class=4][drop-precedence=2]/config/traffic-class:::uint:::4"
--update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/traffic-
class-map[traffic-class=4][drop-precedence=2]/config/drop-precedence:::uint:::2"  --
update "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/traffic-
class-map[traffic-class=4][drop-precedence=2]/config/dscp:::uint:::33"
```

Remark Map TC-DSCP Deletion

Drop-precedence is set to 0 in the key.

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /
home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --
delete "/qos/maps/remark-maps/remark-map[name=rmrkmap1]/traffic-class-maps/traffic-class-
map[traffic-class=4][drop-precedence=0]/config/dscp"
```

Configure Fabric QoS

Fabric QoS manages QoS behavior for VxLAN tunnel traffic to control how the system handles encapsulation and decapsulation. Fabric QoS in Extreme OS ONE supports uniform tunnel mode and pipe tunnel mode.

Uniform Tunnel Mode

In uniform mode, the fabric honors and propagates host markings fabric-wide. Uniform mode has the following functionality:

- Copies QoS markings between inner and outer headers.
- Maintains end-to-end QoS visibility.
- Modifies customer packet markings based on fabric treatment.

Pipe Tunnel Mode

In pipe mode, the fabric uses its own QoS policies, independent of the original packet's markings. The fabric applies its own markings internally, and host markings pass through untouched.

Pipe mode has the following functionality:

- Isolates customer traffic QoS from fabric QoS.
- Preserves original customer packet markings.
- Uses fixed or mapped values for outer header markings.

Configure Tunnel QoS on an NVO Endpoint

You configure QoS behavior at the Network Virtualization Overlay (NVO) level, which applies to all tunnels under that NVO.

1. From privileged EXEC mode, access global configuration mode.
```
device# configure terminal
```
2. Enter QoS configuration mode.
```
device(config)# qos
```
3. Associate an NVO instance with the QoS node.
```
device(config-qos)# nvo nvo1
```
4. Enter QoS NVO tunnel configuration mode.
```
device(config-qos-nvo1)# tunnel-mode
```
5. Set the fabric QoS tunnel encapsulation mode for the NVO endpoint (using the **encapsulation pipe uniform** or **encapsulation pipe** [**dscp** [*dscp_value*]] commands).
```
device(config-qos-nvo-nvo1-tunnel-mode)# encapsulation pipe dscp 44
```
6. Set the fabric QoS tunnel decapsulation mode for the NVO endpoint (using the **uniform** or **pipe** keywords).
```
device(config-qos-nvo-nvo1-tunnel-mode)# decapsulation uniform
```

Note
Tunnel decapsulation on the Extreme 8730 and Extreme 8820 platforms supports only the default mode (pipe) and is not configurable.

7. (Optional) Verify the fabric tunnel QoS configuration.
```
device(config-qos-nvo-nvo1-tunnel-mode)# do show qos nvo

nvo nvo1
  tunnel-mode:
    encapsulation pipe dscp 44
    decapsulation uniform
  !
!
device#
```

The following example configures fabric tunnel QoS encapsulation and decapsulation for an NVO named nvo1.
```
device# configure terminal
device(config)# qos
device(config-qos)# nvo nvo1
device(config-qos-nvo1)# tunnel-mode
device(config-qos-nvo-nvo1-tunnel-mode)# encapsulation pipe dscp 44
device(config-qos-nvo-nvo1-tunnel-mode)# decapsulation uniform
device(config-qos-nvo-nvo1-tunnel-mode)#
```

The following example shows how to display a QoS configuration. In this example, fabric tunnel QoS encapsulation and decapsulation for an NVO named nvo1 are configured.
```
device(config-qos-nvo-nvo1-tunnel-mode)# do show running-config qos

qos
  traffic-class-map tcmap1
```

```
    cos 2 traffic-class 6 drop-precedence 2
    dscp 24 traffic-class 6 drop-precedence 2
  !
  traffic-class-map tcl3map
    dscp 10 traffic-class 7
  !
  remark-map rmrkmap1
    traffic-class 6 drop-precedence 2 cos 2
    traffic-class 6 drop-precedence 2 dscp 1
  !
  interface ethernet 0/2
    output
      remark cos
      remark-map rmrkmap1
    !
  !
  interface ethernet 0/3
    input
      trust dscp
      traffic-class-map tcl3map
    !
  !
  interface ethernet 0/1 subinterface vlan 100
    input
      trust cos
      traffic-class-map tcmap1
    !
  !
  nvo nvo1
    tunnel-mode
      encapsulation pipe dscp 44
      decapsulation uniform
    !
  !
!
device#
```

Display Mappings in a Traffic Class Map

To display the CoS to traffic-class and DSCP to traffic-class mappings in a traffic class map, use the **`show qos traffic-class-map`** command.

The following example displays the CoS to traffic class and DSCP to traffic class mappings in a traffic class map named ing_l2_map on the device.

```
device# show qos traffic-class-map ing_l2_map

 Default Traffic Class for map ing_l2_map: 3
  COS-to-Traffic-Class Map: ing_l2_map
    COS                 : 0  1  2  3  4  5  6  7
    ------------------------------------------
    Traffic-Class       : 1  3  4  3  4  5  6  7
    Drop-Precedence     : 0  0  2  0  0  0  0  0

  DSCP-to-Traffic-Class Map: ing_l2_map (x/y: TC = x, DP = y, DSCP = d1d2)
    d1 : d2 0   1       2       3       4       5       6       7       8       9
    ------------------------------------------------------------------------------

    0 :     1/0 2/0     4/2     1/0     1/0     1/0     1/0     1/0     0/0     0/0
    1 :     0/0 0/0     0/0     0/0     0/0     0/0     2/0     2/0     2/0     2/0
    2 :     2/0 2/0     2/0     2/0     3/0     3/0     3/0     3/0     3/0     3/0
```

```
    3 :     3/0 3/0     4/0     4/0     4/0     4/0     4/0     4/0     4/0     4/0
    4 :     5/0 5/0     5/0     5/0     5/0     5/0     5/0     5/0     6/0     6/0
    5 :     6/0 6/0     6/0     6/0     6/0     6/0     7/0     7/0     7/0     7/0
    6 :     7/0 7/0     7/0     7/0
device#
```

The following example displays all CoS to traffic class and DSCP to traffic class mappings on the device.

```
device# show qos traffic-class-map

  Default Traffic Class for map default: 1
  COS-to-Traffic-Class Map: default
    COS                 : 0  1  2  3  4  5  6  7
    ------------------------------------------
    Traffic-Class       : 1  0  2  3  4  5  6  7
    Drop-Precedence     : 0  0  0  0  0  0  0  0

  DSCP-to-Traffic-Class Map: default (x/y: TC = x, DP = y, DSCP = d1d2)
    d1 : d2 0     1         2         3         4         5         6         7
8         9

-------------------------------------------------------------------------------------------------
-------

    0 :     0/0   0/0       0/0       0/0       0/0       0/0       0/0       0/0
1/0       1/0
    1 :     1/0   1/0       1/0       1/0       1/0       1/0       2/0       2/0
2/0       2/0
    2 :     2/0   2/0       2/0       2/0       3/0       3/0       3/0       3/0
3/0       3/0
    3 :     3/0   3/0       4/0       4/0       4/0       4/0       4/0       4/0
4/0       4/0
    4 :     5/0   5/0       5/0       5/0       5/0       5/0       5/0       5/0
6/0       6/0
    5 :     6/0   6/0       6/0       6/0       6/0       6/0       7/0       7/0
7/0       7/0
    6 :     7/0   7/0       7/0       7/0

 Default Traffic Class for map ing_l2_map: 3
  COS-to-Traffic-Class Map: ing_l2_map
    COS                 : 0  1  2  3  4  5  6  7
    ------------------------------------------
    Traffic-Class       : 1  3  4  3  4  5  6  7
    Drop-Precedence     : 0  0  2  0  0  0  0  0

  DSCP-to-Traffic-Class Map: ing_l2_map (x/y: TC = x, DP = y, DSCP = d1d2)
    d1 : d2 0   1       2       3       4       5       6       7       8       9
    ---------------------------------------------------------------------------

    0 :     1/0 2/0     4/2     1/0     1/0     1/0     1/0     1/0     0/0     0/0
    1 :     0/0 0/0     0/0     0/0     0/0     0/0     2/0     2/0     2/0     2/0
    2 :     2/0 2/0     2/0     2/0     3/0     3/0     3/0     3/0     3/0     3/0
    3 :     3/0 3/0     4/0     4/0     4/0     4/0     4/0     4/0     4/0     4/0
    4 :     5/0 5/0     5/0     5/0     5/0     5/0     5/0     5/0     6/0     6/0
    5 :     6/0 6/0     6/0     6/0     6/0     6/0     7/0     7/0     7/0     7/0
    6 :     7/0 7/0     7/0     7/0
device#
```

Display Traffic Class Mappings for Remarking

To display the traffic class to COS and traffic class to DSCP mappings for remarking the QoS markings on packets, use the **show qos remark-map** command.

The following example displays the mappings within a QoS remark map named rmrkmap1 on the device.

```
device# show qos remark-map rmrkmap1

 Traffic-Class-to-COS Map: rmrkmap1
 Traffic-Class : 0 1 2 3 4 5 6 7
 ---------------------------------------------------
 COS(DP=0) : 1 0 2 3 4 5 6 7
 COS(DP=1) : 1 0 2 3 4 5 6 7
 COS(DP=2) : 1 0 2 3 4 5 6 7

 Traffic-Class-to-DSCP Map: rmrkmap1
 Traffic-Class : 0 1 2 3 4 5 6 7
 ---------------------------------------------------
 DSCP(DP=0) : 8 0 16 24 32 40 48 56
 DSCP(DP=1) : 8 0 16 24 32 40 48 56
 DSCP(DP=2) : 8 0 16 24 32 40 48 56
device#
```

The following example displays the mappings within all QoS remark maps on the device.

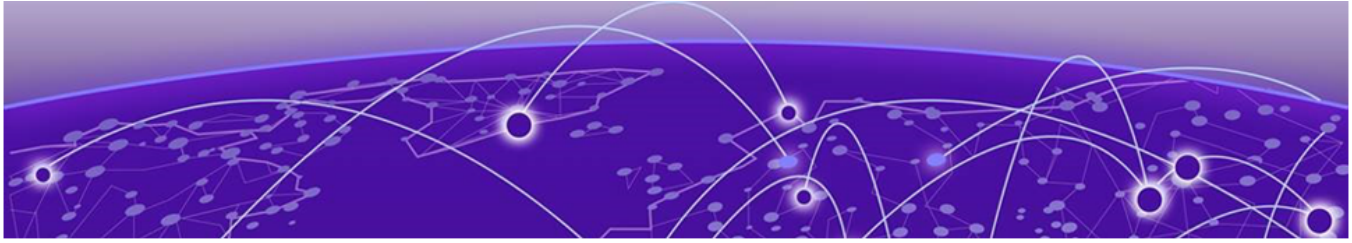
```
device# show qos remark-map

 Traffic-Class-to-COS Map: default
 Traffic-Class : 0 1 2 3 4 5 6 7
 ---------------------------------------------------
 COS(DP=0) : 1 0 2 3 4 5 6 7
 COS(DP=1) : 1 0 2 3 4 5 6 7
 COS(DP=2) : 1 0 2 3 4 5 6 7

 Traffic-Class-to-DSCP Map: default
 Traffic-Class : 0 1 2 3 4 5 6 7
 ---------------------------------------------------
 DSCP(DP=0) : 8 0 16 24 32 40 48 56
 DSCP(DP=1) : 8 0 16 24 32 40 48 56
 DSCP(DP=2) : 8 0 16 24 32 40 48 56

 Traffic-Class-to-COS Map: rmrkmap1
 Traffic-Class : 0 1 2 3 4 5 6 7
 ---------------------------------------------------
 COS(DP=0) : 1 0 2 3 4 5 6 7
 COS(DP=1) : 1 0 2 3 4 5 6 7
 COS(DP=2) : 1 0 2 3 4 5 6 7

 Traffic-Class-to-DSCP Map: rmrkmap1
 Traffic-Class : 0 1 2 3 4 5 6 7
 ---------------------------------------------------
 DSCP(DP=0) : 8 0 16 24 32 40 48 56
 DSCP(DP=1) : 8 0 16 24 32 40 48 56
 DSCP(DP=2) : 8 0 16 24 32 40 48 56
device#
```



Egress Port Scheduler

The following topics describe the egress port scheduling functionality in Extreme OS ONE Switching and Routing.

Egress Port Scheduler Overview

The Egress Port Scheduler provides essential Quality of Service (QoS) capabilities for your Extreme OS ONE network infrastructure by managing network traffic. By understanding the queue architecture, traffic mapping, and scheduling behaviors described in this guide, you can effectively manage network traffic priorities and ensure optimal performance for your applications.

This feature intelligently prioritizes and schedules packet transmission at egress ports, ensuring optimal network performance and traffic flow management. When packets arrive at the network switch, the system routes them to specific output ports and assigns them to priority queues. The Egress Port Scheduler determines which packets transmit first based on configured scheduling disciplines for precise control over traffic prioritization.

Key Benefits

The Egress Port Scheduler delivers several important advantages:

- Traffic Prioritization: The system ensures that critical traffic receives higher priority over less important data flows.
- Bandwidth Management: The system efficiently allocates available bandwidth across different traffic classes.

- Performance Optimization: The network maintains consistent performance even under heavy traffic loads.
- Hardware-Based Processing: The feature leverages dedicated hardware acceleration for maximum throughput

Queue Architecture

The system implements a hierarchical queue structure with the following characteristics.

Queue Distribution

Traffic is distributed among twelve total hardware queues that are split between unicast and multicast traffic types:

- Eight unicast queues that handle known unicast traffic (destinations with learned MAC addresses).
- Extreme 8730 platform only has four multicast queues that process broadcast, unknown unicast, and multicast (BUM) traffic. All other supported platforms have eight multicast queues.

Default Scheduling Modes

The scheduler operates using two distinct modes. These modes are deficit weighted round robin (DWRR) and strict priority (SP) as follows:

- Queues 0-5 use DWRR scheduling: This enables proportional bandwidth allocation based on configured weights. This ensures fair access to network resources.
- Queues 6-7 use strict priority scheduling: Higher priority traffic always transmits before lower priority traffic. This guarantees minimal latency for critical applications.

Default Traffic Class Mapping

The system maps traffic classes to queues according to the following configuration.

Table 4:

Traffic Class	Unicast Queue	Multicast Queue	Scheduling Mode	Weight
0	0	0	DWRR	5
1	1	0	DWRR	10
2	2	1	DWRR	15
3	3	1	DWRR	20
4	4	2	DWRR	25
5	5	2	DWRR	25 (highest priority traffic)

Table 4: (continued)

Traffic Class	Unicast Queue	Multicast Queue	Scheduling Mode	Weight
6	6	3	SP	None
7	7	3	SP	None

Architecture and Operation

The system implements a two-level hierarchical scheduler. The scheduler processes traffic through a four step process as described in this section.

Scheduler Hierarchy

The scheduler hierarchy consists of level 1 (L1) and level 0 (L0) nodes:

- For the L1 nodes, 12 L1 scheduler nodes are split into 8 unicast and 4 multicast queues. Each L1 node connects to specific traffic classes.
- For the L0 nodes, 8 L0 nodes apply weights and scheduling disciplines. These nodes aggregate traffic from L1 nodes before final port transmission.

Traffic Flow Process

The scheduler processes traffic through the following steps:

1. Classification: The system classifies incoming packets into traffic classes.
2. Queue Assignment: Packets route to appropriate unicast or multicast queues.
3. Scheduling Decision: The scheduler selects the next packet for transmission based on strict priority for queues 6–7 and deficit weighted round robin for queues 0–5.
4. Transmission: Selected packets transmit through the egress port.

Egress Port Scheduler Limitations

Be aware of the following egress port scheduler limitations.

Unsupported Features

The following features are not supported:

- Traffic Shaping and port shaping are not available.
- (Extreme 8820 platform only) Dynamic Scheduler Profiles: You cannot configure custom scheduler profiles.
- (Extreme 8820 platform only) Egress queue shaping

Hardware Constraints

Be aware of the following hardware constraints:

- Multicast Queue Sharing: On Extreme 8730 only, traffic classes 6–7 share multicast queues, preventing strict priority between them. Priority 6 traffic may flow even with Priority 7 traffic present. This limitation does not apply to Extreme 8520 or Extreme 8720.
- DWRR Multicast Behavior: Multicast traffic in DWRR queues might not follow configured weights exactly.

Monitor Queue Statistics

You can monitor queue performance as well as QoS queue performance by examining packet statistics per interface and packet statistics per interface subject to the QoS policy respectively.

Monitor Queue Performance

You can monitor queue performance by using the **show counters interface ethernet** { *interface-name* |**all** | **brief** } command. For details, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

For each interface, this command displays the following statistics:

- Packet counts per queue.
- Byte counts per queue.
- Drop statistics (if applicable).

The following example displays statistics for Ethernet interface 0/80.

```
device# show counters interface ethernet 0/80

Interface Statistics: ethernet 0/80
 Carrier Transitions: 2
           LastClear: 0s
Input:
              Total pkts: 24832
           Broadcast pkts: 24101
             Discard pkts: 505
              Errors pkts: 503
               FCS Errors: 0
               MCast pkts: 413
                   Octets: 3104
               UCast pkts: 1922
                Runt pkts: 502
               CRC Errors: 0
Input Distribution:
             64 byte pkts: 24002
         65-127 byte pkts: 131
        128-255 byte pkts: 232
        256-511 byte pkts: 396
       512-1023 byte pkts: 29
      1024-1518 byte pkts: 42
```

```
              Jumbo pkts: 0
Out:
              Total pkts: 2746
           Broadcast pkts: 2002
             Discard pkts: 101
              Errors pkts: 21
              MCast pkts: 42
                  Octets: 343
              UCast pkts: 237
Rate Info:
                   Input: 301.000000 Mbits/sec, 25402 pkts/sec 91.00% of line-rate
                  Output: 322.000000 Mbits/sec, 26312 pkts/sec 93.00% of line-rate
device#
```

The following example displays statistics for all Ethernet interfaces on the device.

```
device# show counters interface ethernet all

Interface Statistics: ethernet 0/80
 Carrier Transitions: 2
           LastClear: 0s
Input:
              Total pkts: 24832
           Broadcast pkts: 24101
             Discard pkts: 505
              Errors pkts: 503
              FCS Errors: 0
              MCast pkts: 413
                  Octets: 3104
              UCast pkts: 1922
               Runt pkts: 502
              CRC Errors: 0
Input Distribution:
             64 byte pkts: 24002
         65-127 byte pkts: 131
        128-255 byte pkts: 232
        256-511 byte pkts: 396
       512-1023 byte pkts: 29
      1024-1518 byte pkts: 42
              Jumbo pkts: 0
Out:
              Total pkts: 2746
           Broadcast pkts: 2002
             Discard pkts: 101
              Errors pkts: 21
              MCast pkts: 42
                  Octets: 343
              UCast pkts: 237
Rate Info:
                   Input: 301.000000 Mbits/sec, 25402 pkts/sec 91.00% of line-rate
                  Output: 322.000000 Mbits/sec, 26312 pkts/sec 93.00% of line-rate

Interface Statistics: ethernet 0/90
 Carrier Transitions: 2
           LastClear: 0s
Input:
              Total pkts: 24822
           Broadcast pkts: 24099
             Discard pkts: 503
              Errors pkts: 501
              FCS Errors: 0
              MCast pkts: 411
                  Octets: 3104
              UCast pkts: 1920
```

```
               Runt pkts: 502
              CRC Errors: 0
Input Distribution:
             64 byte pkts: 23992
          65-127 byte pkts: 129
         128-255 byte pkts: 230
         256-511 byte pkts: 394
        512-1023 byte pkts: 27
       1024-1518 byte pkts: 40
              Jumbo pkts: 0
Out:
              Total pkts: 2736
          Broadcast pkts: 2000
            Discard pkts: 91
             Errors pkts: 19
              MCast pkts: 40
                  Octets: 343
              UCast pkts: 235
Rate Info:
                   Input: 319.000000 Mbits/sec, 25702 pkts/sec 92.00% of line-rate
                  Output: 344.000000 Mbits/sec, 26912 pkts/sec 94.00% of line-rate
device#
```

Monitor QoS Queue Performance

You can validate QoS queue statistics by using the **show counters qos interface ethernet** { *interface-name* | **all** } **queue** { *queue-name* | **all** } [**pfc** | **ecn**] command. For details, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

For each interface, this command displays the following statistics:

- Received (Rx) packets per queue.
- Received (Rx) octets per queue.
- Packets that were dropped per queue according to the QoS policy.
- Octets (bytes) that were dropped per queue according to the QoS policy.

The following example displays statistics for Ethernet interface 0/1:1 QoS queue q0.

```
device# show counters qos interface ethernet 0/1:1 queue q0

-------------------------------------------------------
interface-id := ethernet 0/1:1
-------------------------------------------------------
Queue      Tx Packets Tx Octets  Drop Packets Drop Octets
========== ========== ========== ============ ===========
q0         1120       77818      0            0
device#
```

The following example displays statistics for Ethernet interface 0/1:1 and all QoS queues.

```
device# show counters qos interface ethernet 0/1:1 queue all

-------------------------------------------------------------------------
interface-id := ethernet 0/1:1
-------------------------------------------------------------------------
Queue    Rx Packets             Rx Octets              Drop Packets           Drop Octets
======== ====================== ====================== ======================
```

```
======================
q0       1120                  182560                0                     0
q1       0                     0                     0                     0
q2       6                     564                   0                     0
q3       0                     0                     0                     0
q4       0                     0                     0                     0
q5       0                     0                     0                     0
q6       949                   77818                 0                     0
q7       0                     0                     0                     0
device#
```

The following example displays PFC counter statistics for Ethernet interface 0/1:1:

```
device(config-qos-if-eth-0/1:1)# do sho counters qos interface ethernet 0/1:1 queue all
pfc
-------------------------------------------------------------------------
interface-id := ethernet 0/1:1
-------------------------------------------------------------------------
Priority   Rx Pause Frames    Tx Pause Frames    Tx XOFF            Rx XON
========== ================== ================== ================== ==================
0          0                  0                  0                  0
1          0                  0                  0                  0
2          0                  0                  0                  0
3          0                  0                  0                  0
4          0                  0                  0                  0
5          0                  0                  0                  0
6          0                  0                  0                  0
7          0                  0                  0                  0
```

If you use the optional pfc parameter, this command displays the following statistics:

- Priority per queue.
- Received PFC PAUSE frames per priority.
- Sent PFC PAUSE frames per priority.
- Received PFC PAUSE frames ignored because PFC was not enabled.
- Outgoing PFC PAUSE frames dropped due to internal errors.

The following example displays ECN statistics for Ethernet interface 0/1:4 and all QoS queues:

Note
The current release supports per-port ECN statistics, with statistics updated for queue 0 as a port aggregate. The current release does not support per-port per-queue ECN statistics.

```
DUT# show counters qos interface ethernet 0/1:4 queue all ecn
-------------------------------------------------------------------------
interface-id := ethernet 0/1:4
-------------------------------------------------------------------------
Queue    Marked Packets   Marked Bytes
======== ================ ================
q0       7512060416       961543733248
q1       0                0
q2       0                0
q3       0                0
q4       0                0
q5       0                0
q6       0                0
q7       0                0
```

Verification Methods

To verify proper scheduler operation, use the following practices:

1. Traffic Pattern Analysis: Monitor queue utilization during different traffic loads.
2. Performance Testing: Measure latency and throughput for different traffic classes.
3. Priority Verification: Confirm that strict priority queues receive preference during congestion.

Implementation Recommendations

Traffic Classification

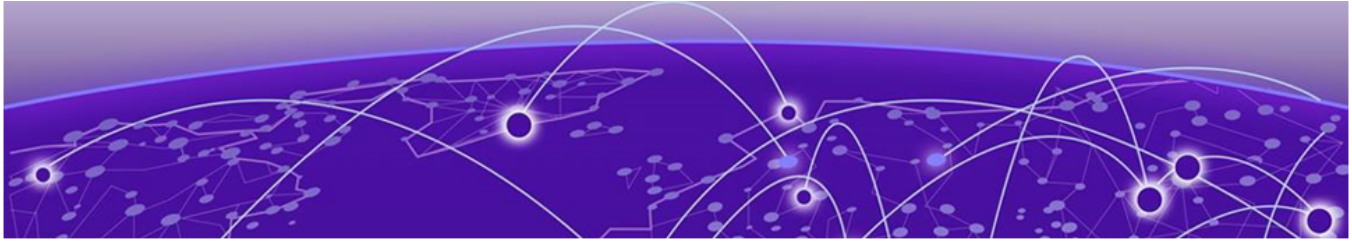
Use the following practices to classify traffic:

- Map Critical Applications: Assign mission-critical traffic to strict priority queues (6-7).
- Balance DWRR Weights: Configure DWRR weights based on application requirements and expected traffic volumes.
- Monitor Utilization: Regularly check queue utilization to identify potential bottlenecks.

Network Design

Use the following practices to design a network that plans for intelligent prioritization and scheduling of packet transmission at egress ports, ensuring optimal network performance and traffic flow management.

- Plan for Growth: Consider future traffic growth when designing QoS policies.
- Test Thoroughly: Validate scheduler behavior under various traffic conditions.
- Document Configuration: Maintain clear documentation of traffic class assignments.



Data Center Bridging

Data Center Bridging (DCB) is a collection of IEEE standards designed to make Ethernet suitable for lossless, low-latency communication in data center environments. DCB mitigates packet loss that occurs in Ethernet during congestion, while data center applications such as Remote Direct Memory Access (RDMA) and storage require guaranteed delivery.

This feature is supported on Extreme 8730 platforms.

OS ONE 22.2.3.0 supports the following features:

- Per-priority lossless transport: Supports Priority Flow Control (PFC) on up to 8 priorities (typically 1–2 for RoCEv2)
-
- PFC deadlock watchdog: Provides configurable detection interval. Recovery action (drop, auto-restore, or alert) is not supported, defaults to transmit (not drop).
- Two-level congestion control: Combines Explicit Congestion Notification (ECN) with PFC as a reactive safety mechanism
- Flexible egress scheduling: Supports Strict Priority and Deficit Weighted Round Robin (DWRR) and Weighted Round Robin (WRR) with configurable weights
- DCB Capability Exchange (DCBX) auto-negotiation: Supports willing and authoritative modes for PFC and Enhanced Transmission Selection (ETS) parameter exchange
- YANG operational state: Provides per-priority PFC counters, port-level ECN statistics, scheduler statistics, and DCBX FSM state

Limitations

This section describes known limitations and constraints for the QoS feature set in this release.

- This feature is only supported on Extreme 8730 platforms.
- The switch supports a maximum of 8 traffic classes or hardware queues per port (hardware limitation)
- The switch uses hardware-default headroom values that cannot be configured.
- PFC watchdog detection is software-based. The switch stores watchdog-timer and watchdog-action settings in YANG operational state, but does not program ASIC-level fields in this release.
- The switch does not support OpenConfig QoS buffer-management-profiles (not-supported deviation).
- The switch supports per-queue ECN only. Per-flow ECN is not supported.
- The switch does not support DCBX Converged Enhanced Ethernet (DCBX CEE) mode; support is limited to IEEE 802.1Qaz.
- LLDP and DCBX configuration does not support interface range syntax (for example, interface ethernet 0/1-0/32). Configure LLDP and DCBX on each physical interface individually.
- ETS bandwidth weights must sum to 100% across non–Strict Priority groups (or 0% when only Strict Priority traffic classes are configured).

Data Center Bridging Key Capabilities

Data Center Bridging (DCB) QoS supports four core features that together enable low-latency RDMA over Converged Ethernet v2 (RoCEv2) transport across leaf-spine data center fabrics.

Priority Flow Control

Priority Flow Control (PFC), IEEE 802.1Qbb, prevents packet loss by sending per-priority PAUSE frames after queue depth exceeds the **`xoff threshold`**. The interface sends PFC RESUME frames after queue depth falls to, or below, the **`xon threshold`**.

Enhanced Transmission Selection

Enhanced Transmission Selection (ETS) , IEEE 802.1Qaz, allocates egress bandwidth across traffic classes using Strict Priority or DWRR scheduling.

Explicit Congestion Notification

Explicit Congestion Notification (ECN) , RFC 3168 / Data Center Quantized Congestion Notification (DCQCN), signals congestion by marking packets with the CE bit, enabling senders to reduce rate before PFC fires. As a best practice, always pair ECN with PFC for RoCEv2.

DCB Capability Exchange

DCB Capability Exchange (DCBX), IEEE 802.1Qaz, negotiates PFC and ETS parameters between directly connected peers using Link Layer Discovery Protocol (LLDP) type-length-values (TLVs).

Two-Level Congestion Control

DCB implements a two-level congestion control model for lossless RoCEv2 transport:

- Level 1 (Proactive) uses ECN: When egress queue depth crosses **`min-threshold`**, the switch marks packets with the Congestion Experienced (CE) bit. The receiver generates a Congestion Notification Packet (CNP) to the sender, which reduces its transmission rate using the Data Center Quantized Congestion Notification (DCQCN) algorithm.
- Level 2 (Reactive) uses PFC: If the sender does not reduce its rate fast enough and queue depth reaches the xoff threshold, the switch sends a PFC PAUSE frame for that priority. This is a last resort to prevent packet loss.

Important
You must configure the ECN **`max-threshold`** well below the PFC **`xoff-threshold`**. If you configure ECN thresholds too close to the PFC **`xoff-threshold`**, ECN does not have time to signal congestion and slow the sender before PFC starts.

RoCE Configuration

RDMA over Converged Ethernet (RoCE) v2 is a transport protocol that enables remote direct memory access (RDMA) over standard Ethernet networks. RoCE v2 provides low-latency, high-throughput data transfer for data center applications, particularly those requiring high-performance computing and storage optimization.

Use the RoCE configuration feature to configure and manage RoCE v2 traffic on Extreme OS ONE TD4 platforms, ensuring optimal performance and interoperability with RoCE-capable endpoints.

RoCEv2 Protocol Overview

RoCEv2 encapsulates RDMA operations over UDP/IP (destination port 4791) and is fully routable over standard Layer 3 fabrics. RoCEv2 relies on IEEE Data Center Bridging

(DCB) standards to provide lossless behavior across Ethernet. The following traffic types must be treated differently:

Table 5: RoCEv2 traffic mapping

Traffic type	Differentiated Services Code Point	Queue/TCwith the default traffic-class-map	Mechanism
RDMA data	26	TC 3 lossless	PFC and ECN
Congestion notification (CNP)	48/CS6	TC 6 Strict Priority	No PFC needed
Network control	CS7	TC 7 Strict Priority	No PFC needed
Best effort/ management		TC 0 DWRR	No PFC or ECN

QoS Profile

Create one or more QoS profiles to group Priority Flow Control (PFC), Enhanced Transmission Selection (ETS), and Explicit Congestion Notification (ECN) configuration, and then bind the profile to an interface.

You can optionally enable DCB Capability Exchange (DCBX) on the interface to automatically negotiate PFC and ETS parameters with the peer device. When DCBX is enabled and negotiations occur, the resolved configuration may differ from the static QoS profile definition.

Default Profile

Before you configure custom named-QoS profiles, every port starts with a default profile named default. The default profile uses the following configuration:

- The profile assigns queues Q0–Q5 as Deficit Weighted Round Robin (DWRR) with graduated weights (5, 10, 15, 20, 25, 25), summing to 100%.
- The profile assigns queues (Q6, Q7) as Strict Priority
- ECN is disabled.
- PFC is disabled across all priorities.

RoCEv2 Reference Configuration

The following example is the best practice configuration for a leaf access port that connects to RoCEv2-capable servers on the device.

```
qos
  profile nik_roce
    description "RoCEv2 lossless profile for leaf access ports"
    pfc
      priority 1
        enable
      priority 3
        enable
        watchdog-timer 200
```

```
    queue-management
      queue 4
        ecn-enabled
        ecn-min-threshold 65536
        ecn-max-threshold 262144
        mark-probability 100
    ets
      tc 7
        scheduler strict-priority
      tc 6
        scheduler strict-priority
      tc 4
        scheduler dwrr
        bandwidth percent 50
      tc 0
        scheduler dwrr
        bandwidth percent 50
  interface ethernet 0/1:1
    profile nik_roce
    input
      trust cos
```

Configure a QoS Profile for RoCE

Use this procedure to create a named QoS profile and bind it to one or more interfaces. To configure a Data Center Bridging (DCB) feature in a profile, see Configure PFC on page 47, Configure ECN on page 50, or Configure ETS on page 55 .

1. From privileged EXEC mode, access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter QoS configuration mode.

   ```
   device(config)# qos
   device(config-qos)#
   ```

3. Create a QoS profile.

   ```
   device(config-qos)# profile <profile-name>
   device(config-qos-profile-<profile-name>)#
   ```

4. (Optional) Create a profile description.

   ```
   device(config-qos-profile-<profile-name>)# description "<text>"
   ```

5. Exit to QoS configuration mode.

   ```
   exit
   ```

6. Bind the profile to an interface.

   ```
   device(config-qos)# interface <intf-name>
   device(config-qos-interface-<intf-name>)# profile <profile-name>
   ```

7. Verify the profile configuration.

   ```
   device# show qos profile [profile-name]
   ```

The following example creates a profile named roce-lossless bound to ethernet 0/1.

```
device# configure terminal
device(config)# qos
device(config-qos)# profile roce-lossless
device(config-qos-profile-roce-lossless)# description "RoCEv2 lossless profile for leaf
access ports"
```

```
device(config-qos-profile-roce-lossless)#exit
device(config-qos)# interface ethernet 0/1
device(config-qos-interface-eth-0/1>)# profile roce-lossless
```

The following example shows how to display a QoS profile configuration.

```
device# show qos profile roce-lossless
show qos profile pfc1
Profile       : pfc1
  PFC:
Priority   Enabled      WD-Timer     WD-Action
---------- ------------ ------------ ---------
0          No           0            -
1          No           0            -
2          No           0            -
3          Yes          0            -
4          Yes          0            -
5          No           0            -
6          No           0            -
7          No           0            -
  ECN:
Queue      ECN          Ecn-Min-Thresh  Ecn-Max-Thresh Ecn-Mark-Prob
---------- ------------ --------------- -------------- -------------
0          No           0               0              0%
1          No           0               0              0%
2          No           0               0              0%
3          No           0               0              0%
4          No           0               0              0%
5          No           0               0              0%
6          No           0               0              0%
7          No           0               0              0%
  ETS:
Queue   Sched-Type   BW-Weight Shaper-Max(%)  Shaper-Min(%)
------- ------------ --------- -------------- --------------
0       DWRR         5         -              -
1       DWRR         10        -              -
2       DWRR         15        -              -
3       DWRR         20        -              -
4       DWRR         25        -              -
5       DWRR         25        80             -
6       SP           -         -              -
7       SP           -         -              -

SQA-RoCEv2#
```

QoS Profile gNMI Commands

Use the gRPC Network Management Interface (gNMI) commands for configuring a QoS profile.

QoS Profile Creation

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/profiles/profile[profile-
name=nik_roce]/config/profile-name:::string:::nik_roce"
```

QoS Profile Deletion

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/schiluveri/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/profiles/profile[profile-name=nik_roce]"
```

Bind a QoS Profile to an Interface

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/schiluveri/repo/
latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/interfaces/
interface[interface-id=ethernet 0/1]/config/profile-name:::string:::nik_roce"
```

Unbind a QoS Profile from an Interface

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks --tls-ca /home/schiluveri/repo/
latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/interfaces/
interface[interface-id=ethernet 0/1]/config/profile-name"
```

PFC Configuration

Priority-based Flow Control (PFC) is an IEEE 802.1Qbb feature that provides per-priority flow control using PAUSE frames. PFC enables lossless transmission by allowing receiving devices to pause transmission on specific priority levels, preventing packet loss due to buffer overflow.

When integrated with RoCE v2, PFC ensures reliable RDMA data transfer by preventing packet loss on RoCE traffic queues, allowing ECN and ETS mechanisms to manage congestion effectively.

Important
If you enable DCBX on an interface, PFC does not operate unless a DCBX peer is present and the peer's PFC priority bitmap exactly matches the locally-configured PFC priorities. A single priority mismatch disables PFC on all priorities. Verify peer negotiation with the **`show qos dcbx interface`** command.

PFC Direction

PFC on each priority operates symmetrically: the interface both generates PAUSE frames (TX) and honors incoming PAUSE frames (RX). You cannot configure TX-only or RX-only PFC.

PFC Watchdog

The PFC watchdog detects and mitigates PFC PAUSE storms. When a priority is paused for longer than the configured watchdog timer, the switch transmits for a short duration, ignoring the XOFF state, to try and drain the queue. This is not configurable.

PFC Headroom

Headroom is the ingress buffer reserved for in-flight packets that arrive after a PFC PAUSE frame is sent but before the upstream sender receives and acts on it. Without enough headroom, those in-flight packets overflow the buffer and are dropped.

The hardware ASIC decides the XOFF and XON thresholds, they are dynamic and not user-configurable.

Configure PFC

Use this procedure to enable Priority-based Flow Control (PFC) on one or more priorities in a QoS profile, configure the PFC watchdog, and configure headroom thresholds.

PFC operates per-priority in a QoS profile. Before configuring PFC, ensure that you create the QoS profile. If you enable DCBX on the target interface, PFC does not activate until the interface detects a matching DCBX peer.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Enter QoS configuration mode.

```
device(config)# qos
device(config-qos)#
```

3. Configure a QoS profile.

```
device(config-qos)# profile <profile-name>
device(config-qos-profile-<profile-name>)#
```

4. Enter the PFC sub-mode of the profile.

```
device(config-qos-profile-<profile-name>)# pfc
device(config-qos-profile-<profile-name>-pfc)#
```

5. Configure PFC priority settings.

```
device(config-qos-profile-<profile-name>-pfc)# priority <0-7>
device(config-qos-profile-<profile-name>-pfc-priority-#)# {enable | watchdog-timer
<100-60000>}
```

6. Exit to QoS profile PFC configuration mode.

```
exit
```

The following example configures PFC for priority 3 in a QoS profile named roce-lossless.

```
device# configure terminal
device(config)# qos
device(config-qos)# profile roce-lossless
device(config-qos-profile-roce-lossless)# pfc
device(config-qos-profile-pfc)# priority 3
device(config-qos-profile-pfc-priority-3)# watchdog-timer 200
device(config-qos-profile-pfc-priority-3)# enable
```

PFC gNMI Commands

Use the gNMI commands for configuring Priority-based Flow Control (PFC) in a QoS profile.

Enable PFC on Priority 3

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \ --tls-ca /home/user1/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/profiles/profile[profile-
name=nik_roce]/pfc/priorities/priority[priority=3]/config/priority:::uint:::3" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/pfc/priorities/
priority[priority=3]/config/enable:::bool:::true"
```

Disable PFC on Priority 3

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \ --tls-ca /home/schiluveri/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/profiles/profile[profile-
name=nik_roce]/pfc/priorities/priority[priority=3]/config/enable:::bool:::false"
```

PFC Watchdog Timer and Action Configuration

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \ --tls-ca /home/schiluveri/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/profiles/profile[profile-
name=nik_roce]/pfc/priorities/priority[priority=3]/config/watchdog-timer:::uint:::500" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/pfc/priorities/
priority[priority=3]/config/watchdog-action:::string:::PFC_WATCHDOG_DROP"
```

Delete PFC Priority

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \ --tls-ca /home/schiluveri/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/profiles/profile[profile-
name=nik_roce]/pfc/priorities/priority[priority=3]"
```

Get PFC Operational State

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \ --tls-ca /home/schiluveri/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem get --path "/qos/profiles/profile[profile-
name=nik_roce]/pfc/priorities/priority[priority=3]/state"
```

Profile-Level Headroom Cable Length and Propagation Delay Configuration

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \ --tls-ca /home/schiluveri/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/profiles/profile[profile-
name=nik_roce]/pfc/headroom/config/cable-length:::uint:::10" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/pfc/headroom/config/
propagation-delay:::uint:::1000"
```

Per-Priority Headroom Threshold Configuration

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \ --tls-ca /home/schiluveri/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --update "/qos/profiles/profile[profile-
name=nik_roce]/pfc/headroom/priorities/priority[priority=3]/config/priority:::uint:::3" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/pfc/headroom/priorities/
priority[priority=3]/config/xoff-threshold:::uint:::1048576" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/pfc/headroom/priorities/
priority[priority=3]/config/xon-threshold:::uint:::786432" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/pfc/headroom/priorities/
priority[priority=3]/config/headroom-size:::uint:::131072"
```

Delete Per-Priority Headroom

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \ --tls-ca /home/schiluveri/repo/latest/tierra-
os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set --delete "/qos/profiles/profile[profile-
name=nik_roce]/pfc/headroom/priorities/priority[priority=3]"
```

ECN Configuration

Explicit Congestion Notification (ECN) enables network devices to communicate congestion information to endpoints without dropping packets. ECN uses reserved bits in the IP header to signal congestion, allowing endpoints to reduce their transmission rate proactively. ECN can prevent congestion from escalating to the point where you need Priority-based Flow Control (PFC).

The ECN feature integrates with RoCE v2 to provide congestion management for RDMA traffic, enabling lossless data transmission in highly utilized networks.

How ECN Works

ECN operates on egress queues. Instead of dropping packets or pausing the sender, ECN marks packets with a congestion indicator, Congestion Experienced (CE), in the IP header. The receiver then generates a Congestion Notification Packet (CNP) back to the sender, which triggers the Data Center Quantized Congestion Notification (DCQCN) algorithm on the sender to reduce its transmission rate.

ECN only marks packets that are ECN-capable; meaning the sender includes the ECN-Capable Transport (ECT) bits in the IP header. The interface does not mark non-ECN-capable packets that pass through the same queue.

You configure ECN per-queue in a QoS profile. In addition to enabling ECN marking per queue, you can also configure three threshold values. The following table explains what happens to packet marking based on these thresholds.

Table 6: Threshold impacts on ECN marking

Queue depth	What happens
Below the **min-threshold**	No marking—traffic flows normally
Between the **min-threshold** and **max-threshold**	Probabilistic marking—the percentage increases from 0% up to **mark-probability**
Above **max-threshold**	100% of ECN-capable packets are marked

Important

You must configure the ECN **max-threshold** well below the PFC **xoff-threshold**. If you configure ECN thresholds too close to the PFC **xoff-threshold**, ECN does not have time to signal congestion and slow the sender before PFC starts.

The following table describes marking and drop behavior.

Table 7: Mark vs drop behavior

IP ECN field	Name	Queue depth > **min-threshold**	Queue depth > **max-threshold**
00	Not ECT	No action	Tail-drop if the queue is full
01	ECT(1)	Probabilistic CE mark	100% CE mark
10	ECT(0)	Probabilistic CE mark	100% CE mark
11	CE	Already marked—forward as is	Already marked—forward as is

Configure ECN

Configure Explicit Congestion Notification (ECN) in a QoS profile to enable CE-bit marking on a per-queue basis for lossless RoCEv2 transport.

Configure ECN per queue in a QoS profile. Ensure you configure **max-threshold** (Kmax) well below the Priority-based Flow Control (PFC) **xoff-threshold** configured for the same traffic class.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Enter QoS configuration mode.

```
device(config)# qos
device(config-qos)#
```

3. Create a QoS profile and enter queue-management configuration mode.

```
device(config-qos)# profile <profile-name>
device(config-qos-profile-<profile-name>)# queue-management
device(config-qos-profile-queue-mgmt)#
```

4. Configure ECN for a specific queue.

```
device(config-qos-profile-queue-mgmt)# queue <0-7>
device(config-qos-profile-queue-mgmt-queue-#)# ecn-enabled
device(config-qos-profile-queue-mgmt-queue-#)# ecn-min-threshold <bytes>
device(config-qos-profile-queue-mgmt-queue-#)# ecn-max-threshold <bytes>
device(config-qos-profile-queue-mgmt-queue-#)# mark-probability <0-100>
```

The following example enables and configures ECN for queue 4 in a QoS profile named roce-lossless.

```
device# configure terminal
device(config)# qos
device(config-qos)# profile roce-lossless
device(config-qos-profile-roce-lossless)# queue-management
device(config-qos-profile-queue-mgmt)# queue 4
device(config-qos-profile-queue-mgmt-queue-4)# ecn-enabled
device(config-qos-profile-queue-mgmt-queue-4)# ecn-min-threshold 65536
device(config-qos-profile-queue-mgmt-queue-4)# ecn-max-threshold 262144
device(config-qos-profile-queue-mgmt-queue-4)# mark-probability 100
```

```
device(config-qos-profile-queue-mgmt-queue-4)# exit
device(config-qos-profile-queue-mgmt)# exit
device(config-qos-profile-roce-lossless)# exit
device(config-qos)# exit
device(config)# exit
device#
```

ECN gNMI Commands

Use the gNMI commands for configuring Explicit Congestion Notification (ECN).

Enable ECN on Queue 4 with Thresholds

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
set --update "/qos/profiles/profile[profile-name=nik_roce]/ecn/queues/queue[queue-id=4]/
config/queue-id:::uint:::4" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/ecn/queues/queue[queue-id=4]/
config/ecn-enabled:::bool:::true" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/ecn/queues/queue[queue-id=4]/
config/min-threshold:::uint:::100000" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/ecn/queues/queue[queue-id=4]/
config/max-threshold:::uint:::200000" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/ecn/queues/queue[queue-id=4]/
config/mark-probability:::uint:::100"
```

Disable ECN on Queue 4

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
set --update "/qos/profiles/profile[profile-name=nik_roce]/ecn/queues/queue[queue-id=4]/
config/ecn-enabled:::bool:::false"
```

Delete ECN Queue Entry

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set
--delete "/qos/profiles/profile[profile-name=nik_roce]/ecn/queues/queue[queue-id=4]"
```

Get ECN Operational State

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem get
--path "/qos/profiles/profile[profile-name=nik_roce]/ecn/queues/queue[queue-id=4]/state"
```

ETS Configuration

Enhanced Transmission Selection (ETS) is an IEEE 802.1Qaz feature that provides bandwidth management through weighted round-robin (WRR) scheduling. Use ETS to allocate minimum guaranteed bandwidth to traffic classes, ensuring that critical applications receive their required bandwidth.

ETS ensures low-latency Remote Direct Memory Access (RDMA) traffic in RDMA over Converged Ethernet v2 (RoCE v2) by guaranteeing minimum bandwidth and permitting bursts when bandwidth is available.

Scheduler Types

Each egress port has eight hardware queues (Q0–Q7). ETS assigns one of the following scheduler types to each traffic class (TC).

Note

Traffic Classes (TC 0–7) map 1:1 to egress hardware queues (Q0–Q7). This document uses `TC` when referring to classification and scheduling policy, and `queue` when referring to physical egress buffers.

Table 8: Scheduler types and recommended use

Scheduler type	Behavior	Recommended use
STRICT_PRIORITY	Always served before lower-priority queues. Can starve other traffic classes if the queue carries high-rate traffic.	Low-bandwidth control traffic: RoCEv2 CNP (Q6/Q7), Bidirectional Forwarding Detection (BFD), Link Aggregation Control Protocol (LACP)
DWRR	Deficit Weighted Round Robin — weight-proportional bandwidth sharing that handles variable frame sizes fairly. This is also the scheduler type applied when a DCBX peer negotiates IEEE 802.1Qaz ETS bandwidth allocation.	RoCEv2 RDMA data traffic

Recommended RoCEv2 ETS Configuration

As a best practice, use the following traffic class assignment for leaf access ports connected to RoCEv2-capable servers.

Table 9: Recommended traffic class assignment for RoCEv2

TC	Scheduler	Bandwidth	Traffic
7	STRICT_PRIORITY	—	OSPF, BGP, BFD (DSCP CS7)
6	STRICT_PRIORITY	—	RoCEv2 CNP — Congestion Notification Packet (DSCP CS6)
4	DWRR	50%	RoCEv2 RDMA data (PFC-enabled priority)
0	DWRR	50%	TCP, management, non-RoCE traffic

ETS Rules

ETS bandwidth allocation follows these rules:

Default scheduling

Every port starts with a default QoS profile that assigns Q0–Q5 as DWRR with graduated weights (5%, 10%, 15%, 20%, 25%, 25%) and Q6–Q7 as strict-priority. When you apply a named profile to an interface, the system overrides only the queues

you explicitly configure. Queues not specified in the profile retain their default scheduling.

Sum-to-100% constraint

The `bandwidth-weight` values for all non-SP traffic classes must sum to exactly 100%. If you configure all traffic classes as strict-priority, the sum must be 0%. The system validates this constraint at profile creation and again when you bind the profile to an interface, and rejects configurations that do not meet it.

Work-conserving behavior

DWRR scheduling is work-conserving. When a DWRR queue has no traffic, the scheduler redistributes its bandwidth share proportionally to the other active DWRR queues, so idle queues waste no bandwidth. For example, if you configure Q4 (50%) and Q0 (50%) but Q0 has no traffic, Q4 receives 100% of the available DWRR bandwidth after the scheduler serves SP queues.

Important
Strict Priority queues are always served before DWRR queues. Assigning high-rate traffic to a strict-priority queue starves DWRR queues. Use strict-priority only for low-bandwidth control traffic such as CNP, BFD, and routing protocol keepalives.

The system enforces the following rules when you configure ETS traffic classes in a profile and rejects any configuration that violates them.

Table 10: ETS configuration rules

Rule	Description
Bandwidth weights must sum to 100%	The bandwidth percent values for all non-strict-priority traffic classes must total exactly 100%.
All strict-priority requires 0% total bandwidth	If every traffic class in the profile is configured as strict-priority, the total bandwidth weight must be 0%.
No duplicate traffic classes	Each traffic class (0–7) can only be configured once in a profile.
Allows mixed scheduling	Strict-priority and DWRR/WRR traffic classes can coexist in the same profile. The system always services strict-priority queues first.

ETS Configuration Examples

The following is an example of a valid configuration.

```
device(config-qos-profile-ets)# sp-count 2 dwrr-weights 10 10 10 25 25 10
device(config-qos-profile-ets)# traffic-class 0
device(config-qos-profile-ets-tc-0)# min-bandwidth 5
device(config-qos-profile-ets-tc-0)# max-bandwidth 50
device(config-qos-profile-ets-tc-0)# exit
device(config-qos-profile-ets)# traffic-class 3
device(config-qos-profile-ets-tc-3)# min-bandwidth 15
```

```
device(config-qos-profile-ets-tc-3)# max-bandwidth 80
device(config-qos-profile-ets-tc-3)# exit
device(config-qos-profile-ets)# traffic-class 4
device(config-qos-profile-ets-tc-4)# min-bandwidth 15
device(config-qos-profile-ets-tc-4)# max-bandwidth 80
device(config-qos-profile-ets-tc-4)# exit
device(config-qos-profile-ets)# exit
device(config-qos-profile-roce-lossless)# exit
device(config-qos)# exit
device(config)# exit
device#
```

The following is an example shows incomplete DWRR bandwidth allocation. When you associate this QoS profile to an interface, the system validates that all DWRR-scheduled traffic classes sum to 100%. In this example, the configuration allocates only 80% (50% + 30%), so the validation fails at association time.

```
device(config-qos-profile-ets)# tc 3
device(config-qos-profile-ets-tc-3)# scheduler dwrr
device(config-qos-profile-ets-tc-3)# bandwidth percent 50
device(config-qos-profile-ets-tc-3)# exit
device(config-qos-profile-ets)# tc 4
device(config-qos-profile-ets-tc-4)# scheduler dwrr
device(config-qos-profile-ets-tc-4)# bandwidth percent 30
```

ETS DCBX Behavior

When you enable Data Center Bridging Capability Exchange (DCBX) on a port, the switch exchanges ETS configuration with the link peer and applies the following rules.

Table 11: ETS DCBX negotiation behavior

Condition	Behavior
Local and peer ETS configurations match	Negotiated ETS is applied to the port.
Local and peer ETS configurations differ	Traffic continues to flow. Only bandwidth distribution is affected — no traffic is dropped or disabled.
Peer advertises invalid ETS (for example, bandwidth does not sum to 100%)	The switch ignores the peer ETS configuration and uses the local configuration.
DCBX is disabled or peer does not support DCBX	The switch applies the local ETS configuration directly.

Note

Unlike PFC, an ETS mismatch with a DCBX peer never disables traffic on any priority. Traffic always continues to flow — only the bandwidth allocation between queues can differ from what was intended.

Configure ETS

Use this procedure to configure Enhanced Transmission Selection (ETS) in a QoS profile to assign a scheduler type and bandwidth allocation to each traffic class.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Enter QoS configuration mode.

```
device(config)# qos
device(config-qos)#
```

3. Create a QoS profile.

```
device(config-qos)# profile roce-lossless
device(config-qos-profile-roce-lossless)#
```

4. Configure ETS with strict-priority count and DWRR weights.

```
device(config-qos-profile-roce-lossless)# ets
device(config-qos-profile-ets)# sp-count 2 dwrr-weights 50 50
device(config-qos-profile-ets)#
```

5. Optional: Configure minimum and maximum bandwidth limits for individual traffic classes.

```
device(config-qos-profile-ets)# traffic-class <0-7>
device(config-qos-profile-ets-tc)# min-bandwidth <0-4294967295>
device(config-qos-profile-ets-tc)# max-bandwidth <0-4294967295>
```

The following example configures ETS with 2 strict-priority queues and 6 DWRR-scheduled queues with equal 50% bandwidth allocation in a QoS profile named roce-lossless.

```
device# configure terminal
device(config)# qos
device(config-qos)# profile roce-lossless
device(config-qos-profile-roce-lossless)# ets
device(config-qos-profile-ets)# sp-count 2 dwrr-weights 50 50
device(config-qos-profile-ets)# traffic-class 0
device(config-qos-profile-ets-tc-0)# min-bandwidth 10
device(config-qos-profile-ets-tc-0)# max-bandwidth 80
device(config-qos-profile-ets-tc-0)# exit
device(config-qos-profile-ets)# traffic-class 4
device(config-qos-profile-ets-tc-4)# min-bandwidth 10
device(config-qos-profile-ets-tc-4)# max-bandwidth 80
device(config-qos-profile-ets-tc-4)# exit
```

.

ETS gNMI Commands

Use the gNMI commands for configuring Enhanced Transmission Selection (ETS) in a QoS profile.

ETS Configuration with Strict Priority Count and DWRR Weights

Using Go gNMI Client (Programmatic):

```
prefix := "/qos/profiles/profile[profile-name=nik_roce]/ets"
updates := []*gnmi.Update{
    {Path: "config/sp-count",      Val: UintVal(2)},
    {Path: "config/dwrr-weights",  Val: LeaflistVal([UintVal(10), UintVal(10),
```

```
UintVal(10), UintVal(25), UintVal(25), UintVal(10)])},
}
// Both must be in the SAME SetRequest (validator rejects sp-count alone)
```

Minimum and Maximum Bandwidth Limits Configuration

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
set --update "/qos/profiles/profile[profile-name=nik_roce]/ets/traffic-classes/traffic-
class[traffic-class=3]/config/min-bandwidth:::uint:::10" \
    --update "/qos/profiles/profile[profile-name=nik_roce]/ets/traffic-classes/traffic-
class[traffic-class=3]/config/max-bandwidth:::uint:::80"
```

Delete ETS Configuration

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem set
--delete "/qos/profiles/profile[profile-name=nik_roce]/ets/config/sp-count" \
    --delete "/qos/profiles/profile[profile-name=nik_roce]/ets/config/dwrr-weights"
```

Get ETS Operational State

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem --
encoding proto get \
  --path "/qos/profiles/profile[profile-name=nik_roce]/ets/state"
```

DCBX Configuration

Data Center Bridging Capability Exchange (DCBX) uses LLDP TLVs to automatically negotiate Priority-based Flow Control (PFC) and Enhanced Transmission Selection (ETS) parameters between directly-connected peers. If you enable DCBX on an interface, the switch exchanges 802.1Qaz TLVs with the connected network interface card (NIC) or peer switch, and the resolved operational state can differ from the locally configured QoS profile.

Willing and Authoritative Modes

Each DCBX feature operates in one of two modes. The willing bit determines who wins if the local switch and the peer disagree on PFC or ETS configuration.

Table 12: Willing bit behavior

Feature	Local mode	Behavior	On mismatch
PFC	Authoritative (willing bit=false)	The switch is always authoritative. The peer must match the switch PFC configuration exactly.	PFC is disabled on all priorities
ETS	Willing (willing bit=true)	The switch can adopt a peer ETS configuration if the peer is authoritative.	ETS uses local config; traffic continues

PFC Resolution

If you enable DCBX on an interface, PFC operates according to the following rules:

Table 13: PFC behavior resolution

DCBX enabled	Peer present	PFC bitmap mismatch	PFC result
No	N/A	N/A	Local PFC configuration is active.
Yes	No	N/A	PFC is disabled on all priorities.
Yes	Yes	Match	PFC symmetric is enabled on all priorities.
Yes	Yes	Mismatch	PFC is disabled on all priorities.

ETS Resolution

ETS mismatch is non-fatal. Traffic always continues. If the peer ETS bandwidth weights are invalid (do not sum to 100%), or if no peer is present, the switch uses the local ETS configuration.

Configure DCBX

Use this procedure to enable Data Center Bridging Capability Exchange (DCBX) on a physical interface so the switch can negotiate Priority-based Flow Control (PFC) and Enhanced Transmission Selection (ETS) parameters with a connected peer.

You must enable LLDP on the interface before you enable DCBX.

DCBX operates per physical interface only. DCBX is not supported on VLANs or individual LAG member ports. DCBX requires LLDP. If you enable DCBX, PFC does not activate until the switch detects a matching DCBX peer.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Specify the Ethernet interface for which you want to configure DCBX.

```
device(config)# interface ethernet <name>
device(config-if-eth-<name>)#
```

3. Enable DCBX on the interface.

```
device(config-if-eth-<name>)# lldp
device(config-if-eth-0/1-lldp)#dcbx
device(config-if-eth-0/1-lldp-dcbx)#enable
```

4. Verify the configuration.

```
device#show qos dcbx interface <name>
```

The following example enables DCBX on Ethernet interface 0/1.

```
device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# lldp
device(config-if-eth-0/1-lldp)# dcbx
device(config-if-eth-0/1-lldp-dcbx)# enable
```

The following example shows the DCBX configuration on Ethernet interface 0/1.

```
device# show qos dcbx interface ethernet 0/1
Interface: ethernet 0/1
  DCBX Enabled: true
  FSM State: RESOLVED_TO_LOCAL
  PFC In-Sync: true
  ETS In-Sync: true
  Peer PFC Bitmap: 0x08 (priority 3)
  Peer ETS Bandwidth: TC0=50%, TC4=50%, TC6=SP, TC7=SP
```

DCBX gNMI Commands

Use the gNMI commands for configuring Data Center Bridging Capability Exchange (DCBX) .

Get DCBX State for an Interface

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem get
--path "/qos/dcbx-operational/interfaces/interface[name=ethernet 0/1]/dcbx"
```

Get DCBX FSM State

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
get --path "/qos/dcbx-operational/interfaces/interface[name=ethernet 0/1]/dcbx/state/fsm-
state"
```

Get PFC Sync Status

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem get
--path "/qos/dcbx-operational/interfaces/interface[name=ethernet 0/1]/dcbx/state/pfc-in-
sync"
```

Get ETS Sync Status

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem get
--path "/qos/dcbx-operational/interfaces/interface[name=ethernet 0/1]/dcbx/state/ets-in-
sync"
```

Subscribe to DCBX State Changes

```
gnmic -a DUT_IP:PORT_NUM -u admin -p rocks \
  --tls-ca /home/user1/repo/latest/tierra-os/mgmt-svc/gnmi_client/extreme-ca.cert.pem
subscribe --path "/qos/dcbx-operational/interfaces/interface[name=ethernet 0/1]/dcbx" --
stream-mode on-change
```

Monitoring Data Center Bridging

Use the following commands to monitor Data Center Bridging (DCB) feature state and troubleshoot PFC, ETS, ECN, and DCBX behavior.

Display QoS Profile Configuration

To display the configuration of a named QoS profile and the number of bound interfaces, use the **show qos profile** *[name]* command. Omit the profile name to display all profiles.

Display Resolved QoS Interface State

To display the resolved PFC and ETS operational state for an interface, showing what the system has programmed into the ASIC, use the **show qos interface <intf>** command. The resolved state can differ from the bound profile when you enable DCBX on the interface.

The following example displays the QoS interface binding, including the bound profile name, profile sync status, trust mode, resolved PFC state per priority, resolved ETS scheduling per traffic class, and DCBX status.

```
device# show qos interface ethernet 0/1:1
Interface: ethernet 0/1:1
  Profile: roce-lossless
  Profile Sync Status: IN_SYNC
  Trust Mode: COS
  Resolved PFC:
    Priority 3: SYMMETRIC  (tx=true, rx=true)
    Others: DISABLED
  Resolved ETS:
    TC 0: DWRR, bandwidth=50%
    TC 4: DWRR, bandwidth=50%
    TC 6: STRICT_PRIORITY
    TC 7: STRICT_PRIORITY
  DCBX:
    Enabled: true
    PFC In-Sync: true
    ETS In-Sync: true
```

Display the DCBX FSM State

The DCBX Finite State Machine (FSM) controls how the local switch resolves its QoS configuration when a DCBX-capable peer is detected. The FSM transitions through four states listed in the following table.

Table 14: DCBX FSM states

FSM state	Description
IDLE	No DCBX peer is detected on the interface. The switch enters this state at link-up or when the peer has DCBX disabled.
RX CACHED	The switch has received and cached the peer DCBX TLV. Resolution is in progress. This is a transient state.
RESOLVED_TO_LOCAL	Resolution is complete. The local and peer configurations are compatible and the switch applies the local configuration.
OVERRIDE_ACTIVE	Resolution is complete. The switch overrides the peer configuration because the local `willing` flag is set to false and a mismatch exists, or the switch has adopted the peer parameters because the local `willing` flag is configured to true.

To display the DCBX FSM operational state and per-feature sync status for an interface, use the **`show qos dcbx interface <intf>`** command.

The following example displays the FSM state as RESOLVED_TO_LOCAL, with PFC and ETS in sync between the local and peer configurations.

```
device# show qos dcbx interface ethernet 0/1:1
Interface: ethernet 0/1:1
  DCBX: Enabled

  FSM State: RESOLVED_TO_LOCAL

  PFC:
    Local Willing: false (authoritative)
    Local PFC Bitmap: 0b00001010 (priority 1, 3)
    Peer PFC Bitmap: 0b00001010 (priority 1, 3)
    PFC In-Sync: true

  ETS:
    Local Willing: true
    Local ETS:
      TC 0: DWRR, bandwidth=50%
      TC 4: DWRR, bandwidth=50%
      TC 6: STRICT_PRIORITY
      TC 7: STRICT_PRIORITY
    Peer ETS:
      TC 0: DWRR, bandwidth=50%
      TC 4: DWRR, bandwidth=50%
      TC 6: STRICT_PRIORITY
      TC 7: STRICT_PRIORITY
    ETS In-Sync: true
```

The following example displays the FSM state as OVERRIDE_ACTIVE, where the local PFC configuration overrides the peer due to a PFC bitmap mismatch and differing ETS bandwidth allocations.

```
device# show qos dcbx interface ethernet 0/1:1
Interface: ethernet 0/1:1
  DCBX: Enabled

  FSM State: OVERRIDE_ACTIVE

  PFC:
    Local Willing: false (authoritative)
    Local PFC Bitmap: 0b00001010 (priority 1, 3)
    Peer PFC Bitmap: 0b00001000 (priority 3 only)
    PFC In-Sync: false

  ETS:
    Local Willing: true
    Local ETS:
      TC 0: DWRR, bandwidth=50%
      TC 4: DWRR, bandwidth=50%
      TC 6: STRICT_PRIORITY
      TC 7: STRICT_PRIORITY
    Peer ETS:
      TC 0: DWRR, bandwidth=30%
      TC 4: DWRR, bandwidth=70%
      TC 6: STRICT_PRIORITY
      TC 7: STRICT_PRIORITY
    ETS In-Sync: false
```

Display PFC statistics

To display PFC statistics for all egress queues on the specified interface, use the **`show counters qos interface <intf> queue all pfc`**.

The following example displays PFC counters for all queues on interface ethernet 0/1:1, including the number of PFC PAUSE frames transmitted (XOFF sent) and received (XOFF received) per priority, watchdog trigger events, and frames dropped due to PFC watchdog action.

```
device# show counters qos interface ethernet 0/1:1 queue all pfc
Interface: ethernet 0/1:1

Priority  PFC TX (XOFF Sent)  PFC RX (XOFF Rcvd)  Watchdog Triggers  Watchdog Drops
--------  ------------------  ------------------  -----------------  --------------
   0                     0                   0                  0               0
   1                    42                  38                  0               0
   2                     0                   0                  0               0
   3                  1284                1197                  1              64
   4                     0                   0                  0               0
   5                     0                   0                  0               0
   6                     0                   0                  0               0
   7                     0                   0                  0               0
```