Nokia Service Router Linux
7220 Interconnect Router
7250 Interconnect Router

Release 25.10

7220 IXR and 7250 IXR Quality of Service Guide

3HE 21400 AAAC TQZZA
Edition: 01
November 2025

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

Table of contents

1 About this guide

This document describes configuration details for the Quality of Service (QoS) feature set used with the Nokia Service Router Linux (SR Linux).

Note:
This guide describes QoS feature support on 7220 IXR and 7250 IXR platforms only. For information about QoS feature support on 7730 SXR platforms, see the *7730 SXR SR Linux Quality of Service Guide*.

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how the router is configured.

Note:
This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *SR Linux Software Release Notes* for information on features supported in each load.

Configuration and command outputs shown in this guide are examples only; actual displays may differ depending on supported functionality and user configuration.

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2 Conventions

Nokia SR Linux documentation uses the following command conventions.

- **Bold** type indicates a command that the user must enter.

- Input and output examples are displayed in `Courier` text.
- An open right-angle bracket indicates a progression of menu choices or simple command sequence (often selected from a user interface). Example: **start** > **connect to**.
- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ([ ]) indicate optional elements.
- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

Generic IP addresses are used in examples. Replace these with the appropriate IP addresses used in the system.

2 What's new

Topic	Location
PFC updated to support: • enabling PFC both per-interface and per-priority • allowing changes to existing configurations to be applied without administratively disabling and re-enabling PFC	Priority-based Flow Control (PFC)
DCBX	DCBX
VOQ statistics collection	VOQ statistics collection

3 Quality of service overview

Quality of Service (QoS) provides an appropriate level of service for packets as they flow inside the switch and between switches in the network. The required level of service depends on the application that generates the flow of packets, and can be defined by the application's sensitivity to packet loss, delay, and jitter.

QoS functionality is supported on the following platforms:

- 7250 IXR series (7250 IXR-6/6e/10/10e/X1b/X3b)
- 7220 IXR D series (7220 IXR-D2/D3/D4/D5)
- 7220 IXR DL series (7220 IXR-D2L/D3L)
- 7220 IXR H series (7220 IXR-H2/H3/H4/H5)

Note:
The 7220 IXR-H4/H5 platforms do not support the following QoS features:

- ingress subinterface traffic policing
- Dot1p classification and marking
- multifield classification policies
- static queue utilization thresholds

The 7220 IXR-D4/D5 platforms do not support the following QoS feature:

- Queue depth sampling

You can group packets that require a similar treatment (per-hop behavior) into an FC, also known as a behavior aggregate. You can specify up to eight FCs. Traffic is scheduled and can optionally be marked based on its FC.

A configurable drop probability expresses the packet loss sensitivity. Assign a low drop probability to packets that are sensitive to loss. To provide the required congestion management and intelligent discard decisions when congestion occurs, balance the traffic classifications between low, medium, and high drop probability.

3.1 How QoS works for transit traffic

This section describes how QoS applies to transit packets on SR Linux.

Note: The full set of SR Linux QoS features are described below; however, not all platforms support all listed features. Ignore any feature information that is not applicable for your platform.

1. Packets are received on a subinterface.
2. Each received packet is classified as belonging to one of eight forwarding classes (corresponding to forwarding class indexes 0 to 7) and one of three drop probabilities (low, medium, or high).
    - **For IP packets**:

  - If the packet matches a multifield classifier policy configured on the ingress subinterface, the FC and drop probability level are determined entirely from that policy. In addition, if this policy includes a DSCP rewrite action, the DSCP value for the packet is rewritten accordingly.
  - Otherwise, if the packet matches a DSCP classifier policy configured on the ingress subinterface, the forwarding class and drop probability level are determined from that policy.

    Note: If there is no entry of this policy matching the received DSCP, the assigned forwarding class index is 0 and the assigned drop probability is low. This FC and drop probability classification corresponds to a best-effort treatment.

  - If there is no multifield classifier or DSCP classifier policy bound to the ingress subinterface, the FC and drop probability are determined from the default DSCP classifier policy. See Table 1: System default DSCP classifier policy.
- **For VLAN-encapsulated, non-IP packets**:
  - If the packet matches a dot1p (IEEE 802.1p) classifier policy configured on the VLAN subinterface, the FC and drop probability are determined from that policy.
  - If there is no matching dot1p classifier policy, or no dot1p policy is explicitly bound to the VLAN subinterface, the FC and drop probability are determined from the default dot1p policy.

3. Both IP and non-IP traffic can be directed to a subinterface traffic policer. In this case, packets are metered to determine compliance with a traffic profile. At the output of the policer, every packet is marked with a color (green, yellow, or red) that represents whether it conforms, exceeds, or violates the traffic profile. The drop probability for all packets can then be updated based on their conformance to the policy, and violating (red) packets can be dropped altogether.
4. A forwarding lookup on the packet determines its egress port.
5. On the 7250 IXR, if the packet is a unicast packet, it is associated with a Virtual Output Queue (VOQ) based on the ingress port, egress port, and FC.

   On a 7220 IXR-D2/D2L, D3/D3L, D4, and D5 or 7220 IXR-H2, H3, H4, and H5, the packet is associated directly with an Egress Queue (EGQ) of the egress port, based on the FC of the packet and its type (either unicast or multicast).
6. While the packet waits for its VOQ or EGQ to be serviced, the packet is stored in buffer memory. The total amount of buffer memory varies by platform.
7. The packet is dropped if the buffer memory is close to full or if the MBS of the VOQ or EGQ is exceeded.

   The MBS is one of the parameters that is configurable in a buffer allocation profile. When a buffer allocation profile is applied to a set of queues, all of those queues have the MBS value specified in the profile. If the MBS is not specified in a buffer allocation profile, the default value is platform dependent. The MBS is not a guaranteed allocation of buffer memory.
8. When the packet is Explicit Congestion Notification (ECN)-capable, and the VOQ or EGQ has an active ECN slope that applies to the packet, the ECN field may be remarked depending on the current (weighted) queue depth.
   - If the current queue depth is below the configured minimum threshold of the ECN slope, the ECN field of the packet is unchanged.
   - If the current queue depth is above the configured maximum threshold of the ECN slope, the ECN field of the packet is marked as Congestion Experienced (CE), ECN = 11.

- If the current queue size is between the minimum threshold and maximum threshold of the ECN slope, the ECN field of the packet is marked as CE, ECN = 11, based on a probability function that increases linearly from 0% at the minimum threshold to *n*% at the maximum threshold, where *n* is the operational **max-drop-probability-percent** of marking the packet.

  Note: The operational values of the **max-drop-probability-percent** may be significantly different from the configured values based on internal hardware calculations. You can check the hardware-configured values for any slope calculations.

9. When the packet is non-ECN-capable (the ECN field is zero) and the egress queue has an active WRED slope for the drop probability of the packet, the packet may be dropped by the WRED algorithm, which operates as follows:
   - If the current queue depth is below the configured minimum threshold of the WRED slope, the packet is admitted to the queue.
   - If the current queue depth is above the configured maximum threshold of the WRED slope, the packet is dropped.
   - If the current queue size is between the minimum threshold and maximum threshold of the WRED slope, the packet is dropped based on a probability function that increases linearly from 0% at the minimum threshold to *n*% at the maximum threshold, where *n* is the operational **max-drop-probability-percent** of dropping the packet.

     Note: The operational values of the **max-drop-probability-percent** may be significantly different from the configured values based on internal hardware calculations. You can check the hardware configured values for any WRED slope calculations.
10. Each unicast queue and each multicast queue of an egress port is associated with a scheduler node. The mapping of queues to scheduler nodes is platform-dependent. See Output queue scheduler policies.
11. Each egress queue can be individually configured with a peak information rate (PIR). The PIR is configured as a percentage of the egress port bandwidth.

    By default, the PIR of each queue is 100%. The operational PIR is stored by the **peak-rate-bps** leaf in bits per second. The bits counted in this rate include the Layer 2 framing of the packet (including the 14-byte Ethernet header, the 4-byte VLAN header, and the 4-byte CRC) but exclude the 20-byte Layer 1 overhead (SFD, preamble, IPG).
12. The DSCP field in the IPv4 or IPv6 header of the outgoing packet can be rewritten. On the 7250 IXR, the DSCP field must be rewritten when ECN is enabled and the packet ECN field is nonzero. When there is a rewrite policy applied, the DSCP in the outgoing packet is based on the FC (and potentially also the drop probability) of the packet. If the FC (and drop-probability) matches an entry in the applied policy, the new DSCP value is based on the policy entry. If there is no matching entry in the applied policy, the new DSCP value is 0.
13. For VLAN-tagged traffic, the PCP field in the 802.1p header of the outgoing packet can be rewritten. When there is a dot1p marking policy applied to a subinterface, the dot1p value in the outgoing packet is based on the FC (and potentially also the drop probability) of the packet. If the FC (and drop-probability) matches an entry in the applied policy, the new PCP value is based on the policy entry.
    - On a bridged subinterface, if there is no matching entry in the applied policy, all pushed 802.1Q VLAN tags on the outgoing frame are marked with a PCP value of `0`.

- On a routed subinterface, if there is no dot1p policy applied, the forwarding class index from the ingress classification is encoded into the PCP field.

System default DSCP classifier policy

Table 1: System default DSCP classifier policy

DSCP values	Included DSCP names	Forwarding class	Drop probability
0, 2 to 7	CS0/BE	fc0	Low
1	LE	fc0	High
8 to 11	CS1, AF11	fc1	Low
12 to 13	AF12	fc1	Medium
14 to 15	AF13	fc1	High
16 to 19	CS2, AF21	fc2	Low
20 to 21	AF22	fc2	Medium
22 to 23	AF23	fc2	High
24 to 27	CS3, AF31	fc3	Low
28 to 29	AF32	fc3	Medium
30 to 31	AF33	fc3	High
32 to 35	CS4, AF41	fc4	Low
36 to 37	AF42	fc4	Medium
38 to 39	AF43	fc4	High
40 to 47	CS5, EF	fc5	Low
48 to 55	CS6/NC1	fc6	Low
56 to 63	CS7/NC2	fc7	Low

3.2 How QoS works for VXLAN traffic

VXLAN QoS on 7220 IXR-D2/D2L, D3/D3L, D4, and D5

When a 7220 IXR-D2/D2L, D3/D3L, D4, or D5 receives a terminating Virtual eXtensible LAN (VXLAN) packet on a given subinterface, it classifies the packet to one of eight forwarding classes and one of three drop probabilities (low, medium, or high). The classification is based on the following considerations:

- The outer IP header DSCP is ignored.
- If the payload packet is non-IP, the classified FC index is 0 and the classified drop probability is low.

- If the payload packet is IP, and the **qos classifiers vxlan-default** command references a classifier policy, that policy is used to determine the FC and drop probability from the header fields of the payload packet.
- If the payload packet is IP, and the **qos classifiers vxlan-default** command does not reference a classifier policy, the default DSCP classifier policy is used to determine the FC and drop probability from the header fields of the payload packet.
- If a dot1p policy is applied on the subinterface, then the PCP field is set to 0. If no dot1p policy is applied, then the FC index value from the ingress classification is encoded into the PCP field.

When the 7220 IXR-D2/D2L, D3/D3L, D4, or D5 adds VXLAN encapsulation to a packet and forwards it out from a subinterface, the inner header IP DSCP value is not modified if the payload packet is IP, even if the egress-routed subinterface has a DSCP rewrite rule policy bound to it that matches the packet FC and drop probability.

On the 7220 IXR-D2/D2L and D3/D3L, if a DSCP rewrite policy is bound to the egress-routed subinterface, that policy modifies the outer header IP DSCP. If no DSCP rewrite policy is configured on the subinterface, then by default the outer header IP DSCP is copied from the inner header IP DSCP.

On the 7220 IXR-D4/D5, if a DSCP rewrite rule policy is applied to a subinterface, it has no effect on the VXLAN originated traffic. On these platforms, you must use the **qos rewrite-rules vxlan-outer-header-dscp-policy** command to explicitly associate a rewrite policy to the VXLAN originated traffic. If no VXLAN DSCP policy is configured on the subinterface, then by default the following platform-specific behavior applies:

1. On 7220 IXR-D4: the outer header IP DSCP is copied from the inner header IP DSCP.
2. On 7220 IXR-D5: the outer header IP DSCP is marked 0.

Note: If transit VXLAN traffic arrives on a subinterface with a configured subinterface traffic policer, it is policed the same as any other transit traffic. But if the VXLAN traffic terminates on the subinterface, the policing does not apply.

VXLAN QoS on 7250 IXR Gen 2 and 7250 IXR Gen 2c+

When a 7250 IXR Gen 2 or 7250 IXR Gen 2c+ receives a terminating VXLAN packet on a given subinterface, it classifies the packet to one of eight forwarding classes and one of three drop probabilities (low, medium, or high). The classification is based on the outer IP header DSCP.

When the system adds VXLAN encapsulation to a packet and forwards it out from a subinterface, the inner header IP DSCP value is not modified if the payload packet is IP, even if the egress-routed subinterface has a DSCP rewrite rule policy bound to it that matches the packet FC and drop probability.

If a DSCP rewrite policy is bound to the egress-routed subinterface, that policy modifies the outer header IP DSCP. If no DSCP rewrite policy is configured on the subinterface, then by default the outer header IP DSCP is copied from the inner header IP DSCP.

3.3 How QoS works for router-terminated traffic

This section describes how QoS applies to traffic that terminates on SR Linux systems.

1. A packet is received on a subinterface and is determined to need extraction toward the CPM. The packet is directed to one of the queues associated with the CPM as a destination based on its protocol and type. Different traffic types have their own independent queue, for example:
   - sFlow

- ICMPv4 ping
- Bidirectional Forwarding Detection (BFD)
- ARP
- ICMPv6 neighbour solicitation and neighbor advertisement
- BGP
- gRPC Remote Procedure Calls (gRPC)
- Link Layer Discovery Protocol (LLDP)
- IPv4 packets with IP options and IPv6 packets with extension headers
- DHCPv6
- IS-IS hello PDUs
- OSPF/OSPFv3 hello PDUs

2. Some of the queues toward the CPM have a PIR shaping rate designed to prevent an overload of one type of traffic. The PIR shaping rates vary by platform.

3.4 How QoS works for router-originated traffic

This section describes how QoS applies to traffic that originates on SR Linux systems.

1. An application on the SR Linux CPM has an IPv4 or IPv6 packet to send to another system.
2. The CPM datapath assigns a DSCP to the self-generated packet based on its protocol and the default mapping shown in Table 2: Default forwarding class and DSCP marking for router-originated traffic. To modify the common DSCP value used for some router-originated management protocols, see Configuring DSCP for management protocols.

   For originated ICMP and ICMPv6 echo-request packets, the DSCP override value can be configured as an optional parameter of the **ping** command.
3. The CPM datapath looks up the DSCP from the previous step (either the fixed value or the override value for echo-request) in the default DSCP classifier policy (see Table 1: System default DSCP classifier policy) to determine the FC and drop probability level.
4. A forwarding lookup determines the egress port.
5. On the 7250 IXR, the packet is sent to the egress line card and added to a VOQ appropriate for its forwarding class and the egress port. The decision to drop or enqueue the packet in the VOQ and the scheduling of the VOQ follows the previous description for transit traffic. There is no scheduling differentiation between router-originated traffic and transit traffic of the same FC on the egress IMM.
6. The packet is directed to the egress queue appropriate for its forwarding class and packet type. On the 7220 IXR-D2, D3, D4, and D5 and the 7220 IXR-H2 and H3, the decision to drop or enqueue the packet in the egress queue and the scheduling of the egress queue follow QoS treatment of transit traffic described in How QoS works for transit traffic.
7. The DSCP field in the IPv4 or IPv6 header is always written based on the hard-coded mapping described in Table 2: Default forwarding class and DSCP marking for router-originated traffic. If the packet also matches a DSCP policy rewrite rule or a dot1p rewrite rule applied to the output subinterface, the rewrite-rule policy is ignored.

Default forwarding class and DSCP marking for router-originated traffic

Table 2: Default forwarding class and DSCP marking for router-originated traffic

Protocol and message type	Forwarding class index	Drop probability	DSCP marking
IPv4 ARP request/reply	6	Low	—
ICMPv4 including echo-request[1], echo-reply[2], dest-unreachable, redirect, time-exceeded, parameter-problem	0	Medium	0
ICMPv4 echo-request with ToS/DSCP override = *x*	Look up *X* in system-default DSCP classifier	Look up *X* in system-default DSCP classifier	*x*
ICMPv4 echo-reply to echo-request with nonzero DSCP *x*	Look up *X* in system-default DSCP classifier	Look up *X* in system-default DSCP classifier	*x*
UDP traceroute	0	Low	0
IPv6 neighbor solicitation	6	Low	48 (CS6/NC1)
IPv6 neighbor advertisement	6	Low	48 (CS6/NC1)
All other ICMPv6 messages including dest unreachable, packet-too-big, time-exceeded, parameter-problem, echo-request, echo-reply, router-solicitation, redirect	0	Medium	0
ICMPv6 echo-request with DSCP override = *x*	Look up *x* in system-default DSCP classifier	Look up *x* in system-default DSCP classifier	*x*
ICMPv6 echo-reply to echo-request with nonzero DSCP *x*	Look up *x* in system-default DSCP classifier	Look up *x* in system-default DSCP classifier	*x*
BFD	6	Low	48 (CS6/NC1)
BGP	6	Low	48 (CS6/NC1)
DNS query	4	Low	32 (CS4)
FTP/TFTP	4	Low	32 (CS4)
gNMI	4	Low	32 (CS4)
gNOI	4	Low	32 (CS4)
gRIBI	4	Low	32 (CS4)

1 Echo-request generated by a ping command with no DSCP parameter specified.

2 Echo-reply to an echo-request packet with DSCP = 0.

Protocol and message type	**Forwarding class index**	**Drop probability**	**DSCP marking**
JSON RPC	4	Low	32 (CS4)
LLDP	—	Low	—
NTP	4	Low	32 (CS4)
P4RT	4	Low	32 (CS4)
RADIUS	4	Low	32 (CS4)
sFlow	0	Low	32 (CS4)
SNMP	4	Low	32 (CS4)
SSH	4	Low	32 (CS4)
Syslog	4	Low	32 (CS4)
TACACS+	4	Low	32 (CS4)

4 QoS interface and subinterface IDs

To enable QoS features on an SR Linux interface or subinterface, you must first configure an interface ID using the following command:

- **qos interfaces interface** *<interface-id>*

You can then associate the custom interface ID with a physical interface and logical subinterface using the following command:

qos interfaces interface *<interface-id>* **interface-ref interface** *<interface-name>* **subinterface** *<subinterface-number>*

Where <interface-name> refers to a base interface, such as a port or LAG and <subinterface-number> specifies the subinterface value.

After the interface or subinterface is defined, you can then assign the QoS policies as required.

Default QoS interface and subinterface IDs

An interface or subinterface can exist under the **/interface** context without an explicitly declared entry under the **/qos interfaces** context. In this case, the interface or subinterface inherits the system default QoS configuration and associated default queues. The system also creates a default interface ID as follows:

1. interface **ethernet-x/y**; where **x/y** refers to the physical port
2. subinterface **ethernet-x/y.z**; where **z** refers to the subinterface index value

If you subsequently configure a QoS interface ID that references an interface with an existing default interface or subinterface ID, the default interface ID is replaced.

If the configured interface ID matches an automatically generated ID, but references a different interface, the automatically created ID is prepended with an underscore, for example: **_ethernet-1/1**.

5 Named queues and forwarding classes

SR Linux provides support for both named queues and named forwarding classes.

By default, forwarding classes have system-reserved names, `fc0` to `fc7`, that map to system-reserved unicast queues `unicast-0` to `unicast-7` and to multicast queues on applicable platforms.

SR Linux provides the flexibility to do the following:

- assign each queue a string name and index value
- assign each forwarding class a string name and index value
- map the named forwarding class to a named queue

Implementation details

The following implementation details apply to named queues and forwarding classes:

- Named queues and named forwarding classes are *not* automatically created under the `/qos` container configuration.

  Note: In case of upgrades from a previous release, the default forwarding classess (`fc0` to `fc7`) are automatically created and mapped to the default queues since that configuration is mandatory for configuration of any QoS policy.

- Even though they do not appear as named forwarding classes in the configuration, the default forwarding class names `fc0` to `fc7` always exist and are reserved names.
- Even though they do not appear as named queues in the configuration, the default queue names `unicast-0` to `unicast-7` always exist and are reserved names. (On applicable platforms, default multicast queues are also reserved names.)
- Every interface always has a full set of egress queues; only the names of the queues are variable.
- If an interface has no explicit configuration for a default queue, and no named queue associated with that queue index, SR Linux displays the queue name in the output as the default value (`unicast-0` to `unicast-7`) with default parameters.
- If you configure a named forwarding class (for example, `forwarding-class-A`) and assign it a queue that references queue index 3, any subsequent configuration that references the default forwarding class name (in this case, `fc3`) fails. You must always reference the named forwarding class when it is configured.

5.1 Configuring named queues

Procedure

Use the **qos queues queue** command to configure a name and index for a queue. When you configure a named queue, it remains an inactive configuration that cannot be referenced by any interface until you map it to a forwarding class.

Queues with a higher index are serviced more preferentially than queues with a lower index (subject to scheduler configuration).

Example: Configure queue name

```
# info with-context qos queues
qos {
        queues {
            queue unicast-queue-1 {
                queue-index 0
            }
            queue multicast-queue-1 {
                queue-index 1
            }
        }
    }
```

5.2 Configuring forwarding class names and queue associations

Procedure

Use the **qos forwarding-classes forwarding-class** <*name*> command to assign a name and output queue to a forwarding class.

You must associate the forwarding class with a unicast queue. All of the following parameters are mandatory: the **forwarding-class** *name*, and the **unicast-queue**.

Example: Configure forwarding class name, and queue association

```
# info with-context qos forwarding-classes
    qos {
        forwarding-classes {
            forwarding-class test-fc {
                output {
                    unicast-queue unicast-queue-1
                }
            }
        }
    }
```

You can reference the named forwarding class in policies including DSCP classifier and rewrite, dot1p classifier and rewrite, multifield classifier and rewrite, MPLS traffic-class and rewrite, and the ingress subinterface policer template.

6 Default forwarding class and drop probability on bridged interfaces

Prerequisites

To reference a forwarding-class in any QoS policy, the forwarding-class must first be explicitly mapped to an output queue. For information about mapping the named forwarding classes to named queues, see Named queues and forwarding classes.

Procedure

On bridged interfaces, you can configure the default forwarding class and drop probability for input packets arriving on a subinterface that do not match any classification rule.

Example: Configure default drop probability and forwarding class

```
# info with-context qos interfaces interface ethernet-1/1 input classifiers default
    qos {
        interfaces {
            interface ethernet-1/1 {
                input {
                    classifiers {
                        default {
                            forwarding-class test-fc
                            drop-probability medium
                        }
                    }
                }
            }
        }
    }
```

7 Multifield classification policies

SR Linux supports rule-based QoS multifield classification of IPv4 and IPv6 packets. Each IPv4 and IPv6 multifield classification policy is structurally similar to an IPv4 or IPv6 interface ACL, containing a list of ordered entries, each specifying a set of match conditions and associated actions.

Each multifield classification rule, or entry, has a sequence ID. The policy evaluates packets starting with the entry with the lowest sequence ID, progressing to the entry with the highest sequence ID. Evaluation stops at the first matching entry (that is, when the packet matches all of the conditions specified by the multifield classification entry).

Multifield classification policies are supported on the following platforms:

- 7220 IXR-D2/D2L/D3/D3L/D4/D5
- 7250 IXR-6/6e/10/10e/X1b/X3b

Match conditions

Each IPv4 or IPv6 policy entry can specify zero or more of the following match conditions.

Table 3: Multifield classification match conditions

Match condition	Description	IPv4 policy support	IPv6 policy support
Destination IP	Matches by prefix or by address and mask	✔	✔
Destination port	Matches by destination TCP or UDP port or range. Comparison operators define whether the matching destination port must be: • equal to the specified value • greater than or equal to the specified value • less than or equal to the specified value	✔	✔
DSCP set	Matches one of the DSCP values listed. This setting matches against the ingress DSCP value (not the rewritten DSCP value). If left empty, any DSCP value matches.	✔	✔
Fragment/first-fragment	Matches a packet that is a fragment, and optionally the first fragment	✔	—
ICMP type/code	Matches one of the specified ICMP type and code combinations	✔	—
ICMPv6 type/code	Matches one of the specified ICMPv6 type and code combinations	—	✔
Next-header number	Matches the first next-header field (in the IPv6 fixed header) if it contains the specified value	—	✔

Match condition	Description	IPv4 policy support	IPv6 policy support
Protocol number	Matches the IP protocol type field	✔	—
Source IP	Matches by prefix or by address and mask	✔	✔
Source port	Matches source TCP or UDP port or range. Comparison operators define whether the matching source port must be: • equal to the specified value • greater than or equal to the specified value • less than or equal to the specified value	✔	✔
TCP flags	Matches the TCP flag names: RST, SYN, and ACK based on a logical expression using the &, |, and ! operators	✔	✔

Supported actions

Each IPv4 or IPv6 policy entry supports the following actions:

- set the forwarding class (mandatory action in each entry)
- set the drop probability (optional action in each entry, default is low)
- rewrite the ingress DSCP value (optional action in each entry, supported only on the 7220 IXR-D2/D2L/D3/D3L)

7.1 Supported interfaces: routed, bridged, and IRB

You can bind a multifield classification policy (IPv4, IPv6, or both) to the following subinterface types:

- routed subinterface of a default or ip-vrf network instance, associated with an Ethernet port, LAG, or IRB
- bridged subinterface of a mac-vrf network instance, associated with an Ethernet port or LAG

Note: 7250 IXR-6/6e/10/10e/X1b/X3b platforms do not support multifield classification on bridged subinterfaces.

DSCP classification policy and multifield classifier policy on the same subinterface

You can apply both a DSCP classification policy and a multifield classifier policy to the same IP/routed subinterface for a specified protocol (IPv4 or IPv6). If an ingress IPv4 or IPv6 packet matches a multifield classification rule, its forwarding class and drop probability are determined solely by the matching multifield classification rule. If an ingress IPv4 or IPv6 packet does not match any multifield classification rule, forwarding class and drop probability are determined as follows:

- **On 7220 IXR-D2/D2L/D3/D3L/D4/D5**:
  Forwarding class and drop probability are determined by the configured or default DSCP policy.
- **On 7250 IXR-6/6e/10/10e/X1b/X3b:**

Forwarding class and drop probability are determined by the configured or default IPv4 DSCP policy (for IPv4 packets) or IPv6 DSCP policy (for IPv6 packets).

7.2 Scaling and restrictions

The following describe scaling and restrictions for multifield classification policies.

7220 IXR-D2/D2L/D3/D3L/D4/D5

On the 7220 IXR-D2/D2L/D3/D3L/D4/D5:

- Multifield classifier policies always operate in subinterface-specific mode, with no option available for a shared mode. As a result, the number of TCAM entries required to implement one multifield classifier policy is $N \times S$, where N is the number of TCAM entries required to implement one instance of the policy and S is the number of subinterfaces where the policy is applied.
- SR Linux blocks the binding of a MAC ACL and an IPv4 or IPv6 multifield classifier policy on the same subinterface. MAC ACL and multifield classification are mutually exclusive options.

7250 IXR/6e/10/10e/X1b/X3b

On the 7250 IXR/6e/10/10e/X1b/X3b:

- Multifield classifier policies cannot operate in a subinterface-specific mode, with no option available to create subinterface-specific TCAM entries. As a result, the number of TCAM entries required to support one multifield classifier policy applied across S subintefaces is just N, where N is the number of TCAM entries required to implement one instance of the policy.
- A maximum of 15 IPv4 and 15 IPv6 multifield classifier instances are supported, with utilization reported under **info from state platform linecard** *slot* **forwarding-complex** *name* **acl resource [input-ipv4-filter-instances | input-ipv6-filter-instances]**.

7.3 Ingress DSCP rewrite (7220 IXR-D2/D2L/D3/D3L)

Ingress DSCP rewrite is supported only on the 7220 IXR-D2/D2L/D3/D3L.

Packets arriving on an interface can have IP DSCP markings that are not trusted. For example, when the upstream devices do not classify or mark the packets properly, or when the interface is at the beginning of a service SLA that is defined in terms of application characteristics instead of DSCP. In this case, an ingress DSCP rewrite action in the multifield classification policy can replace the DSCP value for matching IPv4 or IPv6 packets with a new value.

Note: If an egress DSCP rewrite rule is also applied to a Layer 3 subinterface, it does not overwrite the ingress DSCP rewrite action. In this case, the packet is transmitted with the DSCP specified in the ingress DSCP rewrite rule.

The following table provides more information about the packet flows that are supported with ingress DSCP rewrite.

Table 4: Supported packet flows with ingress DSCP rewrite

Ingress packet	Ingress subif type	Ingress subif MF classifier entry action	Forwarding	IRB subif MF classifier entry action	Egress subif DSCP rewrite policy	Egress packet
IP/ Ethernet	Bridged (mac-vrf)	Set fc = A dscp-rewrite = B	L2 switched	Configured or not configured (no effect in either case)	Bridged subif DSCP rewrite policy: no effect	DSCP = B
IP/ Ethernet	Bridged (mac-vrf)	Set fc = A dscp-rewrite = B	L3 routed between mac-vrf1 and mac-vrf2 using IRB	Not configured	mac-vrf2 IRB subif DSCP rewrite policy: no effect mac-vrf2 bridged subif DSCP rewrite policy: no effect	DSCP = B
IP/ Ethernet	Bridged (mac-vrf)	Set fc = A dscp-rewrite = B	L3 routed between mac-vrf1 and mac-vrf2 using IRB	IRB of mac-vrf1: set fc = C dscp-rewrite = D	mac-vrf2 IRB subif DSCP rewrite policy: no effect mac-vrf2 bridged subif DSCP rewrite policy: no effect	DSCP = D
IP/ Ethernet	Bridged (mac-vrf)	Set fc = A dscp-rewrite = B	L3 routed followed by VXLAN encap (symmetric or asymmetric)	Not configured	Routed subif DSCP rewrite policy: only changes outer DSCP	VXLAN with outer DSCP based on fc = A lookup in the DSCP rewrite policy, payload DSCP = B
IP/ Ethernet	Bridged (mac-vrf)	Set fc = A dscp-rewrite = B	L3 routed followed by VXLAN encap (symmetric or asymmetric)	IRB of mac-vrf1: set fc = C dscp-rewrite = D	Routed subif DSCP rewrite policy: only changes outer DSCP	VXLAN with outer DSCP based on fc = C lookup in the DSCP rewrite policy, payload DSCP = D
IP/ Ethernet	Routed (ip-vrf or default)	Set fc = A dscp-rewrite = B	L3 routed	—	Routed subif DSCP rewrite policy: no effect	DSCP = B

7.4 Configuring multifield classification policies for input traffic

Procedure

To create a multifield classification policy, define either an IPv4 or IPv6 policy name using the **qos classifiers multifield-classifier** command. Within the named policy, configure one or more entries that consist of match conditions and the associated action to apply to matching packets.

The following examples create IPv4 and IPv6 multifield classifier policies, each containing one entry with multiple match conditions and associated actions.

Note: The **rewrite set-dscp** parameter is supported only on the 7220 IXR-D2/D2L/D3/D3L. Also the forwarding class (and associated queue) referenced by the **action** parameter must already be configured for the policy entry to be successfully committed.

Example: Configure IPv4 multifield classification policy

```
--{ candidate shared default }--[  ]--
# info with-context qos classifiers multifield-classifier mf-classifier-test-v4
    qos {
        classifiers {
            multifield-classifier mf-classifier-test-v4 {
                type ipv4
                entry 10 {
                    match {
                        ipv4 {
                            fragment true
                            first-fragment true
                            protocol tcp
                            dscp-set [
                                AF11
                            ]
                            destination-ip {
                                prefix 10.10.20.0/24
                            }
                            source-ip {
                                address 10.10.10.1
                                mask 255.255.255.0
                            }
                        }
                        transport {
                            tcp-flags syn&ack
                            destination-port {
                                operator eq
                                value 25
                            }
                            destination-port {
                                operator eq
                                value 25
                            }
                            source-port {
                                operator ge
                                value 2526
                            }
                        }
                    }
                    action {
                        forwarding-class test-fc6
                        drop-probability low
```

```
                    }
                }
            }
        }
    }
```

Example: Configure IPv6 multifield classification policy

```
--{ candidate shared default }--[  ]--
# info with-context qos classifiers multifield-classifier mf-classifier-test-v6
    qos {
        classifiers {
            multifield-classifier mf-classifier-test-v6 {
                type ipv6
                entry 100 {
                    match {
                        ipv6 {
                            next-header tcp
                            dscp-set [
                                CS7
                            ]
                            destination-ip {
                                prefix 2001:db8:fe10::/64
                            }
                            source-ip {
                                prefix 2001:db8:fc00::/64
                            }
                        }
                        transport {
                            destination-port {
                                range {
                                    start 800
                                    end 1000
                                }
                            }
                            source-port {
                                operator le
                                value 700
                            }
                        }
                    }
                    action {
                        forwarding-class test-fc7
                        drop-probability medium
                            rewrite {
                                set-dscp 56
                        }
                    }
                }
            }
        }
    }
```

7.5 Applying a multifield classification policy to a subinterface

Procedure

To apply an IPv4 or IPv6 multifield classification policy (or both) to a subinterface, use the **qos interfaces interface input classifiers classifier** command.

The following example applies the IPv4 and IPv6 multifield classification policies to inbound traffic on subinterface ethernet-1/1.1.

Example: Apply multifield classification policy to subinterface

```
--{ candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/1
    qos {
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
                input {
                    classifiers {
                        classifier ipv4 {
                            name mf-classifier-test-v4
                        }
                        classifier ipv6 {
                            name mf-classifier-test-v6
                        }
                    }
                }
            }
        }
    }
```

8 DSCP classifier policy configuration for input traffic

When a DSCP classifier policy is applied to a subinterface, the policy attempts to match the 6-bit DSCP value in the IP header of incoming packets to one of its entries. If there is a match, the incoming packet is assigned to the specified forwarding class and drop probability; otherwise, the assigned forwarding class is 0 and the assigned drop probability is low.

Packets that require a similar treatment (per-hop behavior) are grouped into an FC, also known as a behavior aggregate. 7220 IXR and 7250 IXR platforms differentiate up to eight forwarding classes.

The drop probability can be one of high, medium, or low. If a queue management profile with different WRED slopes is bound to a queue, then packets in that queue with a high drop probability are the first to be dropped when the queue experiences congestion, followed by packets with a medium drop probability, then by packets with a low drop probability. The default is low.

8.1 Configuring DSCP classifier policies

Prerequisites

To reference a forwarding-class in any QoS policy, the forwarding-class must first be explicitly mapped to an output queue. For information about mapping the named forwarding classes to named queues, see Named queues and forwarding classes.

Procedure

To configure a DSCP classifier policy, set custom **forwarding-class** and **drop-probability** values to apply to one or more incoming DSCP values using the **qos classifiers dscp-policy** command.

The following example creates a DSCP classifier policy:

Example

```
--{ candidate shared default }--[  ]--
# info with-context qos classifiers
    qos {
        classifiers {
            dscp-policy new-policy {
                dscp 0 {
                    forwarding-class forwarding-class-0
                    drop-probability high
                }
                dscp 8 {
                    forwarding-class forwarding-class-1
                    drop-probability high
                }
            }
        }
    }
```

Note: To create a new DSCP classification policy based on the default policy, you can copy the default policy from state in candidate mode, as shown in the following example:

```
# copy from state /qos classifiers dscp-policy default to /qos classifiers dscp-
policy test
```

8.2 Using a DSCP classifier for VXLAN traffic

About this task

On 7220 IXR systems, you can use a classifier policy to classify ingress packets received from any remote VXLAN VTEP. The policy applies to payload packets after VXLAN decapsulation is performed.

Procedure

To apply a DSCP classifier to all VXLAN traffic, specify a configured DSCP policy under the **qos classifiers vxlan-default** context.

The following example shows how the DSCP classifier policy created in the previous example (**new-policy**) can be applied for VXLAN traffic:

Example

```
--{ candidate shared default }--[  ]--
# info with-context qos classifiers vxlan-default
    qos {
        classifiers {
            vxlan-default new-policy
        }
    }
```

8.3 DSCP classifier policy application to subinterfaces

If you apply a DSCP classifier policy to input traffic on a subinterface, incoming packets are evaluated against the policy, and matching packets are assigned to the forwarding class and drop probability specified by the policy. If no classifier policy is applied to the subinterface, the system default DSCP classifier (with the reserved name *default*) is used.

8.3.1 Applying a DSCP classifier policy to input traffic (7250 IXR)

Procedure

On the 7250 IXR, to apply a DSCP classifier to input traffic on a subinterface, specify an IPv4 or IPv6 DSCP policy (or both) using the **qos interfaces interface input classifiers** command.

The following example applies DSCP classifier policies to inbound IPv4 and IPv6 traffic on a subinterface of a 7250 IXR system:

Example: Apply a DSCP classifier to input traffic (7250 IXR)

```
--{ candidate shared default }--[  ]--
```

```
# info with-context qos interfaces interface ethernet-1/1
    qos {
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
                input {
                    classifiers {
                        ipv4-dscp-policy new-v4-policy
                        ipv6-dscp-policy new-v6-policy
                        }
                    }
                }
            }
        }
    }
```

8.3.2 Applying a DSCP classifier policy to input traffic (7220 IXR)

Procedure

On the 7220 IXR, to apply a DSCP classifier to input traffic on a subinterface, specify a DSCP policy using the **qos interfaces interface input classifiers** command.

Note: The 7220 IXR systems do not support separate classifier policies for IPv4 and IPv6 traffic, but you can apply a common policy that applies to both IPv4 and IPv6 traffic.

The following example applies a DSCP classifier policy to inbound traffic on a subinterface of a 7220 IXR system:

Example: Apply a DSCP classifier to input traffic (7220 IXR)

```
# info with-context qos interfaces interface ethernet-1/1
    qos {
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
                input {
                    classifiers {
                        dscp-policy new-v4-v6-policy
                        }
                    }
                }
            }
        }
    }
```

9 DSCP rewrite-rule policy configuration for output traffic

When a DSCP rewrite-rule policy is applied to a subinterface, the policy attempts to match the forwarding class (and optionally the drop-probability) of outbound packets to one of its entries. If there is a match, the DSCP value of the outbound packet is changed to the value specified by the policy. If the forwarding class of the packet does not match a rule in the rewrite-rule policy, the DSCP value is changed to 0.

On 7220 IXR and 7250 IXR systems, if no DSCP rewrite-rule policy is applied to a subinterface, the incoming packet's DSCP remains unchanged at egress.

9.1 Configuring DSCP rewrite-rule policies

Prerequisites

To reference a forwarding-class in any QoS policy, the forwarding-class must first be explicitly mapped to an output queue. For information about mapping the named forwarding classes to named queues, see Named queues and forwarding classes.

Procedure

To configure a DSCP rewrite-rule, define the policy name using the **qos rewrite-rules dscp-policy** command. Within the policy, configure one or more forwarding class (and optionally drop-probability) match conditions and the associated DSCP value to apply to the matching packets.

The following example creates a rewrite-rule policy:

Example

```
--{ candidate shared default }--[  ]--
# info with-context qos rewrite-rules
    qos {
        rewrite-rules {
            dscp-policy normalize {
                map forwarding-class-0 {
                    dscp 7
                }
                map forwarding-class-1 {
                    dscp 10
                    drop-probability low {
                        dscp 11
                    }
                    drop-probability high {
                        dscp 13
                    }
                }
                map forwarding-class-2 {
                    dscp 23
                }
                map forwarding-class-3 {
                    dscp 31
                }
            }
```

```
        }
    }
```

9.2 Using a DSCP rewrite-rule for VXLAN traffic (7220 IXR-D2/D3/D4/D5 and 7250 IXR Gen 2/Gen2c+)

About this task

You can configure policies to modify the outer IP DSCP for VXLAN traffic as follows:

- **7220 IXR-D2/D2L/D3/D3L, 7250 IXR Gen 2, and 7250 IXR Gen 2c+.**

  On 7220 IXR-D2/D2L/D3/D3L, 7250 IXR Gen 2, and 7250 IXR Gen 2c+ platforms, if you configure a DSCP rewrite rule policy on the egress routed subinterface, this same policy modifies the outer IP DSCP value for the VXLAN traffic also.

  If no DSCP rewrite policy is configured on the subinterface, then by default, the inner header IP DSCP value is not modified, and the outer header IP DSCP is copied from the inner header IP DSCP.

- **7220 IXR-D4/D5**

  On 7220 IXR-D4/D5, if a DSCP rewrite rule policy is applied to a subinterface, it has no effect on the VXLAN originated traffic. On these platforms, you must use the **qos rewrite-rules vxlan-outer-header-dscp-policy** command to explicitly associate a rewrite policy to the VXLAN originated traffic.

  If no VXLAN DSCP policy is configured on the subinterface, then by default, the inner header IP DSCP value is not modified, and the following platform-specific behavior applies:

  – on 7220 IXR-D4: the outer header IP DSCP is copied from the inner header IP DSCP

  – on 7220 IXR-D5: the outer header IP DSCP is marked 0

Procedure

On 7220 IXR-D4/D5 systems, use the **qos rewrite-rules vxlan-outer-header-dscp-policy** command to apply a rewrite-rule policy for all VXLAN traffic, as shown in the following example:

Example: Apply a DSCP rewrite-rule for VXLAN traffic on 7220 IXR-D4/D5

```
--{ candidate shared default }--[  ]--
# info with-context qos rewrite-rules
    qos {
        rewrite-rules {
            vxlan-outer-header-dscp-policy vxlan-rewrite-test
            dscp-policy vxlan-rewrite-test
            }
        }
```

9.3 Rewrite-rule policy application to subinterfaces

When a rewrite-rule policy is applied to output traffic on a subinterface, outbound packets are evaluated against the policy. The policy subjects all packets to remarking, with some exceptions. If no rewrite-rule policy is applied to the subinterface, the DSCP marking of the traffic leaving the subinterface is unchanged, unless it is ECN-capable traffic forwarded by a 7250 IXR system or VXLAN traffic originated by a 7220 IXR-D2/D2L, D3/D3L, D4, and D5 system. For these exceptions, DSCP may be remarked even in the absence of a rewrite-rule policy applied to the egress subinterface.

On all platforms, rewrite-rule policies do not affect DSCP marking of self-generated traffic.

9.3.1 Applying a rewrite-rule policy to output traffic (7250 IXR)

Procedure

On the 7250 IXR, to apply a DSCP rewrite-rule to output traffic on a subinterface, specify an IPv4 or IPv6 policy (or both) using the **qos interfaces interface output rewrite-rules** command.

Note: 7250 IXR systems support separate rewrite policies for IPv4 and IPv6 egress traffic.

The following example applies a rewrite-rule policy to outbound IPv4 traffic on a subinterface with a 7250 IXR system:

Example

```
--{ candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/1
    qos {
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
                output {
                    rewrite-rules {
                        ipv4-dscp-policy new-rule
                    }
                }
            }
        }
    }
```

9.3.2 Applying a rewrite-rule policy to output traffic (7220 IXR)

Procedure

On the 7220 IXR, to apply a DSCP rewrite-rule for both IPv4 and IPv6 output traffic on a subinterface, specify a policy using the **qos output rewrite-rules** command.

Note: Common rewrite policies that apply to both IPv4 and IPv6 traffic are supported on 7220 IXR systems.

The following example applies a rewrite-rule policy to outbound traffic on a subinterface with a 7220 IXR system:

Example

```
--{ candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/1
    qos {
```

```
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
                output {
                    rewrite-rules {
                        dscp-policy new-rule
                    }
                }
            }
        }
    }
```

9.4 Configuring DSCP for management protocols

About this task

By default, SR Linux applies a common DSCP value (default value: 32) to all of the following router-originated management traffic:

- DNS query
- FTP/TFTP
- gNMI
- gNOI
- gRIBI
- JSON RPC
- NTP
- P4RT
- RADIUS
- sFlow
- SNMP
- SSH
- Syslog
- TACACS+

This common default DSCP value is configurable.

Procedure

To modify the common DSCP value for the router-originated management protocols listed above, use the **system control-plane-traffic output qos management-protocols-dscp** command.

Example: Set the DSCP value for management protocols

```
--{ + candidate shared default }--[  ]--
# info with-context system control-plane-traffic output qos
    system {
        control-plane-traffic {
```

```
            output {
                qos {
                    management-protocols-dscp 34
                }
            }
        }
    }
```

10 Dot1p classification and marking

SR Linux supports IEEE 802.1p (dot1p) classification and marking using the Priority Code Point (PCP) field. When one or more IEEE 802.1Q VLAN tags are added to an Ethernet frame, the Class of Service (CoS) of the frame can be set using the PCP field in the outermost VLAN tag. The 3-bit PCP field can specify eight different classes of service, allowing Ethernet frames to be assigned a required service level.

Supported platforms

Dot1p classification and marking is supported on the following platforms.

Table 5: Dot1p classification and marking feature support by platform

Platform	Dot1p classification	Dot1p marking
7220 IXR-D2/D2L/D3/D3L	✓	✓
7220 IXR-D4/D5	✓	✓
7220 IXR-H*x*	✓	Not supported

Note: 7250 IXR platforms support default dot1p behaviour without custom policies. However, 7250 IXR platforms cannot preserve dot1Q p-bit value on L2/L3 subinterfaces.

Note: 7220 IXR-H*x* platforms preserve dot1Q p-bit value on L2 subinterfaces.

10.1 Dot1p classification

Dot1p classification refers to classification of a frame based on the PCP field in the outermost VLAN. The system assigns a forwarding-class and drop-probability to every packet at an early point in the packet forwarding pipeline. These assignments determine which packets to schedule or drop first when congestion occurs.

Each dot1p classifier policy can contain up to eight mapping rules. Each rule binds one of the eight possible PCP values (0 to 7) to a forwarding-class (fc0 to fc7) and to a drop-probability level (low, medium, or high).

For a dot1p classifier policy to take effect, you must apply the policy to at least one bridged subinterface. SR Linux supports dot1p classifier policies on any bridged subinterface of any Ethernet port or LAG. No limit exists on the number of bridged subinterfaces that can apply the same policy. (Routed subinterfaces are not supported.) Dot1p classification is applicable for non-IP packets only.

When a dot1p classifier policy is applied to a subinterface, if the PCP value for an incoming Ethernet frame does not match any configured dot1p rule, the frame is classified as `fc0` and `drop-probability low`.

Default dot1p classifier policy

SR Linux supports a default dot1p classifier policy, which always exists and is not modifiable. It is invisibly applied to all bridged subinterfaces that do not have a configured dot1p classifier policy applied. The following table describes the rules of the default policy.

Table 6: Default dot1p classifier policy

Dot1p	FC	Drop probability
0	fc0	Low
1	fc1	Low
2	fc2	Low
3	fc3	Low
4	fc4	Low
5	fc5	Low
6	fc6	Low
7	fc7	Low

Dot1p classifier policy effects

The following table describes the effects of applying the dot1p classifier policy in relation to other configuration.

Table 7: Effect of dot1p classifier policy in relation to other configuration

Ingress sub-interface type	Ingress packet type	Default fc of ingress sub-interface	Default drop-probability of ingress subif	Default dot1p-policy	Explicit dot1p-policy (bound to ingress sub-interface)	Default dscp-policy	Explicit dscp-policy (bound to ingress sub-interface)
Routed, untagged	IPv4/IPv6 untagged	—	—	—	Not supported	Used if no explicit dscp-policy	Used
Routed, untagged	IPv4/IPv6 priority-tagged	—	—	—	Not supported	Used if no explicit dscp-policy	Used
Routed, single-tagged	IPv4/IPv6 single-tagged	—	—	—	Not supported	Used if no explicit dscp-policy	Used
Bridged, untagged	IPv4/IPv6 untagged	—	—	—	—	Used if no explicit dscp-policy	Used

Ingress sub-interface type	Ingress packet type	Default fc of ingress sub-interface	Default drop-probability of ingress subif	Default dot1p-policy	Explicit dot1p-policy (bound to ingress sub-interface)	Default dscp-policy	Explicit dscp-policy (bound to ingress sub-interface)
Bridged, untagged	IPv4/IPv6 priority-tagged	—	—	—	—	Used if no explicit dscp-policy	Used
Bridged, single-tagged	IPv4/IPv6 single-tagged	—	—	—	—	Used if no explicit dscp-policy	Used
Bridged, untagged	Non-IP untagged	Used	Used	—	—	—	—
Bridged, untagged	Non-IP priority-tagged	—	—	Used if no explicit dot1p-policy	Used	—	—
Bridged, single-tagged	Non-IP single-tagged	—	—	Used if no explicit dot1p-policy	Used	—	—
Bridged, single-tagged	Non-IP double-tagged	—	—	Used if no explicit dot1p-policy	Used	—	—

10.1.1 Configuring dot1p classifiers for input traffic

Prerequisites

To reference a forwarding-class in any QoS policy, the forwarding-class must first be explicitly mapped to an output queue. For information about mapping the named forwarding classes to named queues, see Named queues and forwarding classes.

Procedure

To configure a dot1p classifier policy, map one or more incoming dot1p values to custom **forwarding-class** and **drop-probability** values using the **qos classifiers dot1p-policy** command.

Note: The dot1p-policy name can be any name string other than `default`.

The following example creates a dot1p classifier policy:

Example: Create a dot1p classifier policy

```
--{ candidate shared default }--[  ]--
# info with-context qos classifiers
    qos {
        classifiers {
            dot1p-policy new-dot1p-policy {
                dot1p 0 {
                    forwarding-class forwarding-class-0
                    drop-probability high
                }
                dot1p 7 {
                    forwarding-class forwarding-class-7
                    drop-probability low
                }
            }
        }
    }
```

10.1.2 Applying a dot1p policy to a subinterface

Procedure

To apply a dot1p policy to input traffic on a subinterface, specify the policy using the **qos interfaces interface input classifiers** command.

The following example applies a dot1p policy to inbound traffic on a subinterface:

Example: Apply a dot1p policy to a subinterface

```
--{ * candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-1/2
    qos {
        interfaces {
            interface ethernet-1/2 {
                interface-ref {
                    interface ethernet-1/2
                    subinterface 1
                }
                input {
                    classifiers {
                        dot1p-policy new-dot1p-policy
                    }
                }
            }
        }
    }
```

10.2 Dot1p marking

Dot1p marking refers to rewriting of the PCP value in the outermost VLAN tag. The node rewrites the value in the PCP field before a packet is transmitted out an egress interface. Downstream nodes handle the remarked traffic based on the updated code point. SR Linux implements dot1p marking using dot1p rewrite policies.

Each dot1p rewrite policy contains up to eight mapping rules, and each rule associates one of the eight possible internal forwarding classes (fc0 to fc7) to a PCP value (0 to 7).

Unless explicitly configured otherwise, a new mapping rule applies to all drop-probability levels. You can optionally configure a mapping rule for a specific forwarding class and drop probability combination, as required.

For a dot1p rewrite policy to take effect, you must apply the policy to at least one subinterface. SR Linux supports rewrite policies on any bridged or routed subinterface of any Ethernet port or LAG. No limit exists on the number of subinterfaces that can apply the same policy.

When a dot1p rewrite policy is applied to a subinterface, if the forwarding class of an outgoing packet does not match any configured dot1p rewrite rule, all pushed 802.1Q VLAN tags on the outgoing frame are marked with a PCP value of `0`.

If a dot1p rewrite policy is not applied to a subinterface:

- For a routed subinterface, the dot1p value is taken from the forwarding class.
- For a bridged subinterface, the behavior is platform-dependent:
  - On 7220 IXR-Hx platforms, the dot1p value is preserved.
  - On 7220 IXR-Dx and 7250 IXR platforms, the dot1p value is 0.

No default dot1p rewrite policy

No default dot1p rewrite policy exists.

Effect of dot1p rewrite policy

Table 8: Effect of dot1p rewrite policy in relation to other configuration

Egress sub-interface type	Egress packet type	Behavior with dot1p-policy applied to egress sub-interface	Behavior without dot1p-policy applied to egress sub-interface
Untagged	Any	None, can be blocked.	—
Routed, single-tagged	Non-originated, forwarded IPv4/IPv6	Rewrites the PCP in the VLAN tag pushed at egress by the subinterface definition.	PCP = fc in the VLAN tag pushed at egress when no policy attached.
Routed, single-tagged	Self-originated IPv4/IPv6	None, ignored for self-originated traffic.	—
Routed, single-tagged	Non-originated, forwarded after VXLAN encapsulation, Ethernet untagged payload	PCP = fc in the VLAN tag pushed at egress by the subinterface definition.	PCP = fc in the VLAN tag pushed at egress when no policy attached.
Bridged, single-tagged	Non-originated, L2 forwarded	Rewrites the PCP in the VLAN tag pushed at egress by the subinterface definition.	PCP = 0 in the VLAN tag pushed at egress when no policy attached. Does not modify the PCP of VLAN tags in the payload. A VLAN tag is

Egress sub-interface type	Egress packet type	Behavior with dot1p-policy applied to egress sub-interface	Behavior without dot1p-policy applied to egress sub-interface
		• On 7220 IXR-D2/ D2L/D3/D3L, does not modify the PCP of VLAN tags in the payload. • On 7220 IXR-D4/ D5, also modifies the PCP of VLAN tags in the payload. A VLAN tag is carried as payload when the ingress subinterface is configured with vlan-tagging = true and vlan-id = any.	carried as payload when the ingress subinterface is configured with vlan-tagging = true and vlan-id = any.
Bridged, single-tagged	Self-originated IPv4/IPv6	None, ignored for self-originated traffic.	—
Bridged, single-tagged	Non-originated, L3 forwarded (IRB)	Rewrites the PCP in the VLAN tag pushed at egress by the subinterface definition.	PCP = fc in the VLAN tag pushed at egress when no policy attached.
Bridged, single-tagged	Non-originated, forwarded after VXLAN decapsulation, Ethernet untagged payload	Rewrites the PCP in the VLAN tag pushed at egress by the subinterface definition.	PCP = fc in the VLAN tag pushed at egress when no policy attached.

10.2.1 Configuring dot1p rewrite rules for output traffic

Prerequisites

To reference a forwarding-class in any QoS policy, the forwarding-class must first be explicitly mapped to an output queue. For information about mapping the named forwarding classes to named queues, see Named queues and forwarding classes.

Procedure

To configure dot1p rewrite rules, map internal forwarding classes (and optionally drop-probability) to the required dot1p values using the **qos rewrite-rules dot1p-policy** command.

The following example creates a dot1p rewrite-rule policy:

Example: Configure a dot1p rewrite rule

```
--{ candidate shared default }--[  ]--
# info with-context qos rewrite-rules
    qos {
        rewrite-rules {
```

```
            dot1p-policy rewrite-dot1p-example {
                map forwarding-class-0 {
                    dot1p 0
                }
                map forwarding-class-1 {
                    dot1p 3
                    drop-probability low {
                        dot1p 1
                    }
                    drop-probability high {
                        dot1p 2
                    }
                }
                map forwarding-class-3 {
                    dot1p 3
                }
                map forwarding-class-7 {
                    dot1p 7
                }
            }
        }
    }
```

10.2.2 Applying a dot1p rewrite rule to a subinterface

Procedure

To apply a dot1p rewrite rule to output traffic on a subinterface, specify the required policy using the **qos interfaces interface output rewrite-rules** command.

The following example applies a dot1p rewrite-rule policy to outbound traffic on a subinterface:

Example

```
--{ * candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-1/2
    qos {
        interfaces {
            interface ethernet-1/2 {
                interface-ref {
                    interface ethernet-1/2
                    subinterface 1
                }
                output {
                    rewrite-rules {
                        dot1p-policy rewrite-dot1p-example
                    }
                }
            }
        }
    }
```

11 Buffer allocation profile

QoS queue management features are configured using buffer allocation profiles and queue management profiles. These profiles are groups of configuration information that apply to a set of queues. On 7250 IXR systems, the controlled set of queues are VOQs; on 7220 IXR systems, the controlled set of queues are egress queues.

The maximum number of buffer allocation profiles and queue management profiles per system varies by platform. On 7250 IXR systems, the maximum is eight; on 7220 IXR systems, the maximum is 62.

A buffer allocation profile contains all configuration related to queue-depth parameters, including the following parameters:

- The MBS of each queue: this defines the length of each queue. When the queue builds to the MBS level, further packets are dropped. Be aware that discards may occur before the queue reaches MBS (for example, resulting from shared buffer exhaustion, or from the effects of WRED slopes defined for the queue).
- Queue utilization threshold: when a router receives a burst of traffic, and the incoming rate exceeds the available transmission rate, the router queues the excess traffic. If the burst lasts long enough, or it is followed by additional bursts, the queues may overflow, resulting in traffic loss. To respond to onsets of congestion, you can subscribe to telemetry information that generates an event when specific queues exceed a specified occupancy level.

If a VOQ does not have a buffer allocation profile binding, it inherits the settings of the default buffer allocation profile. The default buffer allocation profile has a platform-specific MBS default value and no defined queue utilization threshold. You cannot display the default buffer allocation profile, but its effect is visible by reading the state of individual queues that lack a buffer allocation profile binding.

11.1 Configuring buffer allocation profiles

Procedure

To create a buffer allocation profile, use the **qos buffer-management buffer-allocation-profile** command. You can then define the parameters for the profile as described in subsequent sections.

Example: Create a buffer allocation profile

```
--{ + candidate shared default }--[ ]--
# info with-context qos buffer-management buffer-allocation-profile
    qos {
        buffer-management {
            buffer-allocation-profile mbs-high-threshold-1 {
            }
        }
    }
```

Note: This example is only the starting point of a full configuration. Subsequent sections build on this example to create a full configuration.

11.2 Maximum burst size

In a buffer allocation profile, the **maximum-burst-size** parameter sets the maximum length of an egress queue or set of VOQs. The MBS is also known as the queue depth. You must set the **maximum-burst-size** parameter to a nonzero value to configure WRED slope and ECN slope parameters.

On the 7250 IXR systems, the **maximum-burst-size** parameter applies to a set of VOQs. If the parameter is not configured, the effective MBS of these VOQs is 256 MB.

On the 7220 IXR systems, the **maximum-burst-size** parameter applies to a set of egress queues. If the parameter is not configured or is set to 0, the effective MBS of these egress queues is calculated based on a fair allocation algorithm. You can assign a non-zero MBS value to multicast queues, but Nokia does not recommend this configuration (especially if multicast traffic is being shaped by configuring **peak-rate-percent**), because it can lead to a shortage of multicast-related buffering resources on 7220 IXR systems.

11.2.1 Configuring maximum burst size

Procedure

To configure the MBS within a buffer allocation profile, set a value using the **queue maximum-burst-size** command.

The following example specifies a **maximum-burst-size** for queue test-unicast-0:

Example: Configure MBS

```
--{ candidate shared default }--[ ]--
# info with-context qos buffer-management buffer-allocation-profile test-buffer-profile
 queues queue test-unicast-0
    qos {
        buffer-management {
            buffer-allocation-profile test-buffer-profile {
                queues {
                    queue test-unicast-0
                        maximum-burst-size 31457280
                    }
                }
            }
        }
    }
```

11.3 Committed burst size (7220 IXR-H4/H5)

In a buffer allocation profile, the **committed-burst-size** parameter sets the committed (guaranteed) length of an egress queue. The CBS must be less than or equal to MBS. However, unlike MBS, the CBS value cannot be oversubscribed because the CBS defines a guaranteed queue length. The difference of MBS minus the CBS is equal to the excess burst size (EBS).

A configuration change of the **committed-burst-size** value can fail in the following scenarios:

- The new CBS value, when added to all the existing CBS and PFC headroom reservations, oversubscribes the total shared memory available to front-panel I/O ports.

- The utilization of the queue is already greater than the new CBS value.

If the configuration fails, the new CBS value does not take effect. You can identify a failure by comparing the running configuration value of CBS (which displays the new value) with the state value (which displays the old value).

After CBS values are allocated system-wide, the remaining buffers can only be used for MBS (system-wide).

For LAG interfaces, CBS is assigned per physical port.

On IXR platforms, the CBS is supported only on the 7220 IXR-H4/H5.

CBS interaction with WRED and ECN thresholds

Regardless of the CBS setting, WRED and ECN slopes apply to the total queue length defined by MBS. For example if CBS = 4 MB and MBS = 8 MB, then an ECN slope with a **min-threshold-percent** value of 20% starts marking traffic when the queue length depth reaches 0.2 × 8 MB = 1.6 MB, which is still within the CBS range.

11.3.1 Configuring committed burst size (7220 IXR-H4/H5)

Prerequisites

Before you configure a CBS value, stop the flow of traffic on applicable interfaces. If you change the CBS value when traffic is flowing on an interface, the configuration may not take effect.

Procedure

To configure the CBS within a buffer allocation profile on 7220 IXR-H4/H5 platforms, set a value using the **queue committed-burst-size** command.

Example: Configure CBS

The following example specifies a **commited-burst-size** for queue `test-unicast-0`:

```
--{ candidate shared default }--[ ]--
# info with-context qos buffer-management buffer-allocation-profile test-buffer-profile
 queues queue test-unicast-0
    qos {
        buffer-management {
            buffer-allocation-profile test-buffer-profile {
                queues {
                    queue test-unicast-0
                        committed-burst-size 100000
                    }
                }
            }
        }
    }
```

11.4 Static queue utilization thresholds

When a router receives a burst of traffic, and the incoming rate exceeds the available transmission rate, the router queues the excess traffic. If the burst lasts long enough, or it is followed by additional bursts, the queues may overflow, resulting in traffic loss.

To respond to onsets of congestion, you can subscribe to telemetry information that generates an event when specific queues exceed a specified occupancy level.

To assign a utilization threshold to a queue, you must apply a non-default buffer allocation profile to the queue, and that buffer allocation profile must specify a nonzero **high-threshold-bytes** value. When the utilization of the queue crosses the specified **high-threshold-bytes** value, a hardware interrupt is raised. The Nokia XDP records the current system time and clears the interrupt. In a scaled setup, XDP may take 10 to 15 ms to process and clear each interrupt, meaning multiple threshold crossings within a very short period of time across one or more queues using the same buffer allocation profile may appear as only a single event in the telemetry stream. When the **high-threshold-bytes** value is 0, the functionality is disabled and no threshold events are generated for the queues covered by the buffer allocation profile.

SR Linux supports queue utilization thresholds on 7250 IXR, 7220 IXR-D2/D2L, D3/D3L, D4 and D5, and 7220 IXR-H2 and H3 systems; however, the behavior varies by system.

Note: You can only configure queue utilization thresholds for unicast queues; multicast queues do not support queue utilization thresholds.

Related topics

Queue depth sampling (7250 IXR)
Queue depth sampling (7220 IXR-H4/H5)

11.4.1 Configuring static queue utilization thresholds (7250 IXR)

Procedure

On a 7250 IXR system, bind a buffer allocation profile with a nonzero **high-threshold-bytes** value to an egress queue to assign that threshold value to all the VOQs that logically feed this egress queue.

You can configure each buffer allocation profile that the system supports with a different **high-threshold-bytes** value as needed.

Example: Configuring high-threshold-bytes

The following example configures the **high-threshold-bytes** value to 256255. For the configured value to be committed, a **maximum-burst-size** value must also be defined.

```
--{ candidate shared default }--[  ]--
 # info with-context qos buffer-management buffer-allocation-profile test-buffer-profile
 queues queue unicast-test-0
    qos {
        buffer-management {
            buffer-allocation-profile test-buffer-profile {
                queues {
                    queue unicast-test-0 {
                        maximum-burst-size 1203200768
                        high-threshold-bytes 256255
                    }
                }
            }
        }
    }
```

Each configured threshold value is rounded up to the nearest multiple of 256 bytes, up to a maximum capped value of MBS. You can observe the rounding (on a per VOQ-set basis) using the **info from**

state interface qos output queue-statistics queue <*queue-name*> **virtual-output-queue queue-depth** output. (A VOQ-set consists of the VOQ for core 0 and the VOQ for core 1.)

Example: Rounding high-threshold-bytes

In the following example, the **high-threshold-bytes** value was configured to 256255, but is rounded to the lower 256000 value (that is, a multiple of 256 bytes):

```
--{ candidate shared default }--[  ]--
# info from state with-context qos interfaces interface ethernet-1/35 output queues queue
 unicast-0 queue-statistics aggregate-statistics virtual-output-queue 1 queue-depth
    qos {
        interfaces {
            interface ethernet-1/35 {
                output {
                    queues {
                        queue unicast-0 {
                            queue-statistics {
                                aggregate-statistics {
                                    virtual-output-queue 1 {
                                        queue-depth {
                                            high-threshold-bytes 256000
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

The state tree maintains the time of the last threshold crossing in the **info from state qos interfaces interface** <*interface-name*>**output queues queue** <*queue-name*>**queue-depth last-high-threshold-time** leaf. This value represents the last time when either VOQ in the VOQ-set (core0/core1) exceeded the operational threshold. The value of this leaf is not cleared when you delete or modify the buffer allocation profile that is bound to the queue/VOQs or the **high-threshold-bytes** configuration in the applied buffer allocation profile.

11.4.2 Configuring static queue utilization thresholds (7220 IXR-D2/D2L/D3/D3L/D4/D5)

Procedure

On 7220 IXR-D2/D2L/D3/D3L/D4/D5 systems, bind a buffer allocation profile with a nonzero **high-threshold-bytes** value to an egress queue to assign that threshold value for that specific queue, as long as it is a unicast queue. The configuration of this leaf is ignored when this buffer allocation profile is attached to a multicast queue.

No more than seven different configured **high-threshold-bytes** values are allowed across all the buffer allocation profiles used. The management server rejects a commit that would leave more than seven different values after all adds, deletes, and modifies are processed.

Example: Configuring high-threshold-bytes

The following example configures the **high-threshold-bytes** value to 2048999:

```
--{ candidate shared default }--[ ]--
```

```
# info with-context qos buffer-management buffer-allocation-profile test-buffer-profile
 queues queue unicast-0
    qos {
        buffer-management {
            buffer-allocation-profile test-buffer-profile {
                queues {
                    queue unicast-test-0 {
                        maximum-burst-size 2049024
                        high-threshold-bytes 2048999
                    }
                }
            }
        }
    }
```

Each configured threshold value (that SR Linux accepts) is rounded up to the nearest multiple of 2048 bytes, up to a maximum capped value of MBS. For this reason, do not configure values that round to the same multiple of 2048 bytes. This configuration causes duplication among the **high-threshold-bytes** values, of which only seven are allowed. You can display the effect of this rounding using the **info from state interface qos output unicast-queue queue-depth** command.

Note: On 7220 IXR-D4 systems, the **high-threshold-bytes** values are rounded to multiples of 318 bytes. On 7220 IXR-D5 systems, the values are rounded to multiples of 254 bytes.

Example: Rounding high-threshold-bytes

In the following example, the **high-threshold-bytes** value was configured to 2048999, but is rounded to a lower 2048000 value (that is, a multiple of 2048 bytes):

```
--{ candidate shared default }--[  ]--
# info with-context from state qos interfaces interface ethernet-1/1 output queues queue
 unicast-0 queue-depth
    qos {
        interfaces {
            interface ethernet-1/1 {
                output {
                    queues {
                        queue unicast-0 {
                            queue-depth {
                                maximum-burst-size 2049024
                                high-threshold-bytes 2048000
                            }
                        }
                    }
                }
            }
        }
    }
```

The state tree maintains the time of the last threshold crossing in the **info from state qos interfaces interface** *<interface-name>***output queues queue** *<queue-name>***queue-depth last-high-threshold-time** leaf. This value represents the last time the queue exceeded the operational threshold. The value of this leaf is not cleared when you delete or modify the buffer allocation profile that is bound to the queue or the **high-threshold-bytes** configuration in the applied buffer allocation profile.

11.4.3 Configuring static queue utilization thresholds on (7220 IXR-H2/H3)

Procedure

On 7220 IXR-H2 and H3 systems, bind a buffer allocation profile with a nonzero **high-threshold-bytes** value to an egress queue to assign that threshold value to be used by each ITM that serves the queue. For a high-threshold event, the queue utilization threshold must be exceeded on either ITM.

No more than seven different configured **high-threshold-bytes** values are allowed across all the buffer allocation profiles used. The management server rejects a commit that would leave more than seven different values after all adds, deletes, and modifies are processed.

Example: Configuring high-threshold-bytes

The following example configures the **high-threshold-bytes** value to 254255:

```
--{ candidate shared default }--[  ]--
# info with-context qos buffer-management buffer-allocation-profile test-buffer-profile
 queues queue unicast-0
    qos {
        buffer-management {
            buffer-allocation-profile test-buffer-profile {
                queues {
                    queue unicast-test-0 {
                        maximum-burst-size 2049024
                        high-threshold-bytes 254255
                    }
                }
            }
        }
    }
```

Each configured threshold value (that the management server accepts) is rounded up to the nearest multiple of 254 bytes, up to a maximum capped value of MBS. For this reason, do not configure values that round to the same multiple of 254 bytes. This configuration causes duplication among the **high-threshold-bytes** values, of which only seven are allowed. You can display the effect of this rounding using the **info from state interface qos output unicast-queue queue-depth** command.

Example: Rounding high-threshold-bytes

In the following example, the **high-threshold-bytes** value was configured to 254255, but is rounded to a lower 254254 value (that is, a multiple of 254 bytes):

```
--{ candidate shared default }--[  ]--
# info with-context from state qos interfaces interface ethernet-1/1 output queues queue
 unicast-0 queue-depth
    qos {
        interfaces {
            interface ethernet-1/1 {
                output {
                    queues {
                        queue unicast-0 {
                            queue-depth {
                                maximum-burst-size 2049272
                                high-threshold-bytes 254254
                            }
                        }
                    }
                }
```

```
        }
    }
}
```

The state tree maintains the time of the last threshold crossing in the **info from state qos interfaces interface** *<interface-name>***output queues queue** *<queue-name>***queue-depth last-high-threshold-time** leaf. This value represents the last time when either ITM exceeded the operational threshold. The value of this leaf is not cleared when you modify or delete the buffer allocation profile that is bound to the queue or the **high-threshold-bytes** configuration in the applied buffer allocation profile.

11.5 Applying buffer allocation profiles to an interface

Procedure

To apply a buffer allocation profile to an interface, use the **qos interfaces interface output buffer-allocation-profile** command.

Example: Apply buffer allocation profile to an interface

```
--{ * candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-1/1
    qos {
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                }
                output {
                    buffer-allocation-profile test-buffer-profile
                }
            }
        }
    }
```

12 Queue depth sampling

As an alternative to static queue utilization thresholds, 7250 IXR and 7220 IXR-H4/H5 platforms support queue depth sampling. The feature implementation is platform-specific, allowing for the monitoring of VOQ congestion events on the 7250 IXR and for the collection of queue utilization statistics on the 7220 IXR-H4/H5.

12.1 Queue depth sampling (7250 IXR)

On 7250 IXR platforms, you can enable queue depth sampling on one or more queues. With queue depth sampling enabled on an egress queue, the system monitors all VOQs that can hold output traffic in the queue. You can then use state commands or telemetry to display or stream the collected queue depth sampling data for any congested VOQs.

On the 7250 IXR platforms, the interval between queue depth samples is short enough to allow for the detection of microbursts.

Threshold-triggered queue depth sampling

To determine the state of congestion for a VOQ, the system uses high and low threshold levels. When the VOQ depth exceeds the high threshold, the system records the start time of the congested state and begins collecting regular samples of the VOQ depth. When the VOQ depth subseqently drops below the low threshold and remains below that threshold for a configurable number of consecutive samples (default: 3), the system stops collecting samples from the VOQ and records the end time of the congested state as well as the high-watermark sample value.

Adaptive sampling frequency

The interval of time between consecutive samples of a particular VOQ is dependent on the number of VOQs that are currently in a congested state. When only a single VOQ is in a congested state, the time between consecutive samples should be less than 1 ms. Each additional congested VOQ adds to the system workload for collecting queue samples and therefore the time between consecutive samples for each VOQ increases.

Per-forwarding complex limit

Queue depth sampling limits the maximum number of VOQs for which congestion events are recorded to 32 per forwarding complex.

Configuration using buffer allocation profile

To enable queue depth sampling, you must configure the required per-queue parameters as part of a buffer allocation profile, which you then associate with one or more interfaces.

12.1.1 Configuring queue depth sampling (7250 IXR)

Procedure

To enable queue depth sampling on 7250 IXR platforms, use the **qos buffer-management buffer-allocation-profile queues queue queue-depth-sampling** command.

To apply a buffer allocation profile to an interface, use the **qos interfaces interface output buffer-allocation-profile** command.

Example: Configure queue depth sampling (7250 IXR)

```
--{ * candidate shared default }--[  ]--
7250-ixr# info with-context qos buffer-management buffer-allocation-profile test-buffer-
profile queues queue test-queue
    qos {
        buffer-management {
            buffer-allocation-profile test-buffer-profile {
                queues {
                    queue test-queue {
                        high-threshold-bytes 50000000
                        low-threshold-bytes 40000000
                        low-threshold-count 3
                        queue-depth-sampling true
                    }
                }
            }
        }
    }
```

The queue depth sampling parameters are defined as follows:

- **high-threshold-bytes** – Defines the high threshold level.
  The default value is 0. This parameter must be set to a non-zero value when queue depth sampling is enabled.

  Note: The **high-threshold-bytes** parameter applies to both the queue depth sampling feature and the static queue utilization feature. When **queue-depth-sampling** is **true**, the configured **high-threshold-bytes** value is applicable for queue depth sampling. But when **queue-depth-sampling** is **false** (the default value), the configured **high-threshold-bytes** value is applicable as a static queue utilization threshold.

- **low-threshold-bytes** – Defines the low threshold level.
  The default value is 0. This parameter must be set to a non-zero value when queue depth sampling is enabled.
- **low-threshold-count** – Defines the number of consecutive queue depth samples below the low threshold that are required to end the congested state.
  The default value is 3.
- **queue-depth-sampling** – Enables and disables queue depth sampling.
  The default value is **false**.

Example: Apply buffer allocation profile to an interface

```
--{ * candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-2/10
    qos {
```

```
        interfaces {
            interface ethernet-2/10 {
                interface-ref {
                    interface ethernet-2/10
                }
                output {
                    buffer-allocation-profile test-buffer-profile
                }
            }
        }
    }
```

Related topics

Configuring static queue utilization thresholds (7250 IXR)

12.1.2 Displaying queue depth sampling congestion event information (7250 IXR)

Procedure

When a congestion event is triggered for a particular interface and VOQ, polling is started on this VOQ (provided that the per-forwarding complex limit is not exceeded). To obtain state information related to the congestion event, use the following state command:

info from state qos interfaces interface output voq-interface queue queue-depth congestion event

Example: Display in-progress congestion event information

While polling of the congested VOQ is under way, the **queue-depth congestion event** state command displays the start time for the event and a status of **in-progress**.

```
--{ candidate shared default }--[  ]--
# info from state qos interfaces interface ethernet-2/10 output voq-interface voq-1-0-1
 queue uc0 queue-depth congestion-event 1

    status in-progress
    start-time "2025-06-19T13:33:18.120Z (a minute ago)"
    high-watermark 0
```

To specify a value for the **voq-interface** parameter, use the format *voq-x-y-z*, where *x* specifies the ingress slot, *y* specifies the ingress complex, and *z* specifies the ingress core (**0** or **1**).

Only the most recent congestion event for each VOQ is maintained in YANG state (index 0) and therefore the recording of the start time for a new congestion event overwrites all of the state for any previous congestion event. However, the system can stream telemetry off-platform so that historical information is not lost.

Example: Display completed congestion event information

When polling of the congested VOQ is complete, the **queue-depth congestion event** state command displays the end time for the event and a status of **done**. It also includes the high watermark queue depth level of the VOQ for this event.

```
--{ candidate shared default }--[  ]--
# info from state qos interfaces interface ethernet-2/10 output voq-interface voq-1-0-1
 queue uc0 queue-depth congestion-event 1

    status done
    start-time "2025-06-19T13:33:18.120Z (a minute ago)"
```

```
    end-time "2025-06-19T13:35:09.929Z (3 seconds ago)"
    high-watermark 100000032
```

Example: Display congestion event start time when the per-complex limit is reached

If a VOQ is congested when the per-complex limit has been reached, the system records the start time of the congestion, but polling cannot be started on the VOQ. In this case, the **queue-depth congestion-event** state command displays a status of **no-information-available**.

```
--{ candidate shared default }--[  ]--
# info from state qos interfaces interface ethernet-2/15 output voq-interface voq-3-0-0
 queue uc0 queue-depth congestion-event 1

    status no-information-available
    high-watermark 0
```

The start time value also overwrites the value from any previous congestion event.

12.1.3 Displaying forwarding complex associations for queue depth sampling (7250 IXR)

About this task

To enter the **queue-depth congestion-event** state command, the **voq-interface** parameter requires you to specify the forwarding complex that is associated with the VOQ.

Procedure

To determine the interfaces associated with a forwarding complex, use the **info from state platform linecard forwarding-complex interfaces** command.

Example: Display interfaces associated with a forwarding complex

```
--{ candidate shared default }--[  ]--
# info with-context from state platform linecard 1 forwarding-complex 0 interfaces
platform {
        linecard 1 {
            forwarding-complex 0 {
                interfaces [
                    ethernet-1/1
                    ethernet-1/2
                    ethernet-1/3
                    ethernet-1/4
...
                    ethernet-1/16
                    ethernet-1/17
                    ethernet-1/18
                ]
            }
        }
    }
```

12.2 Queue depth sampling (7220 IXR-H4/H5)

While static queue utilization thresholds allow the system to generate events in response to congestion, they do not provide a granular means to determine whether queue length parameters are configured correctly. If CBS and MBS values are too small, then too many packets may be dropped from the queue, even though the shared buffer space can accommodate them. And if CBS and MBS values are too large, then the traffic flowing through the queue may consume a disproportionate amount of the shared buffer space. To address these concerns, the 7220 IXR-H4/H5 platform supports queue depth sampling, which provides a detailed set of queue utilization statistics with no requirement for setting static queue thresholds in advance.

When queue depth sampling is enabled, the system reads all unicast output queue depths at every defined polling interval. The polling interval is a configurable range between 30 and 1000 ms (defaut: 1000 ms). The system then uses these samples to calculate and publish the aggregate queue depth statistics to state every 1, 5, and 15 minutes. As a reference, the following table shows the number of datapoints created per collection period based on a polling interval of 30 ms.

Table 9: Queue depth sampling datapoints

Collection period	Polling interval	Datapoints created
1 minute	30 ms	2000
5 minute	30 ms	10 000
15 minutes	30 ms	30 000

The aggregate statistics also include the number of poll results that were expected but never received during each 1 minute polling window.

The system maintains the collected queue depth sampling data under the **info from state qos interfaces interface output queues queue queue-depth** context using the following YANG state leafs:

- **average-1**, **average-5**, and **average-15**: the mean (average) values of all the measurements received in the last 1, 5, or 15 minute period
- **high-watermark-1**, **high-watermark-5**, and **high-watermark-15**: the maximum values of all the measurements received in the last 1, 5, or 15 minute period
- **missed-polling-intervals**: the running count of the number of samples that were expected, but did not arrive within the 1-minute window (because of high system utilization) and were therefore not included in any calculations

12.2.1 Configuring queue depth sampling (7220 IXR-H4/H5)

Procedure

To enable queue depth sampling, use the **qos queues queue-depth-sampling** command. To set the polling interval, use the **polling-interval** parameter.

Example: Enable queue depth sampling

```
--{ candidate shared default }--[  ]--
ixrh4# info with-context qos queues queue-depth-sampling
    qos {
        queues {
            queue-depth-sampling {
                admin-state enable
                polling-interval 60
            }
        }
```

```
    }
```

Example: Displaying queue depth sampling statistics

To display the queue depth sampling statistics, use the **info from state qos interfaces interface output queues queue queue-depth** command

```
--{ candidate shared default }--[  ]--
ixrh4# info from state with-context qos interfaces interface ethernet-1/54 output queues
 queue unicast-0 queue-depth
    qos {
        interfaces {
            interface ethernet-1/54 {
                output {
                    queues {
                        queue unicast-0 {
                            queue-depth {
                                maximum-burst-size 1024128
                                committed-burst-size 1024128
                                average-1 2044954
                                average-5 2045970
                                average-15 1507354
                                high-watermark-1 2048764
                                high-watermark-5 2048764
                                high-watermark-15 5040630
                                missed-polling-intervals 0
                            }
                        }
                    }
                }
            }
        }
    }
```

13 Queue management profile

Queue management profiles, like buffer allocation profiles, are groups of configuration information that apply to a set of VOQs on 7250 IXR systems and to a set of egress queues on 7220 IXR systems.

The maximum number of queue management profiles and buffer allocation profiles per system varies by platform. On 7250 IXR systems, the maximum is eight; on 7220 IXR systems, the maximum is 62.

The following parameters are configurable inside a queue management profile:

- WRED slopes that define probability curves for discarding packets as a function of weighted average queue depth. WRED slopes are not supported for multicast queues.
- ECN slopes that define probability curves for marking ECN-capable packets as having experienced congestion, instead of discarding them. ECN slopes are not supported for multicast queues.

If a VOQ does not have a queue management profile binding, it inherits the settings of the default queue management profile. The default queue management profile has a platform-specific MBS default value, no defined queue utilization thresholds, no WRED slopes, and no ECN slopes. You cannot display the default queue management profile, but its effect is visible by reading the state of individual queues that lack a queue management profile binding.

13.1 WRED slope

In a queue management profile, you can configure WRED policies to handle congestion when queue space is depleted. Without WRED, when a queue reaches its maximum fill size, the queue discards any packets arriving at the queue (known as tail drop).

WRED policies help to prevent congestion by starting random discards when the queue reaches a configurable threshold value. This behavior avoids the impact of discarding all the new incoming packets. By starting random discards at this threshold, an end system can adjust its sending rate to the available bandwidth.

The WRED curve algorithm is based on configurable thresholds (**min-threshold** [or **min-threshold-percent**] and **max-threshold** [or **max-threshold-percent**]) and a discard probability factor (**max-drop-probability-percent**).

On the 7220 IXR, you can configure a WRED slope to apply only to TCP or to non-TCP traffic. This configuration can be useful because TCP has built-in mechanisms to adjust its sending rate in response to packet drops. TCP-based senders lower the packet transmission rate when some of the packets fail to reach the far end.

13.2 ECN slope

Some IP applications support the ECN mechanism. With ECN, IP packets originated by such applications are not discarded when they enter a congested queue; instead, they are marked using the two ECN bits in the traffic class field of the IPv4 or IPv6 packet header. The receiver of IP packets marked as having experienced congestion can signal to the sender (through Layer 4 or higher protocols) to reduce its sending rate. The advantage of this feedback mechanism is that the sending rate can drop more gradually than the normal response of a TCP sender to packet discards. A more gradual back-off can result in higher effective throughput in the network.

An ECN slope is similar to a WRED slope and uses the same configurable thresholds (**min-threshold** [or **min-threshold-percent**] and **max-threshold** [or **max-threshold-percent**]) and the marking probability factor (**max-drop-probability-percent**).

To use an ECN slope, you must configure **enable-ecn true**.

13.3 Configuring queue management profiles

Procedure

To create a queue management profile, use the **qos buffer-management queue-management-profile** command. You can then define the parameters for the queue management profile as described in the following sections.

The following example creates a queue managment profile that you can use for any of the following:

- a set of VOQs on a 7250 IXR
- an egress queue on a 7220 IXR

Example: Create a queue management profile

```
--{ candidate shared default }--[  ]--
# info with-context qos buffer-management queue-management-profile wred-ecn-1
    qos {
         buffer-management {
            queue-management-profile wred-ecn-1 {
            }
        }
    }
```

Note: This example is only the starting point of a configuration. The following section builds on this example to create a full configuration.

13.4 Configuring WRED and ECN slopes

About this task

WRED slope and ECN slope are not configured separately. Instead SR Linux populates the ECN slope settings when **ecn-enable true** is set under the WRED slope configuration.

WRED and ECN slopes are not supported for multicast queues.

Procedure

To configure a WRED slope, within a queue management profile use the **weight-factor** command to define the weight to use in the calculation of the average weighted queue depth, and use the **wred wred-slope** command to configure the following:

- the type of traffic that the WRED slope applies to: **tcp**, **non-tcp**, or **all** (on the 7250 IXR, traffic type must be set to **all**, indicating both TCP and non-TCP traffic)

- the **drop-probability** that the WRED slope applies to (on both 7250 IXR and 7220 IXR, to enable ECN, the **drop-probability** must be set to **all**)
- the **min-threshold** (or **min-threshold-percent**), **max-threshold** (or **max-threshold-percent**), and **max-drop-probability-percent**
- the **enable-ecn** parameter, which controls whether ECN is enabled or not
- the **slope-enabled** parameter, which controls whether or not WRED peforms random discards

The following example specifies a WRED slope for low drop probability traffic flowing through a set of VOQs on a 7250 IXR. This WRED slope applies to both TCP and non-TCP traffic. ECN is also enabled.

Example: Configure WRED slope

```
--{ * candidate shared default }--[  ]--
# info with-context qos buffer-management queue-management-profile wred-ecn-1
    qos {
        buffer-management {
            queue-management-profile wred-ecn-1 {
                weight-factor 5
                wred {
                    wred-slope all drop-probability all enable-ecn true {
                        min-threshold-percent 10
                        max-threshold-percent 25
                        slope-enabled true
                        max-drop-probability-percent 50
                    }
                }
            }
        }
    }
```

13.5 Configuring an ECN slope

About this task

On 7250 IXR systems, the ECN configuration requires you to specify an ECN DSCP policy; this is the DSCP rewrite policy that is used when an ECN field rewrite must be performed. In addition, you can only have one ECN slope per queue that applies to all drop-probability levels.

On the 7220 IXR-D2/D2L, D3/D3L, D4, and D5 or the 7220 IXR-H2, H3, H4, and H5, one ECN slope is configurable per drop-probability level of traffic flowing through an egress queue.

Procedure

To configure an ECN slope:

- For 7250 IXR only, in the **explicit-congestion-notification** context, specify the DSCP policy to use when ECN rewrite is enabled.
- For both 7250 IXR and 7220 IXR, in the **queue-management-profile** context:
  - Enable ECN using the **wred wred-slope** *<traffic-type>* **drop-probability** *<probability>* **enable-ecn true** command.

    The *<traffic-type>* must be set to **all**, and on 7250 IXR the **drop-probability** must also be set to **all**.
  - Within that entry, set the desired ECN values: **min-threshold** (or **min-threshold-percent**), **max-threshold** (or **max-threshold-percent**), and **max-drop-probability-percent**.

Example: Configure an ECN slope (7250 IXR)

The following example specifies an ECN slope applicable to a 7250 IXR system:

```
--{ candidate shared default }--[  ]--
# info with-context qos
    qos {
        explicit-congestion-notification {
            ecn-dscp-policy normalize
        }
        buffer-management {
            queue-management-profile wred-ecn-1 {
                weight-factor 5
                wred {
                    wred-slope all drop-probability all enable-ecn true {
                        min-threshold-percent 50
                        max-threshold-percent 50
                        slope-enabled true
                        max-drop-probability-percent 100
                    }
                }
            }
        }
    }
```

Example: Configure an ECN slope (7220 IXR)

The following example specifies an ECN slope applicable to a 7220 IXR-D2/D2L, D3/D3L, D4, and D5 or 7220 IXR-H2, H3, H4, and H5 system:

```
--{ candidate shared default }--[  ]--
# info with-context qos buffer management
    qos {
        buffer-management {
            queue-management-profile ecn-2 {
                wred {
                    wred-slope all drop-probability high enable-ecn true {
                        min-threshold-percent 0
                        max-threshold-percent 80
                        max-drop-probability-percent 90
                    }
                }
            }
        }
    }
```

13.5.1 Displaying ECN statistics per forwarding class (7250 IXR-6e/10e/X1b/X3b)

About this task

On transmitting interfaces that are assigned an ECN slope, 7250 IXR-6e/10e/X1b/X3b systems provide per-forwarding-class ECN statistics.

Procedure

To display the ECN statistics, use the following commands under the **info from state qos interfaces interface output queues queue queue-statistics aggregate-statistics** context:

- **ecn-marked-packets** – displays the number of packets for which the ECN codepoint changed from ECN-capable transport (ECT) to congestion experienced (CE)
- **ecn-marked-octets** – displays the number of octets in packets for which the ECN codepoint changed from ECT to CE

To reset the queue statistics counters to zero for a queue, use the following command.

tools qos interfaces interface output queues queue queue-statistics clear

Example: Display ECN statistics per forwarding class

The following example displays ECN statistics for queue oc-unicast-0 on interface ethernet-1/33.

```
--{ candidate shared default }--[  ]--
# info from state with-context qos interfaces interface ethernet-1/33 output queues queue
 oc-unicast-0 queue-statistics aggregate-statistics ecn-marked-packets
    qos {
        interfaces {
            interface ethernet-1/33 {
                output {
                    queues {
                        queue oc-unicast-0 {
                            queue-statistics {
                                aggregate-statistics {
                                    ecn-marked-packets 11955776
                                }
                            }
                        }
                    }
                }
            }
        }
    }
-{ candidate shared default }--[  ]--
# info from state with-context qos interfaces interface ethernet-1/33 output queues queue
 oc-unicast-0 queue-statistics aggregate-statistics ecn-marked-octets
    qos {
        interfaces {
            interface ethernet-1/33 {
                output {
                    queues {
                        queue oc-unicast-0 {
                            queue-statistics {
                                aggregate-statistics {
                                    ecn-marked-octets 11764483584
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Reset queue statistics

The following example resets the queue statistics for queue unicast-7 on interface ethernet-1/33.

```
--{ candidate shared default }--[  ]--
# tools qos interfaces interface ethernet-1/33 output queues queue unicast-7 queue-
statistics clear
```

13.6 Applying queue management profiles to an interface

Procedure

To apply a queue management profile to an interface, use the **output queues queue queue-management-profile** command.

To specify a queue management profile for an interface, the **interface-ref** parameter must not have a subinterface configured.

Example: Apply queue management profile to an interface

```
--{ * candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-1/1
    qos {
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                }
                output {
                    buffer-allocation-profile test-buffer-profile
                    queues {
                        queue test-unicast-queue {
                            queue-management-profile test-queue-mgmt-profile
                        }
                    }
                }
            }
        }
    }
```

14 Output queue scheduler policies

SR Linux supports the configuration of queue scheduler policies, providing the flexibility to define:

- which queues are served strict priority
- which queues are served WRR (and their weights)

Each output queue is mapped to either scheduler 0 or scheduler 1, as defined by the scheduler policies, with the following restrictions:

- Scheduler 0 — must have a **priority** setting of **strict**, providing strict priority scheduling behavior
- Scheduler 1 — must have no **priority** configuration defined, which provides WRR scheduling behavior

You can apply the defined policies to specified interfaces, as required.

Queue scheduler policies are supported on 7220 IXR and 7250 IXR platforms.

14.1 Configuring queue scheduler policies

About this task

When you configure scheduler polices, be aware of the following considerations:

- By default, all queues in the policy are attached to scheduler 0, which is served strict priority with a PIR of 100.
- Queues that are mapped to scheduler 0 are strict priority queues (which ignore any configured weight value) and queues that are mapped to scheduler 1 are WRR queues. The schedulers are processed from lowest sequence to highest.
- When strict priority is enabled (**priority strict**), any configured weight is ignored.
- When strict priority is not enabled, the associated queue or scheduler node is configured as WRR.

Procedure

To configure queue scheduler policies, use the **qos scheduler-policies scheduler-policy** command.

Example: Configure a strict priority policy

```
# info with-context qos scheduler-policies scheduler-policy SP
    qos {
        scheduler-policies {
            scheduler-policy SP {
                scheduler 0 {
                    priority strict
                    input q0 {
                        queue-name unicast-0
                        peak-rate-percent 100
                        weight 1
                    }
                }
            }
        }
```

```
    }
```

Example: Configure a WRR policy

```
# info with-context qos scheduler-policies scheduler-policy WRR
    qos {
        scheduler-policies {
            scheduler-policy WRR {
                scheduler 1 {
                    input q1 {
                        queue-name unicast-1
                        peak-rate-percent 100
                        weight 1
                    }
                }
            }
        }
    }
```

Example: Displaying the hardware programmed PIR values

You can also use the **info from state qos scheduler-policies** command to display the hardware programmed PIR values.

```
# info from state with-context qos scheduler-policies
    qos {
        scheduler-policies {
            scheduler-policy SP {
                scheduler 0 {
                    priority strict
                    input q0 {
                        input-type queue
                        queue-name unicast-0
                        peak-rate-percent 100
                        weight 1
                    }
                }
            }
            scheduler-policy WRR {
                scheduler 1 {
                    input q1 {
                        input-type queue
                        queue-name unicast-1
                        peak-rate-percent 100
                        weight 1
                    }
                }
            }
```

14.2 Applying a queue scheduler policy to an interface

Procedure

To apply the queue scheduler policy to an interface, use the **qos interfaces interface output scheduler** command.

Example: Apply a queue scheduler policy to an interface

```
# info with-context qos interfaces interface ethernet-1/2 output scheduler
    qos {
        interfaces {
            interface ethernet-1/2 {
                output {
                    scheduler {
                        scheduler-policy SP
                    }
                }
            }
        }
    }
```

15 Ingress subinterface traffic policing

Some SR Linux-compatible hardware platforms (7220 IXR-D2/D3/D2L/D3L/D4/D5) support the ability to direct selected traffic flows to hardware policers. Traffic directed to a policer is metered to determine compliance with a traffic profile. At the output of the policer, every packet is marked with a color (green, yellow, or red) that represents whether it conforms, exceeds, or violates the traffic profile.

With a two-rate-three-color marker (RFC 2698), the traffic profile is defined using two traffic rates and their associated burst sizes:

- comitted information rate (CIR) and committed burst size (CBS)
- peak information rate (PIR) and maximum burst size (MBS)

15.1 Token buckets

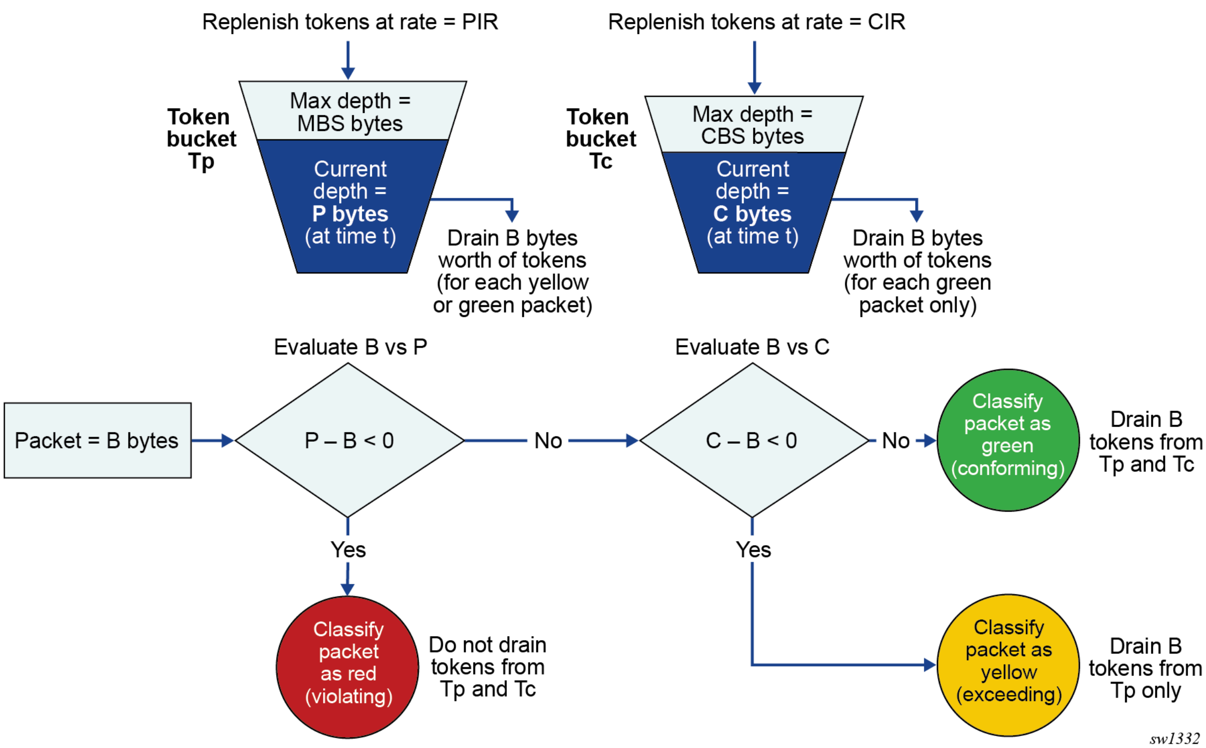
To determine compliance with the traffic profile, each policer uses two token buckets:

- **CIR bucket (Tc)**
  Tc has a fill rate equal to the CIR and a maximum depth of CBS bytes (with current depth at time *t* of *C* bytes).
- **PIR bucket (Tp)**
  Tp has a fill rate equal to the PIR and a maximum depth of MBS bytes (with current depth at time *t* of *P* bytes).

Initially (at time 0) the token buckets Tp and Tc are full, so that *P* = MBS and *C* = CBS. From then onwards, each bucket is continuously refilled at the rate of PIR and CIR.

The following diagram shows the token bucket process for each packet that arrives at the policer.

Figure 1: Token buckets (trTCM)



Each policer instance operates in a nonconfigurable color-aware mode. When a packet of size *B* bytes arrives at time *t*, the policer processes the packet as follows:

- If the packet is precolored as red or if *P* – *B* < 0, the packet is red (violating) and no tokens are drained from Tp or Tc. The policer either drops the packet or updates its drop probability as defined in the policer template (low, medium, or high).
- If the packet is precolored as yellow or if *C* – *B* < 0, the packet is yellow (exceeding), and *B* bytes are drained from Tp. The policer assigns the packet an updated drop probability as defined in the policer template (low, medium, or high).
- Otherwise, the packet is green (conforming), and *B* bytes are drained from Tp and Tc. The policer forwards the packet with no modifications, and drop probability remains unchanged (low).

Note: A drop probability of medium or high increases the chance that the packet is discarded (or ECN-marked) when it enters the egress queue, if that egress queue has a WRED/ECN slope.

Pre-coloring based on drop probability

All packets arrive at the input of the policer with an assigned drop probability, based on the DSCP classifier policy. Each policer treats these packets as pre-colored as described in the preceding section. The following table describes the colors associated with each packet based on the drop probability at input.

Table 10: Drop probability to color mapping

Drop probability at input	Color associated at input
Low	Green (conforming)

Drop probability at input	Color associated at input
Medium	Yellow (exceeding)
High	Red (violating)

15.2 Policer template

To assign policers to subinterfaces, you must first configure policer templates. A policer template specifies a group of 1 to 32 policers, each with a specified sequence ID. Policers with lower sequence IDs are evaluated before policers with higher sequence IDs. You can configure each policer to match a forwarding class and optionally, forwarding type.

You can apply policer templates to the following subinterface types:

- bridged subinterfaces of Ethernet ports or LAGs on a mac-vrf network instance
- routed subinterfaces of Ethernet ports or LAGs on either the default network instance or an ip-vrf network instance

Note:

- On routed subinterfaces, all traffic is considered to match the unicast forwarding type, even if it is received with a broadcast destination IP.
- Classification to forwarding class and drop probability occurs before policing in the ingress pipeline.
- There is no ingress policing of traffic of a particular forwarding class and forwarding type if that traffic has no match in the associated policer template.

IRB subinterface

Attachment of a policer template to an IRB subinterface is not currently supported.

Multiple subinterfaces referring to same policer template

Subinterfaces cannot share the same policer. If two or more different subinterfaces (routed or bridged) of the same port, same line card, or same chassis refer to the same policer template, each subinterface applies a separate instance of the template, consuming an equal number of TCAM entries.

Policing on LAG subinterfaces

Policers applied to subinterfaces are instantiated on each pipeline. The 7220 IXR-D2/D3/D2L/D3L platforms each have two pipelines, with half the ports mapping to pipeline 0 and the other half of the ports mapping to pipeline 1. The 7220 IXR-D4 has four pipelines and 7220 IXR-D5 has eight pipelines, with each pipeline supporting a variable number of ports.

These pipelines impact policing of ingress traffic on LAG subinterfaces as follows:

- The actual PIR for LAG subinterface traffic is: N × the quantized PIR, where N is the number of pipelines spanned by the LAG.
- The actual CIR for LAG subinterface traffic is: N × the quantized CIR, where N is the number of pipelines spanned by the LAG.

15.3 Policer statistics

For each policer template, you can choose between two statistics modes:

- **violating-focus**

  Collects the number of:

  - accepted (not dropped) packets and octets (counting all drop probabilities at policer output).
  - violating packets and octets.
- **forwarding-focus**

  Collects the number of:

  - committed packets and octets (conforming traffic only).
  - accepted (not dropped) exceeding packets and octets.

15.4 TCAM resources and scale

The following considerations apply to traffic policers and Ternary Content Addressable Memory (TCAM) resources:

- If a policer template is configured on a subinterface, and any linecard supporting that subinterface cannot program all the TCAM rules of all the policers defined in that policer template, then policing is not activated on the subinterface. In this case, the **info from state** output for the subinterface shows no policer template bound to the subinterface.
- When a policer template bound to a subinterface is in a failed state due to TCAM resource exhaustion, all further configuration of the policer template fails except for deletion of policers from the policer template and unbinding the policer template from a subinterface.

15.5 Configuring a subinterface traffic policer template

Prerequisites

To reference a forwarding-class in any QoS policy, the forwarding-class must first be explicitly mapped to an output queue. For information about mapping the named forwarding classes to named queues, see Named queues and forwarding classes.

Procedure

To configure a policer template, use the **qos policer-templates** command.

Note: For PIR and CIR, the configured value and the operational value can differ as a result of rate quantization in the hardware. The actual operational PIR and CIR rates used in the hardware are available in the state representation of each policer instance created from the template, using the following command: **info from state qos interfaces interface** <*name*> **input policer-templates policer** <*sequence-id*>.

Example: Configure subinterface traffic policer

The following example configures a policer template containing one policer with sequence ID 100 that has a defined PIR, CIR, MBS, and CBS, and that matches unicast fc1 traffic. Yellow packets are marked with a **drop-probability** of **medium**, and red packets are dropped. The **statistics-mode** is set to **violating-focus**.

```
--{ candidate shared default }--[  ]--
# info with-context qos policer-templates
    qos {
        policer-templates {
            policer-template test-policer-1 {
                statistics-mode violating-focus
                policer 100 {
                    peak-rate-kbps 15000
                    committed-rate-kbps 10000
                    maximum-burst-size 100000
                    committed-burst-size 20000
                    forwarding-class forwarding-class-1 {
                        forwarding-type [
                            unicast
                        ]
                    }
                    exceed-action {
                        drop-probability medium
                    }
                    violate-action {
                        drop
                    }
                }
            }
        }
    }
```

*Table 11: Parameters for **qos policer-templates***

Parameter	Definition
policer-template <*name*>	Assigns a name to the policer template.
statistics-mode {**violating-focus** | **forwarding-focus**}	(Optional) Defines the statistics mode (default: **violating-focus**).
policer <*sequence-id*>	Assigns the sequence ID for the policer.
peak-rate-kbps <*0 to 4294967295*>	Sets PIR in kb/s. The minimum supported PIR is 8 kb/s.
committed-rate-kbps: <*0 to 4294967295*>	Sets CIR in kb/s. The minimum supported CIR is 8 kb/s.
maximum-burst-size <*512 to 4294967295*>	Sets MBS in bytes (4294967295 bytes = 268 MB).
committed-burst-size <*512 to 4294967295*>	Sets CBS in bytes (4294967295 bytes = 268 MB).
forwarding-class <*fc*> [**forwarding-type** {**broadcast**|**multicast**|**unicast**|**unknown-unicast**}]	(Optional) Matches the policer to the specified forwarding class and optionally, forwarding type. If no forwarding class is specified, all traffic is matched.

Parameter	Definition
	If traffic of a specific forwarding class has no mapping in your policer-template, it is not policed at ingress. To match any traffic that is not explicitly mapped, include a policer with no forwarding class specified (for example, as the lowest-priority policer in the template). This policy ensures that the policer template matches all traffic at ingress.
exceed-action drop-probability {**high** | **low** | **medium**}	(Optional) Applies a drop-probability to yellow packets (default: **drop-probability medium**).
violate-action {**drop** | **drop-probability** {**high** | **low** | **medium**}}	(Optional) Applies an action (drop packets or assign a drop-probability) to red packets (default: **drop-probability high**).

15.6 Assigning a traffic policer template to a subinterface

Procedure

To apply a policer template to a subinterface, specify the required template using the **qos interfaces interface** <*name*> **input policer-templates policer-template** command.

The following example applies policer template 100 to subinterface 1/2.1

Example: Assign traffic policer template to a subinterface

```
--{ * candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-1/2
    qos {
        interfaces {
            interface ethernet-1/2 {
                interface-ref {
                    interface ethernet-1/2
                    subinterface 1
                }
                input {
                    policer-templates {
                        policer-template 100
                    }
                }
            }
        }
    }
```

15.7 Displaying subinterface traffic policer statistics

Procedure

Use the **info from state** command to display the subinterface traffic policer statistics.

The following example displays traffic policer statistics.

Example: Display traffic policer subinterface statistics

```
--{ candidate shared default }--[  ]--
# info from state interface ethernet-1/2 subinterface 1 qos input policer-templates
 policer 1 statistics
```

You can use the following options to narrow the scope of the statistics output.

- In violating-focus mode only:
  - **accepted-octets**
  - **accepted-packets**
  - **violating-octets**
  - **violating-packets**
- In forwarding-focus mode only:
  - **committed-octets**
  - **committed-packets**
  - **exceeding-octets**
  - **exceeding-packets**

15.8 Clearing subinterface traffic policer statistics

Procedure

To reset the policer statistics counters for an interface, use the **tools qos interfaces interface** <*name*> **input policer-templates clear** command.

Example: Reset all policer statistics counters on a subinterface

The following example resets all policer statistics counters on an interface:

```
--{ running }--[  ]--
# tools qos interfaces interface ethernet-1/3 input policer-templates clear
```

Example: Reset statistics counters for specific policer

The following example resets statistics counters for policer 2 on ethernet-1/3:

```
--{ running }--[  ]--
# tools qos interfaces interface ethernet-1/3 input policer-templates policer 2 clear
```

16 MPLS QoS overview

SR Linux supports QoS capabilities in MPLS networks using traffic classification and marking.

MPLS traffic classification and marking

SR Linux supports EXP-inferred LSPs as described in RFC 3270, which allow multiple classes of service to be transported by a single LSP. The EXP marking of each packet determines the correct per-hop behavior (PHB) to apply to each router.

On SR Linux, the mapping between an EXP value and a PHB is provided by an MPLS traffic-class classifier policy. A single router can have one or more of these policies so that some subinterfaces can have one policy applied and other subinterfaces can have another policy applied. Each traffic-class classifier policy consists of multiple mapping entries, each of which maps one unique EXP value to a (forwarding class, drop probability) tuple.

SR Linux also supports MPLS traffic-class rewrite policies. If MPLS-encapsulated packets are transmitted out an egress subinterface with such a policy bound to it, the EXP field in all the pushed labels of these packets is based on the mapping rules of the policy. MPLS traffic-class rewrite rules associate a forwarding class or a (forwarding class, drop probability) tuple with an EXP rewrite value.

SR Linux does not support the short-pipe model of RFC 3270.

Note: All of the MPLS QoS behavior documented in this chapter assumes that the MPLS-enabled subinterfaces have no other QoS configuration that takes precedence over MPLS QoS, such as dot1p classifiers.

16.1 Ingress LER

When an SR Linux router that is acting as an ingress label edge router (LER) matches an IP packet to a label distribution protocol (LDP) tunnel or a static MPLS forwarding entry, the following apply.

- The ingress LER determines the forwarding class and drop probability of the packet from the IP DSCP of the received unlabeled packet, based on the DSCP classifier policy applied to the ingress subinterface (or the default DSCP classifier policy if there is no explicit association). If an MPLS traffic-class (TC) policy is applied to the ingress subinterface, it has no effect.
- If a DSCP rewrite policy is applied to the egress subinterface, the IP header DSCP value is rewritten before the egress MPLS encapsulation is applied.
- If no MPLS TC rewrite policy is associated with the egress subinterface, EXP = 0 is written into all pushed labels.
- If an MPLS TC rewrite policy is associated with the egress subinterface, and it matches the forwarding class (and possibly also the drop probability) of the packet, the EXP provided by the mapping rule is written into the EXP field of all pushed labels.
- If ECN is enabled globally, and the packet hits an ECN slope in a congested queue such that the ECN marking should be `11`, the ECN field of the packet is modified accordingly, and the DSCP field is also remarked according to the ECN DSCP policy.

16.2 Transit LSR

When an SR Linux router that is acting as a transit label switching router (LSR) matches an MPLS packet to a swap ILM entry, the following apply.

- The transit LSR determines the forwarding class and drop probability of the packet from the EXP in the topmost label stack entry of the received labeled packet (before popping), based on the MPLS TC classifier policy applied to the ingress subinterface (or the default MPLS TC classifier policy, if there is no explicit association).
- If a DSCP classifier policy is applied to the ingress subinterface, it has no effect on the packet classification.
- If a DSCP rewrite policy is applied to the egress subinterface, it has no effect on the transmitted MPLS packet.
- If no MPLS TC rewrite policy is associated with the egress subinterface, the classified FC of the packet is written as a value 0 to 7 into the EXP field of all pushed labels. This behavior does not guarantee that the EXP of the popped labels matches the EXP of the pushed labels (that is, if a non-default MPLS TC classifier policy is applied to the ingress subinterface).
- If an MPLS TC rewrite policy is associated with the egress subinterface, and it matches the forwarding class (and possibly also the drop probability) of the packet, the EXP provided by the mapping rule is written into the EXP field of all pushed labels.
- If ECN is enabled globally, it has no effect on the MPLS packet. The MPLS packet is considered non-ECT capable, even if the buried IP ECN bits indicate otherwise. The IP ECN field is not modified.

16.3 PHP LSR

When an SR Linux router that is acting as a penultimate hop popping (PHP) LSR matches an MPLS packet to a pop and swap-to-implicit-null ILM entry, the following apply.

- The PHP LSR determines the forwarding class and drop probability of the packet from the EXP in the topmost label stack entry of the received labeled packet (before popping), based on the MPLS TC classifier policy applied to the ingress subinterface (or the default MPLS TC classifier policy, if there is no explicit association). If a DSCP classifier policy is applied to the ingress subinterface, it has no effect on the classification of the packet.
- If an MPLS TC rewrite policy is applied to the egress subinterface, it has no effect on the transmitted IP packet.
- If no DSCP rewrite policy is associated with the egress subinterface, the DSCP field of the IP payload packet is transmitted unchanged. There is no attempt to copy the EXP field into the IP DSCP of the IP payload packet.
- If a DSCP rewrite policy is associated with the egress subinterface, and it matches the forwarding class (and possibly also the drop probability) of the packet, the DSCP provided by the mapping rule is written (as an override) into the DSCP field in the transmitted IP packet. This behavior is consistent with the uniform model of RFC 3270.
- If ECN is enabled globally, it has no effect on the PHP packet. The PHP packet is considered non-ECT capable even if the IP ECN bits indicate otherwise. The IP ECN field is not modified.

16.4 Egress LER

When an SR Linux router that is acting as an egress LER matches an MPLS packet to a pop ILM entry that leads to all labels being popped, the following apply.

- The egress LER determines the forwarding class and drop probability of the packet from the EXP in the topmost label stack entry of the received labeled packet (before popping), based on the mpls-tc classifier policy applied to the ingress subinterface (or the default mpls-tc classifier policy, if there is no explicit association).

  If a DSCP classifier policy is applied to the ingress subinterface, it has no effect on the classification of the packet.
- If an mpls-tc rewrite policy is applied to the egress subinterface, it has no effect on the transmitted IP packet.
- If no DSCP rewrite policy is associated with the egress subinterface, the DSCP field of the IP payload packet is transmitted unchanged. There is no attempt to copy the EXP field into the IP DSCP of the IP payload packet. This behavior is consistent with the pipe model of RFC 3270.
- If a DSCP rewrite policy is associated with the egress subinterface, and it matches the forwarding class (and possibly also the drop probability) of the packet, the DSCP provided by the mapping rule is copied into the IP DSCP of the transmitted IP packet, overwriting the previous value. This behavior is consistent with the uniform model of RFC 3270.
- If ECN is enabled globally, it has no effect on the terminating MPLS packet. The terminating packet is considered non-ECT capable even if the IP ECN bits indicate otherwise. The IP ECN field is not modified.

Note: The DSCP marking of terminating MPLS traffic cannot be decoupled from the DSCP marking of transit IP traffic through the same egress subinterface.

16.5 Default MPLS traffic-class classifier policy

The following table shows the default MPLS TC classifier policy.

Table 12: Default MPLS TC classifier policy

Traffic class (EXP)	Forwarding class	Drop probability
0	0	Low
1	1	Low
2	2	Low
3	3	Low
4	4	Low
5	5	Low
6	6	Low

Traffic class (EXP)	Forwarding class	Drop probability
7	7	Low

17 MPLS QoS configuration

MPLS QoS configuration on SR Linux involves configuring MPLS traffic-class policies for input traffic and configuring MPLS rewrite rules for output traffic. These policies are applied to individual subinterfaces as required.

17.1 Configuring MPLS traffic-class policy

Prerequisites

To reference a forwarding-class in any QoS policy, the forwarding-class must first be explicitly mapped to an output queue. For information about mapping the named forwarding classes to named queues, see Named queues and forwarding classes.

Procedure

To configure an MPLS traffic-class policy, map one or more **traffic-class** values to the desired **forwarding-class** and **drop-probability** values using the **qos classifiers mpls-traffic-class-policy** command.

The following example creates an MPLS traffic-class policy:

Example

```
--{ candidate shared default }--[  ]--
# info with-context qos classifiers mpls-traffic-class-policy mpls-policy-1
    qos {
        classifiers {
            mpls-traffic-class-policy mpls-policy-1 {
                traffic-class 7 {
                    forwarding-class fc7
                    drop-probability medium
                }
            }
        }
    }
```

17.2 Applying MPLS traffic-class policy to input traffic

Procedure

To apply an MPLS traffic-class policy to input traffic on a subinterface, specify the desired **mpls-traffic-class-policy** using the **qos interfaces interface input classifiers** command.

The following example applies an MPLS traffic-class policy to inbound traffic on a subinterface.

Example

```
--{ candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/1
    qos {
```

```
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
                input {
                    classifiers {
                        mpls-traffic-class-policy mpls-policy-1
                    }
                }
            }
        }
    }
```

17.3 Configuring MPLS rewrite rules

Prerequisites

To reference a forwarding-class in any QoS policy, the forwarding-class must first be explicitly mapped to an output queue. For information about mapping the named forwarding classes to named queues, see Named queues and forwarding classes.

Procedure

To configure an MPLS rewrite-rule policy, map one or more forwarding classes to the desired **traffic-class** using the **qos rewrite-rules mpls-traffic-class-policy** command.

The following example creates an MPLS rewrite-rule policy:

Example

```
--{ candidate shared default }--[  ]--
# info with-context qos rewrite-rules
    qos {
        rewrite-rules {
            mpls-traffic-class-policy mpls-rewrite-2 {
                map fc7 {
                    traffic-class 7
                }
            }
        }
    }
```

17.4 Applying MPLS rewrite rules to output traffic

Procedure

To apply an MPLS rewrite rule policy to output traffic on a subinterface, specify the desired **mpls-traffic-class** policy using the **qos interfaces interface output rewrite-rules** command.

The following example applies a rewrite-rule policy to outbound traffic on a subinterface.

Example

```
--{ candidate shared default }--[  ]--
```

```
# info with-context qos interfaces interface ethernet-1/1
    qos {
        interfaces {
            interface ethernet-1/1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                    }
                output {
                    rewrite-rules {
                        mpls-traffic-class-policy mpls-rewrite-2
                    }
                }
            }
        }
    }
```

18 Buffer utilization display

The following table describes the buffer utilization differences between the 7250 IXR, 7220 IXR-D2, D3, and D5 or 7220 IXR-H2 and H3.

Table 13: Buffer utilization

Hardware	Buffer memory
7250 IXR	• SRAM size = 32 MB • DRAM (HBM) size = 8 GB
7220 IXR-D2 and D3	• Total buffer size = 32 MB • Reserved buffer size = 4.65 MB
7220 IXR-D5	• Total buffer size = 132 MB • Reserved buffer size = 3.7 MB
7220 IXR-H2 and H3	• Total buffer size = 64 MB • Reserved buffer size = 6.7 MB

18.1 Displaying buffer utilization

Procedure

To display buffer utilization, use the **info from state** command.

The following examples show overall buffer usage. The output varies depending on the hardware deployed.

Example: Displaying buffer utilization (7250 IXR)

```
# info with-context from state platform linecard 1 forwarding-complex 0 buffer-memory
    platform {
        linecard 1 {
            forwarding-complex 0 {
                buffer-memory {
                    sram {
                        used 15808512 >> in bytes
                        free 17745920  >> in bytes
                    }
                    dram {
                        used 48   >>> it is in % of DRAM
                    }
                }
            }
        }
    }
```

Example: Displaying buffer utilization (7220 IXR-D2, D3, and D5 or 7220 IXR-H2 and H3)

```
# info with-context from state platform linecard 1 forwarding-complex 0 buffer-memory
    platform {
        linecard 1 {
            forwarding-complex 0 {
                buffer-memory {
                   used  2097152
                   free  27263246
                   reserved 4194034
                }
            }
        }
    }
```

19 VOQ statistics collection

The **qos interfaces interface voq-statistics** command allows you to explicitly configure VOQ statistics collection on an interface (Ethernet or LAG). When **voq-statistics** is set to **true** for an interface, qos_mgr attempts to allocate VOQ statistics collection resources from the global VOQ statistics resource pool to that interface. This feature allows you to prevent VOQ statistics resource exhaustion by allocating VOQ statistics resources to those interfaces that require them most.

Note: Exhaustion of VOQ statistics resources is typically only a concern for 7250 IXR platforms that support a higher number of slots, such as the 7250 IXR-18e.

If **voq-statistics** is not explicitly configured, the implicit default setting is **true**.

Failure to allocate resources

If an interface has VOQ statistics enabled, but qos_mgr cannot successfully allocate statistics resources due to resource exhaustion, the **qos interfaces interface voq-statistics-allocation-status** state command for the interface displays a status of **none**. In addition, the statistics under the following state paths reset to zero and do not increment:

- **qos interfaces interface output queues queue queue-statistics aggregate-statistics virtual-output-queue** <*x*>**-drop-probability ***
- **qos interfaces interface output queues queue queue-statistics per-lag-member-statistics member-interface virtual-output-queue** <*x*>**-drop-probability ***

If the interfaces are members of a LAG, each member interface behaves as described above. However, for the parent LAG, as long as one member interface has statistics resources allocated, the VOQ statistics related to the LAG do not reset. Instead, the statistics continue to increment based on the member interfaces that have VOQ statistics resources allocated.

If only a subset of the LAG member interfaces are allocated resources, the **qos interfaces interface voq-statistics-allocation-status** state command for the LAG displays a status of **partial**. If no member interfaces are allocated resources, the **qos interfaces interface voq-statistics-allocation-status** state command displays a status of **none**, and the VOQ statistics for the LAG reset to zero and do not increment.

Dropped traffic statistics

When **qos interfaces interface voq-statistics-allocation-status** displays as **none**, the following statistics in the **qos interfaces interface output queues queue queue-statistics** context do not reflect VOQ drops:

- **aggregate-statistics dropped-packets**
- **per-lag-member-statistics member-interface dropped-packets**
- **aggregate-statistics dropped-octets**
- **per-lag-member-statistics member-interface dropped-octets**

Retry

If **voq-statistics-allocation-status** for an interface is **none** or **partial**, you can free up resources in the global pool by deallocating resources from other QoS interfaces or unconfiguring some QoS interfaces altogether. In this case, to trigger a retry of VoQ statistics resource allocation on the desired interface, do the following:

1. Disable VOQ statistics on the interface (**voq-statistics false**)
2. Reenable VOQ statistics either explicitly (**voq-statistics true**), or implicitly by deleting the configuration altogether (**delete voq-statistics**).

No retry of failed resource allocation is triggered by configuring voq-statistics from **true** (implicit or explicit) to **false**.

19.1 Configuring VOQ statistics collection per interface

Procedure

To enable VOQ statistics collection on an ethernet interface, use the **voq-statistics** command. When **voq-statistics** is set to **true**, qos_mgr attempts to allocate VOQ statistics collection resources to that interface from the global pool.

If **voq-statistics** is explicitly set to **false**, qos_mgr deallocates any assigned resources for that port and returns the resources to the global pool.

Example: Configure VOQ statistics collection for an interface

```
--{ candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/2 voq-statistics
    qos {
        interfaces {
            interface ethernet-1/2 {
                voq-statistics true
            }
        }
    }
```

Example: Display VOQ statistics allocation status for an interface

To confirm whether VOQ statistics collection resources are allocated to the interface, use the **voq-statistics-allocation-status** state command. If the allocation of VOQ statistics to the interface is successful, the status is shown as **complete**. If the allocation is not successful (as a result of resource exhaustion), the status is shown as **none**.

```
--{ candidate shared default }--[  ]--
# info with-context from state qos interfaces interface ethernet-1/2 voq-statistics-
allocation-status
    qos {
        interfaces {
            interface ethernet-1/2 {
                voq-statistics-allocation-status complete
            }
        }
    }
```

19.2 Configuring VOQ statistics collection per LAG

Procedure

To enable VOQ statistics collection on a LAG, use the **voq-statistics** command. When **voq-statistics** is set to **true**, qos_mgr attempts to allocate VOQ statistics collection resources to all member interfaces of the LAG from a global pool.

If **voq-statistics** is explicitly set to **false**, qos_mgr deallocates any assigned resources for all member ports of the LAG, and returns the resources to the global pool.

Example: Configure VOQ statistics collection for a LAG

```
--{ candidate shared default }--[  ]--
# info with-context qos interfaces interface lag1 voq-statistics
    qos {
        interfaces {
            interface lag1 {
                voq-statistics true
            }
        }
    }
```

Example: Display VOQ statistics allocation status on a LAG

To confirm whether VOQ statistics collection resources are allocated to the LAG, use the **voq-statistics-allocation-status** state command. The status displayed is dependent on the status of the LAG's member interfaces. If all member interfaces are successfully assigned VOQ statistics resources, the LAG status is shown as **complete**. If no member interfaces are successfully assigned VOQ statistics resources, the status is shown as **none**. If there is a mix of successful and unsuccessful allocation across the member interfaces, then the status for the LAG is **partial**.

```
--{ candidate shared default }--[  ]--
# info with-context from state qos interfaces interface lag1 voq-statistics-allocation-
status
    qos {
        interfaces {
            interface lag1 {
                voq-statistics-allocation-status partial
            }
        }
    }
```

Example: Display VOQ statistics allocation status for LAG members

To identify which of the member interfaces have successful allocation of VOQ statistics resources, enter the **voq-statistics-allocation-status** state command for the member interfaces.

```
--{ + candidate shared default }--[ ]--
# info with-context from state interface ethernet-1/{1..3} voq-statistics-allocation-
status
    qos {
        interfaces {
            interface ethernet-1/1 {
                voq-statistics-allocation-status complete
            }
            interface ethernet-1/2 {
                voq-statistics-allocation-status none
```

```
            }
            interface ethernet-1/3 {
                voq-statistics-allocation-status complete
            }
        }
    }
```

20 Displaying QoS statistics

Procedure

To display traffic statistics for each output queue on an interface, use the **info from state qos interfaces interface** <*id*> **output queues queue queue-statistics** command.

Example: Display ethernet interface queue statistics

```
# info from state qos interfaces interface ethernet-1/1 output queues queue * queue-statistics
 aggregate-statistics | filter fields * | as table

+-------------+-----------+------+-------+--------+-------+-------+-------+-------+-------+--------+
| Interface   | Queue     |Last- |Transm |Transm  |Droppe |Droppe |Egq-dr |Egq-dr |Queue- |Queue-  |
|             |           |clear |itted- |itted-  |d-pack |d-     |opped- |opped- |depth  |depth   |
|             |           |      |packet |octets  |ets    |octets |packet |octets |last-  |high-t  |
|             |           |      |s      |        |       |       |s      |       |high-t |hresho  |
|             |           |      |       |        |       |       |       |       |hresho |ld-     |
|             |           |      |       |        |       |       |       |       |ld-    |bytes   |
|             |           |      |       |        |       |       |       |       |time   |        |
+=============+===========+======+=======+========+=======+=======+=======+=======+=======+========+
+-------------+-----------+------+-------+--------+-------+-------+-------+-------+-------+--------+
+=============+===========+======+=======+========+=======+=======+=======+=======+=======+========+
|ethernet-1/1 |multicast-0 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |multicast-1 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |multicast-2 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |multicast-3 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |multicast-4 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |multicast-5 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |multicast-6 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |multicast-7 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |unicast-0   |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |unicast-1   |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |unicast-2   |      |     5 |    340 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |unicast-3   |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |unicast-4   |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |unicast-5   |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |unicast-6   |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
|ethernet-1/1 |unicast-7   |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
+-------------+-----------+------+-------+--------+-------+-------+-------+-------+-------+--------+
```

Example: Display LAG interface queue statistics

```
# info from state qos interfaces interface lag1 output queues queue * queue-statistics aggregate-
statistics | filter fields * | as table
+-------------+-----------+------+-------+--------+-------+-------+-------+-------+-------+--------+
| Interface   | Queue     |Last- |Transm |Transm  |Droppe |Droppe |Egq-dr |Egq-dr |Queue- |Queue-  |
|             |           |clear |itted- |itted-  |d-pack |d-     |opped- |opped- |depth  |depth   |
|             |           |      |packet |octets  |ets    |octets |packet |octets |last-  |high-t  |
|             |           |      |s      |        |       |       |s      |       |high-t |hresho  |
|             |           |      |       |        |       |       |       |       |hresho |ld-     |
|             |           |      |       |        |       |       |       |       |ld-    |bytes   |
|             |           |      |       |        |       |       |       |       |time   |        |
+=============+===========+======+=======+========+=======+=======+=======+=======+=======+========+
+-------------+-----------+------+-------+--------+-------+-------+-------+-------+-------+--------+
+=============+===========+======+=======+========+=======+=======+=======+=======+=======+========+
| lag1        |multicast-0 |      |     0 |      0 |     0 |     0 |     0 |     0 |       |        |
```

```
| lag1        |multicast-1 |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |multicast-2 |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |multicast-3 |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |multicast-4 |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |multicast-5 |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |multicast-6 |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |multicast-7 |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |unicast-0   |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |unicast-1   |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |unicast-2   |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |unicast-3   |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |unicast-4   |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |unicast-5   |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
| lag1        |unicast-6   |      |     5 |     340 |     0 |     0 |     0 |     0 |       |        |
| lag1        |unicast-7   |      |     0 |       0 |     0 |     0 |     0 |     0 |       |        |
+-------------+------------+------+-------+--------+-------+-------+-------+-------+-------+--------+
```

Example: Display LAG member queue statistics (truncated output)

```
# info from state qos interfaces interface lag1 output queues queue * queue-statistics per-lag-member-
statistics member-interface ethernet-1/3 | filter fields * | as table
+--------------+--------------+--------------+--------------+--------------+--------------+
|   Interface  |    Queue     |   Member-    |  Last-clear  | Transmitted- | Transmitted- |
|              |              |  interface-  |              |   packets    |   octets     |
|              |              |    name      |              |              |              |
|              |              |              |              |              |              |
+==============+==============+==============+==============+==============+==============+
| lag1         | multicast-0  | ethernet-1/3 |              |            0 |            0 |
| lag1         | multicast-1  | ethernet-1/3 |              |            0 |            0 |
| lag1         | multicast-2  | ethernet-1/3 |              |            0 |            0 |
| lag1         | multicast-3  | ethernet-1/3 |              |            0 |            0 |
| lag1         | multicast-4  | ethernet-1/3 |              |            0 |            0 |
| lag1         | multicast-5  | ethernet-1/3 |              |            0 |            0 |
| lag1         | multicast-6  | ethernet-1/3 |              |            0 |            0 |
| lag1         | multicast-7  | ethernet-1/3 |              |            0 |            0 |
| lag1         | unicast-0    | ethernet-1/3 |              |            0 |            0 |
| lag1         | unicast-1    | ethernet-1/3 |              |            0 |            0 |
| lag1         | unicast-2    | ethernet-1/3 |              |            0 |            0 |
| lag1         | unicast-3    | ethernet-1/3 |              |            0 |            0 |
| lag1         | unicast-4    | ethernet-1/3 |              |            0 |            0 |
| lag1         | unicast-5    | ethernet-1/3 |              |            0 |            0 |
| lag1         | unicast-6    | ethernet-1/3 |              |            5 |          340 |
| lag1         | unicast-7    | ethernet-1/3 |              |            0 |            0 |
+--------------+--------------+--------------+--------------+--------------+--------------+
```

20.1 Clearing QoS statistics

Procedure

To reset the queue statistics counters for an interface or subinterface, use the **tools qos interfaces interface output queues** command.

Example: Reset all statistics counters on an interface

The following example resets all output queue statistics counters on an interface:

```
--{ running }--[  ]--
# tools qos interfaces interface eth-1/1 output queues clear-statistics
```

Example: Reset statistics counters for multicast egress queue

The following example resets statistics counters for a specified egress queue on an interface:

```
--{ running }--[  ]--
# tools qos interfaces interface eth-1/1 output queues queue queue-01 queue-statistics
 clear
```

20.2 QoS profile resource usage

A QoS profile resource refers to the number of classifier and rewrite policies that are applied to interfaces on a line card. Each classifier or rewrite policy that is applied to an interface on a line card counts as one profile resource used.

For example, if you create classifier policy `dscp1` and apply it to input IPv4 traffic on an interface, and apply the same `dscp1` policy to input IPv6 traffic on a different interface on the same line card, it counts as two classifier profile resources used.

The SR Linux supports up to 15 classifier profile resources and up to 32 rewrite profile resources per line card. You can display the number of QoS profile resources in use for each line card.

20.2.1 Displaying QoS profile resource usage on a 7250 IXR system

Procedure

To display QoS profile resource usage on a 7250 IXR system, use the **info from state** command.

The following example displays the number of used and free classifier and rewrite profile resources for a line card:

Example

```
# info with-context from state platform linecard 1 forwarding-complex 0 qos
    platform {
        linecard 1 {
            forwarding-complex 0 {
                qos {
                    resource classifier-profiles {
                        used 1
                        free 15
                    resource rewrite-profiles {
                        used 1
                        free 31
                    }
                }
            }
        }
    }
```

21 Priority-based Flow Control (PFC)

Note: PFC is supported on 7220 IXR-H4/H5/D4/D5 and 7250 IXR-6e/10e/18e/X1b/X3b platforms.

Priority-based flow control (PFC), based on the IEEE 802.1Qbb standard, is a link-level flow control mechanism that extends the capabilities of IEEE 802.3x-based Ethernet flow control. With the 802.3x standard, when a receiving interface experiences congestion, it can send a pause frame to the transmitting interface to suspend the flow of traffic for all priority values. PFC operates using a similar pause frame, but unlike 802.3x, the PFC pause frame can encode a different pause time for each of the eight different 802.1p CoS values. Instead of suspending traffic for all priority values at once, PFC suspends the flow of traffic for individual priority values. The pause time is measured in quanta, which is the time to transmit 512 bits (and where a quanta value of 0 indicates to unpause).

The main application of PFC is to support Fibre channel over Ethernet (FCoE). With FCoE, the FC-2 Fibre Channel layer assumes a lossless medium. When a receiving interface exceeds its buffer threshold, the interface sends pause frames to the transmitter to stop it from sending more FCoE frames.

SR Linux supports only statically-enabled PFC, and only for unicast traffic. SR Linux also supports the Data Center Bridging Capability Exchange (DCBX) protocol for the exchange of PFC data between peers to identify misconfigurations. However, autonegotiation of PFC settings using DCBX is not supported.

The PFC feature can be enabled on a per-interface and per-priority basis.

On an interface, PFC and traditional 802.3x-based Ethernet flow control (**interface ethernet flow-control**) are mutually exclusive.

Basic operation

With SR Linux, the basic operation of PFC is as follows:

- When an interface receives a packet at ingress, it is assigned a specific PFC queue based on mapping from the incoming packet's FC or dot1p value.
- The PFC feature monitors for congestion on the interface ingress buffers at the level of each PFC queue.
- When PFC detects congestion in a PFC queue based on exceeding the **mbs** (on 7220 IXR) or **pfc-on-threshold** (defined as a percentage of MBS on 7250 IXR), it signals the upstream traffic-transmitting interface to pause sending traffic of that particular priority.
- On the traffic-transmitting interface, a PFC priority can be associated with a single unicast egress queue. So when the traffic-transmitting interface receives a PFC pause frame from the traffic-receiving interface, it pauses traffic on the egress queue associated with that PFC priority.

The following sections provide platform-specific implementation details for PFC.

21.1 PFC on 7250 IXR platforms

The following sections describe PFC operation on 7250 IXR-6e/10e/18e/X1b/X3b platforms.

21.1.1 Ingress PFC operation on the traffic-receiving interface (7250 IXR)

On 7250 IXR-6e/10e/18e/X1b/X3b platforms, the operation of PFC on a traffic-receiving interface is a function of the following elements:

Mapping of forwarding classes to PFC queues

The incoming packets are mapped to one of eight PFC queues based on the forwarding class index of the packet, as determined by the applicable subinterface-level classification policy. The following table shows the default mapping of forwarding class index values to PFC queue values.

Table 14: Forwarding class index to PFC queue mapping

Forwarding class index	PFC queue
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7

You can also display this mapping using the **info from state qos interfaces interface** *<interface>* **pfc pfc-queue** *pfc** **forwarding-class** command.

In addition, the PFC queues are associated one-to-one with the PFC priority values; that is, PFC queue 0 is associated with priority 0, PFC queue 1 is associated with priority 1, and so on.

Receive buffer sections

The interface buffer serves eight PFC queues, each having a length equal to the maximum burst size (MBS), as defined in the PFC queue settings of the buffer allocation profile.

Pause frame generation thresholds (percentage of MBS)

The system generates PFC pause frames for a PFC queue after the queue length reaches the **pfc-on-threshold** value, and stops generating PFC pause frames after the queue length falls under the **pfc-off-threshold** value. Both values are defined in the buffer allocation profile as a percentage of the PFC queue MBS.

21.1.2 Egress PFC operation on the traffic-transmitting interface (7250 IXR)

On 7250 IXR-6e/10e/18e/X1b/X3b platforms, the operation of PFC on the traffic-transmitting (and therefore PFC pause frame-receving) interface is a function of the following elements:

Pause frame priority to egress queue mapping

The mapping of PFC pause frame priorities to egress queues is configurable in the PFC mapping profile. This profile determines which egress queues react to which PFC pause frame priorities. A single PFC pause frame priority can be configured per egress queue. When the transmitting system receives a PFC pause frame from a downstream receiver, it stops transmitting from the applicable queue. If the given egress queue is configured to react to a PFC priority of 1, it does so as long as the PFC pause frame contains a PFC priority of 1, whether or not traffic of other priorities is present in the queue.

Deadlock recovery

SR Linux also supports a deadlock recovery mechanism, which prevents permanent shutdown of the egress interface based on deadlock timers configured in the PFC mapping profile. When the deadlock timers are configured, if a queue receives PFC pause frames that prevent it from forwarding traffic for longer than the defined detection period, the system ignores the PFC pause frames on the queue and resumes forwarding traffic for a defined recovery period.

21.2 PFC on 7220 IXR platforms

The following sections describe PFC operation on 7220 IXR-H4/H5/D4/D5platforms.

21.2.1 Ingress PFC operation on the traffic-receiving interface (7220 IXR)

On 7220 IXR-H4/H5/D4/D5 platforms, the operation of PFC on a traffic-receiving interface is a function of the following elements:

Mapping of dot1p or forwarding class values to PFC queues

The PFC mapping profile maps ingress packets into one of eight PFC queues based on the packet's dot1p value (for tagged frames) or forwarding class (for untagged frames), as determined by the applicable subinterface classification policy. Multiple dot1p values or forwarding classes can be mapped into a single PFC queue. However, dot1p and forwarding class settings are mutually exclusive within a single PFC mapping profile.

Receive buffer allocation

The main difference in PFC operation between the 7250 IXR platforms and the 7220 IXR platforms is the size of the receive buffer. 7220 IXR-H4/H5/D4/D5 platforms have a smaller buffering capacity than the 7250 IXR platforms. As a result, the PFC queues share the interface buffer dynamically.

The buffer allocation profile includes the following options for allocating available buffer space to the PFC queues:

- The **maximum-burst-size** command allocates the maximum amount of shared buffer memory available for an individual PFC queue.
- The **maximum-pfc-reserved-share-percentage** command (or alternatively, **maximum-pfc-reserved-share-bytes**) defines the maximum level the PFC queue can take from the PFC reserved buffer per forwarding complex, to prevent starvation of other PFC queues.

PFC buffer reservation

On 7220 IXR-H4/H5/D4/D5 platforms, a PFC buffer reservation section is required for PFC queues to accommodate in-flight frames (the frames which are received after the PFC pause frame has been generated to the sender, but the sender has not yet reacted to it). The PFC buffer reservation section is implemented on each forwarding complex using the **qos linecard forwarding-complex input pfc-buffer-reservation** command. You must provision this buffer space to accommodate for in-flight frames depending on the number of PFC-enabled queues and their respective speed.

Pause frame generation based on MBS

In the PFC mapping profile, a single PFC pause frame priority can be mapped per individual PFC queue. When a PFC queue is congested (queue size reaches the MBS, as defined in the buffer allocation profile), the system generates a PFC pause frame indicating the priority that is experiencing the congestion. When the queue size falls below MBS, the system stops generating the PFC pause frames.

21.2.2 Egress PFC operation on the traffic-transmitting interface (7220 IXR)

On 7220 IXR-H4/H5/D4/D5 platforms, the operation of PFC on the traffic-transmitting (and therefore PFC pause frame-receiving) interface is a function of the following elements:

Pause frame priority to egress queue mapping

The mapping of PFC pause frame priorities to egress queues is configurable using a PFC mapping profile. This profile determines which egress queues react to which PFC pause frame priorities. A single PFC pause frame priority can be configured per egress queue. When the transmitting system receives a PFC pause from a downstream receiver, it stops transmitting from the applicable queue. If the egress queue is configured to react to a PFC priority of 1, it does so as long as the PFC pause frame contains the PFC priority of 1, whether or not traffic of other priorities is present in the queue. All traffic in the egress queue remains paused until either the pause time expires or the interface receives an unpause message from the downstream receiver.

Deadlock recovery

SR Linux supports a deadlock recovery mechanism, which prevents permanent shutdown of the egress queue based on deadlock timers configured in the PFC mapping profile. When the deadlock timers are configured, if a queue receives PFC pause frames that prevent it from forwarding traffic for longer than the defined detection period, the system ignores the PFC pause frames on the queue and resumes forwarding traffic for a defined recovery period.

21.3 PFC configuration

The following table describes the configuration elements available for the PFC feature.

Note: Changes to any existing PFC configuration are applied without the need to administratively disable and re-enable PFC.

Table 15: PFC configuration elements

Element	Description
PFC queue name and index	Defines a custom PFC queue name and index (similar to custom egress queue configuration)
PFC mapping profile	• Defines which egress queues are PFC-enabled and defines the mapping of priorities to those egress queues **Note:** If you enable PFC only for a subset of queues at egress, PFC is effectively disabled for the remaining queues in the upstream, transmitting node, because any PFC message sent on those queues is ignored by the receiving node. • Defines mapping of priorities to ingress PFC queues (on 7220 IXR-H4/H5/D4/D5) • Is applied to an interface
Buffer allocation profile	• Defines the PFC queue ingress buffer settings • Is applied to an interface
Ingress PFC administrative state (interface-level)	Determines whether PFC is enabled at ingress on the interface
PFC buffer reservation	Defines the PFC buffer reservation per forwarding complex (on 7220 IXR-H4/H5/D4/D5)

At ingress, the PFC feature can be enabled per interface only (using the **interface pfc-enable** command), while at egress, PFC can be enabled or disabled for each egress queue using a PFC mapping profile, which is then applied to an interface.

Default PFC mapping profile

By default, PFC is disabled on SR Linux interfaces. However, a default PFC mapping profile named **default** is attached to all interfaces. As a result, if the PFC feature is enabled at the interface level, the default profile is available to support the feature.

The PFC behavior can be altered using a custom-defined PFC mapping profile. When a new PFC mapping profile is created, all parameters are initially populated with the same values as the default profile.

To view the default profile, use the **info from state qos pfc-mapping-profile default** command.

Default PFC buffer allocation profile

The system also provides a default buffer allocation profile (**pfc-default**). To view this profile, use the **info from state qos buffer-management buffer-allocation-profile pfc-default** command.

21.3.1 Configuring PFC queue name and index (ingress)

Procedure

To configure a custom PFC queue name and associate it with an index, use the **qos queues pfc-queue** command.

Example: Configure a PFC queue

```
--{ + candidate shared default }--[  ]--
# info with-context qos queues pfc-queue pfc-0
    qos {
        queues {
            pfc-queue pfc-0 {
                queue-index 0
            }
        }
    }
```

21.3.2 Configuring PFC mapping profiles

About this task

SR Linux supports a maximum of three PFC mapping profiles.

Procedure

To configure a PFC mapping profile, use the **qos pfc-mapping-profile** command, which defines PFC settings using the following contexts:

- **pfc-priority** — configures the administrative state of the PFC priorities.
- **received-pfc-pause-frames** — configures the egress (PFC pause frame receiver) parameters.
- **received-traffic** (configurable on 7220 IXR-H4/H5/D4/D5 only) — configures the ingress (traffic receiver) parameters. At ingress, PFC can be enabled per interface only.

Note: On 7250 IXR-6e/10e/18e/X1b/X3b platforms, the ingress mapping of forwarding class indexes to PFC queues is static, therefore this option is not available. See Ingress PFC operation on the traffic-receiving interface (7250 IXR).

Example: Configure a PFC mapping profile (7250 IXR-6e/10e/18e/X1b/X3b)

The following example shows the PFC mapping profile configuration on 7250 IXR-6e/10e/18e/X1b/X3b platforms, including the following egress parameters (under **received-pfc-pause-frames**):

- **deadlock** — specifies the deadlock state and timers
- **queue** — specifies the PFC priority associated with the egress queue (referenced queue is the egress queue, not the PFC queue)

```
--{ + candidate shared default }--[  ]--
# info with-context qos pfc-mapping-profile custom-pfc-mapping-profile
    qos {
        pfc-mapping-profile custom-pfc-mapping-profile {
            pfc-priority 0 {
                pfc-enable true
```

```
            }
            received-pfc-pause-frames {
                deadlock {
                    enable true
                    detection-timer 750
                    recovery-timer 750
                }
                queue unicast-0 {
                    pfc-pause-frame-priority 0
                }
            }
        }
    }
```

Example: Configure a PFC mapping profile (7220 IXR-H4/H5/D4/D5)

The following example shows the PFC mapping profile configuration on 7220 IXR-H4/H5/D4/D5 platforms, including the following ingress parameter (under **received-traffic**):

- **pfc-queue** — defines the PFC queue mapping to PFC priority and to dot1p or forwarding class values

The example also configures the following egress settings (under **received-pfc-pause-frames**):

- **deadlock** — specifies the deadlock state and timers
- **queue** — specifies the PFC priority associated with the egress queue

```
--{ * candidate shared default }--[  ]--
# info with-context qos pfc-mapping-profile custom-pfc-mapping-profile
    qos {
        pfc-mapping-profile custom-pfc-mapping-profile {
            pfc-priority 0 {
                pfc-enable true
            }
            received-traffic {
                unicast-mapping {
                    pfc-queue pfc-0 {
                        forwarding-class fc0
                        pfc-pause-frame-priority 0
                   }
                }
            }
            received-pfc-pause-frames {
                deadlock {
                    enable true
                    detection-timer 750
                    recovery-timer 750
                }
                queue unicast-0 {
                    pfc-pause-frame-priority 0
                }
            }
        }
    }
```

21.3.3 Applying a PFC mapping profile to an interface

Procedure

To apply a PFC mapping profile to an interface, use the **qos interfaces interface pfc pfc-mapping-profile** command.

Example: Apply a PFC mapping profile to an interface

```
--{ + candidate shared default }--[  ]--
# info with-context qos interfaces interface eth-1/4
    qos {
        interfaces {
            interface eth-1/4 {
                interface-ref {
                    interface ethernet-1/4
                }
                pfc {
                    pfc-mapping-profile custom-pfc-mapping-profile
                }
            }
        }
    }
```

21.3.4 Configuring a buffer allocation profile for PFC

About this task

A buffer allocation profile can define settings for either PFC queues (as shown in this procedure) or for egress queues (see Buffer allocation profile). Because the buffer allocation profile can be applied to an interface under different contexts (**input** for PFC queues and **output** for egress queues), SR Linux blocks the configuration of PFC queues and egress queues in the same profile.

Procedure

To configure a buffer allocation profile for PFC, use the **qos buffer-management buffer-allocation-profile** command.

Example: Configure a buffer allocation profile for PFC (7250 IXR-6e/10e/18e/X1b/X3b)

The following example shows the buffer allocation profile configuration on 7250 IXR-6e/10e/18e/X1b/X3b platforms, including the following PFC queue ingress settings:

- **maximum-burst-size** — MBS
- **pfc-on-threshold**, **pfc-off-threshold** — PFC thresholds (on and off)

The 7250 IXR platforms support only one custom non-default buffer allocation profile.

```
--{ + candidate shared default }--[  ]--
# info with-context qos buffer-management buffer-allocation-profile custom-pfc-buffer-
profile
    qos {
        buffer-management {
            buffer-allocation-profile custom-pfc-buffer-profile {
                queues {
                    pfc-queue pfc-0 {
                        maximum-burst-size 102400
```

```
                        pfc-on-threshold 100
                        pfc-off-threshold 80
                    }
                }
            }
        }
    }
```

Example: Configure a buffer allocation profile for PFC (7220 IXR-H4/H5/D4/D5)

The following example shows the buffer allocation profile configuration on 7220 IXR-H4/H5/D4/D5 platforms, including the following PFC queue ingress settings:

- **maximum-burst-size** — MBS
- **maximum-pfc-reserved-share-percentage** — Maximum level the PFC queue can take from the PFC reserved buffer per forwarding complex

```
--{  candidate shared default }--[ qos buffer-management ]--
# info with-context qos buffer-management buffer-allocation-profile custom-pfc-buffer-
profile
    qos {
        buffer-management {
            buffer-allocation-profile custom-pfc-buffer-profile {
                queues {
                    pfc-queue pfc-0 {
                        maximum-burst-size 102400
                        maximum-pfc-reserved-share-percentage 10
                    }
                }
            }
        }
    }
```

21.3.5 Applying a PFC buffer allocation profile to an interface

Procedure

To apply a buffer allocation to an interface, use the **qos interfaces interface input pfc-buffer-allocation-profile** command.

Example: Apply buffer allocation profile for PFC

```
--{  candidate shared default }--[  ]--
#info with-context qos interfaces interface eth-1/4
    qos {
        interfaces {
            interface eth-1/4 {
                interface-ref {
                    interface ethernet-1/4
                }
                input {
                    pfc-buffer-allocation-profile custom-pfc-buffer-profile
                }
            }
        }
    }
```

21.3.6 Enabling ingress PFC on an interface

About this task

Ingress PFC refers to the ability of an interface to react to received PFC frames. With ingress PFC enabled, when the interface receives a PFC pause frame, it pauses the transmission of traffic from the egress queue associated with that priority. If ingress PFC is not enabled and the interface receives a pause frame, it continues to transmit traffic as normal.

Note: Enabling or disabling PFC on a per-interface or per-priority basis can cause packet loss. However, other PFC parameters such as MBS are modifiable without packet losss.

Prerequisites

To enable ingress PFC, you must first explicitly disable Ethernet flow control on the interface.

```
--{ + candidate shared default }--[  ]--
# info with-context interface ethernet-1/4 ethernet flow-control
    interface ethernet-1/4 {
        ethernet {
            flow-control {
                receive false
            }
        }
    }
```

Procedure

To enable ingress PFC on an interface, use the **pfc-enable true** command.

Example: Enable ingress PFC on an interface

```
--{  candidate shared default }--[  ]--
# info with-context qos interfaces interface eth-1/4
    qos {
        interfaces {
            interface eth-1/4 {
                interface-ref {
                    interface ethernet-1/4
                }
                pfc {
                    pfc-enable true
                }
            }
        }
    }
```

21.3.7 Configuring the PFC buffer reservation (7220 IXR-H4/H5/D4/D5)

Procedure

On 7220 IXR-H4/H5/D4/D5 platforms, to configure the PFC buffer reservation section, use the **qos linecard forwarding-complex input pfc-buffer-reservation** command. This command defines the buffer reservation section as a percentage of the total buffer available.

Example: Configure PFC buffer reservation (7220 IXR-H4/H5/D4/D5)

```
--{ * candidate shared default }--[  ]--
# info with-context qos linecard 1 forwarding-complex 0 input pfc-buffer-reservation
    qos {
        linecard 1 {
            forwarding-complex 0 {
                input {
                    pfc-buffer-reservation 1
                }
            }
        }
    }
```

On 7250 IXR-6e/10e/18e/X1b/X3b platforms, the PFC buffer reservation section is set to a fixed, non-configurable size; therefore, this configuration does not apply.

21.3.8 Displaying PFC headroom buffer utilization (7220 IXR-H4/H5/D4/D5)

Procedure

On 7220 IXR-H4/H5/D4/D5 platforms, you can display the PFC headroom buffer utilization using the **info from state platform linecard forwarding-complex buffer-memory pfc-headroom-buffer** command with one of the following options:

- **used** — displays the used PFC headroom buffer space
- **free** — displays the free PFC headroom buffer space

Example: Display PFC headroom buffer utilization (7220 IXR-H4/H5/D4/D5)

```
--{  candidate shared default }--[  ]--
# info with-context from state platform linecard 1 forwarding-complex 0 buffer-memory pfc-
headroom-buffer
    platform {
        linecard 1 {
            forwarding-complex 0 {
                buffer-memory {
                    pfc-headroom-buffer {
                        used 160782
                        free 0
                    }
                }
            }
        }
    }
```

21.3.9 Displaying per-interface PFC buffer usage

Procedure

To display buffer usage per PFC interface, use the **info from state qos interfaces interface pfc** command.

Example: Display per-interface PFC buffer usage (7220 IXR)

```
--{  candidate shared default }--[  ]--
```

```
# info from state qos interfaces interface ethernet-1/1 pfc
    pfc-mapping-profile 1
    pfc-enable true
    source-pfc-mac 00:00:5e:00:53:4B
    deadlock-detection-timer 0
    statistics {
        total-pfc-pause-frames-received 0
        total-pfc-pause-frames-generated 1294985
        total-packet-pfc-discards 0
        pfc-priority 0 {
            pfc-pause-frames-received 0
            pfc-pause-frames-generated 1298805
            pfc-transitions 0
            deadlock-recovery-occurrences 0
        }
...
        pfc-priority 7 {
            pfc-pause-frames-received 0
            pfc-pause-frames-generated 1298798
            pfc-transitions 0
            deadlock-recovery-occurrences 0
        }
    }
    pfc-queue pfc-0 {
        pfc-enable true
        pfc-maximum-burst-size 4738116
        pfc-maximum-pfc-reserved-share 3048
        pfc-buffer-used 4738116
        pfc-reserved-buffer-used 3048
    }
...
    pfc-queue pfc-7 {
        pfc-enable true
        pfc-maximum-burst-size 6409182
        pfc-maximum-pfc-reserved-share 1016
        pfc-buffer-used 6409182
        pfc-reserved-buffer-used 762
    }
```

Example: Display per-interface PFC buffer usage (7250 IXR)

```
--{  candidate shared default }--[  ]--
# info from state qos interfaces interface ethernet-1/1 pfc
    pfc-mapping-profile 1
    pfc-enable true
    source-pfc-mac 00:00:5e:00:53:38
    deadlock-detection-timer 0
    statistics {
        total-pfc-pause-frames-received 0
        total-pfc-pause-frames-generated 1447818
        total-packet-pfc-discards 0
        pfc-priority 0 {
            pfc-pause-frames-received 0
            pfc-pause-frames-generated 1447722
            pfc-transitions 0
            deadlock-recovery-occurrences 0
        }
...
        pfc-priority 7 {
            pfc-pause-frames-received 0
            pfc-pause-frames-generated 1447718
            pfc-transitions 0
            deadlock-recovery-occurrences 0
```

```
        }
    }
    pfc-queue pfc-0 {
        pfc-enable true
        forwarding-class fc0
        pfc-on-threshold-bytes 204544
        pfc-off-threshold-bytes 196352
        pfc-maximum-burst-size 206592
        pfc-buffer-used 224640
        peak-pfc-buffer-used 224640
    }
...
    pfc-queue pfc-7 {
        pfc-enable true
        forwarding-class fc7
        pfc-on-threshold-bytes 94464
        pfc-off-threshold-bytes 13824
        pfc-maximum-burst-size 113664
        pfc-buffer-used 103680
        peak-pfc-buffer-used 103680
    }
```

21.3.10 Displaying per-queue PFC buffer usage

Procedure

To display buffer usage per PFC queue on an interface, use the **info from state qos interfaces interface pfc pfc-queue** command with one of the following options:

- **pfc-buffer-used** – displays the PFC buffer usage by PFC queue
- **pfc-maximum-burst-size** – displays the actual maximum burst size of the PFC queue
- **pfc-maximum-pfc-reserved-share** (7220 IXR only) – displays the actual maximum share the PFC queue can take from the PFC reserved buffer configured for a given forwarding complex
- **pfc-reserved-buffer-used** (7220 IXR only) – displays the reserve PFC buffer usage by PFC queue
- **forwarding-class** (7250 IXR only) – displays the forwarding class mapped to the specified PFC queue
- **peak-pfc-buffer-used** (7250 IXR only) – displays the peak value for PFC buffer usage by PFC queue (the highest value since the last reboot)
- **pfc-off-threshold-bytes** (7250 IXR only) – displays the actual off-threshold of the PFC queue
- **pfc-on-threshold-bytes** (7250 IXR only) – displays the actual on-threshold of the PFC queue

Example: Display per-queue PFC buffer usage (7220 IXR)

```
# info from state qos interfaces interface ethernet-1/1 pfc pfc-queue pfc-0
    pfc-enable true
    pfc-maximum-burst-size 4738116
    pfc-maximum-pfc-reserved-share 3048
    pfc-buffer-used 4738116
    pfc-reserved-buffer-used 3048
```

Example: Display per-queue PFC buffer usage (7250 IXR)

```
# info from state qos interfaces interface ethernet-1/1 pfc pfc-queue pfc-0
    pfc-enable true
    forwarding-class fc0
```

```
    pfc-on-threshold-bytes 204544
    pfc-off-threshold-bytes 196352
    pfc-maximum-burst-size 206592
    pfc-buffer-used 256608
    peak-pfc-buffer-used 256608
```

21.3.11 Displaying PFC statistics

Procedure

To display PFC statistics, use the **info from state** command.

Example: Display PFC statistics

The following example displays PFC statistics for 7250 IXR platforms. On 7220 IXR-H4/H5/D4/D5 platforms, the output differs in that the `pfc-on-threshold-bytes` and `pfc-off-threshold-bytes` fields do not apply.

```
--{ + candidate shared default }--[  ]--
# info with-context from state qos interfaces interface eth-1/4 pfc
    qos {
        interfaces {
            interface eth-1/4 {
                pfc {
                    pfc-mapping-profile 1
                    source-pfc-mac A8:24:B8:82:E7:70
                    oper-state up
                    deadlock-detection-timer 0
                    statistics {
                        total-pfc-pause-frames-received 0
                        total-pfc-pause-frames-generated 9906478
                        total-packet-pfc-discards 0
                        pfc-priority 0 {
                            pfc-pause-frames-received 0
                            pfc-pause-frames-generated 9906478
                            pfc-transitions 0
                        }
                        pfc-priority 1 {
                            pfc-pause-frames-received 0
                            pfc-pause-frames-generated 0
                            pfc-transitions 0
                        }
                        pfc-priority 2 {
                            pfc-pause-frames-received 0
                            pfc-pause-frames-generated 0
                            pfc-transitions 0
                        }
                        pfc-priority 3 {
                            pfc-pause-frames-received 0
                            pfc-pause-frames-generated 0
                            pfc-transitions 0
                        }
                        pfc-priority 4 {
                            pfc-pause-frames-received 0
                            pfc-pause-frames-generated 0
                            pfc-transitions 0
                        }
                        pfc-priority 5 {
                            pfc-pause-frames-received 0
                            pfc-pause-frames-generated 0
                            pfc-transitions 0
```

```
            }
            pfc-priority 6 {
                pfc-pause-frames-received 0
                pfc-pause-frames-generated 0
                pfc-transitions 0
            }
            pfc-priority 7 {
                pfc-pause-frames-received 0
                pfc-pause-frames-generated 0
                pfc-transitions 0
            }
        }
        pfc-queue pfc-0 {
            pfc-on-threshold-bytes 230400
            pfc-off-threshold-bytes 179456
            pfc-maximum-burst-size 256000
            pfc-maximum-pfc-reserved-share 10485760
            forwarding-class [
                fc0
            ]
        }
        pfc-queue pfc-1 {
            pfc-on-threshold-bytes 230400
            pfc-off-threshold-bytes 179456
            pfc-maximum-burst-size 256000
            pfc-maximum-pfc-reserved-share 10485760
            forwarding-class [
                fc1
            ]
        }
        pfc-queue pfc-2 {
            pfc-on-threshold-bytes 230400
            pfc-off-threshold-bytes 179456
            pfc-maximum-burst-size 256000
            pfc-maximum-pfc-reserved-share 10485760
            forwarding-class [
                fc2
            ]
        }
        pfc-queue pfc-3 {
            pfc-on-threshold-bytes 230400
            pfc-off-threshold-bytes 179456
            pfc-maximum-burst-size 256000
            pfc-maximum-pfc-reserved-share 10485760
            forwarding-class [
                fc3
            ]
        }
        pfc-queue pfc-4 {
            pfc-on-threshold-bytes 230400
            pfc-off-threshold-bytes 179456
            pfc-maximum-burst-size 256000
            pfc-maximum-pfc-reserved-share 10485760
            forwarding-class [
                fc4
            ]
        }
        pfc-queue pfc-5 {
            pfc-on-threshold-bytes 230400
            pfc-off-threshold-bytes 179456
            pfc-maximum-burst-size 256000
            pfc-maximum-pfc-reserved-share 10485760
            forwarding-class [
                fc5
```

```
                    ]
                }
                pfc-queue pfc-6 {
                    pfc-on-threshold-bytes 230400
                    pfc-off-threshold-bytes 179456
                    pfc-maximum-burst-size 256000
                    pfc-maximum-pfc-reserved-share 10485760
                    forwarding-class [
                        fc6
                    ]
                }
                pfc-queue pfc-7 {
                    pfc-on-threshold-bytes 230400
                    pfc-off-threshold-bytes 179456
                    pfc-maximum-burst-size 256000
                    pfc-maximum-pfc-reserved-share 10485760
                    forwarding-class [
                        fc7
                    ]
                }
            }
        }
    }
}
```

22 DCBX

Note: DCBX is supported on the 7220 IXR-H4/H5, 7220 IXR-D4/D5, 7250 IXR Gen 2, and 7250 IXR Gen 3 platforms.

Data Center Bridging eXchange (DCBX) protocol is a discovery and exchange protocol for advertising configurations and capabilities between directly connected peers. In addition to propagating configurations, DCBX also allows for the detection of misconfigurations between peers.

DCBX is defined by Section 38 of the IEEE 802.1Q-2022 specification. The DCBX protocol information is propagated by LLDP using TLVs as defined in Annex D.2.10 of the 802.1Q specification.

Because LLDP is a unidirectional protocol, each node sends its local configuration to its neighbor, and the remote neighbor's state machine determines how to process and apply the received information.

DCBX can support the exchange of PFC information.

Note: SR Linux supports only IEEE DCBX (OUI type 0x0080c2) and the PFC configuration TLV (0x0b).

DCBX operation

In SR Linux, DCBX is enabled on every interface by default. The DCBX TLV for PFC propagates the status for each PFC priority (0 to 7) to the peer as follows:

- If the interface has one or more PFC priorities enabled, DCBX advertises the per-priority status (enabled or disabled) for all PFC priorities.
- If the interface has no PFC priorities enabled, DCBX advertises all PFC priorities as disabled.

In state, the system maintains information about the operational state of DCBX for the local node and for the remote peer. With SR Linux, all DCBX-enabled interfaces are effectively in **unwilling** state, which means the system never reacts to the state received from the remote peer. Instead, the system maintains the local and remote state to be available for display. Any discrepancy identified between the local and remote state allows for the detection of misconfigurations. You can then update the configuration as required to address the discrepancy.

22.1 Configuring DCBX

Prerequisites

To enable DCBX, LLDP must be explicitly enabled at the system level. For more information see the "LLDP" chapter of the *SR Linux Interfaces Guide*.

About this task

By default, DCBX is enabled on every interface. It is possible to disable it administratively under the **qos interfaces interface dcbx admin-state** context. If disabled, LLDP stops advertising the DCBX capability.

Procedure

To configure DCBX on an interface, use the **dcbx admin-state** command in the **qos interfaces interface** context.

Example: Configuring DCBX

The following example enables DCBX on interface eth-1/4.

```
--{ candidate shared default }--[  ]--
# info with-context qos interfaces interface eth-1/4
    qos {
        interfaces {
            interface eth-1/4 {
                interface-ref {
                    interface ethernet-1/4
                }
                dcbx {
                    admin-state enable
                }
            }
        }
    }
```

The DCBX configuration is at interface level only. There is no system-level **admin-state** setting for DCBX.

Note: DCBX operates over point-to-point links only. If DCBX detects multiple peers on a single interface, the system operationally disables DCBX on the interface.

Example: Displaying DCBX state information

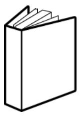
To display the local and remote operational state for DCBX, use the **info from state** command.

```
--{ state }--[  ]--
# info from state qos interfaces interface eth-1/4 dcbx
    admin-state disable
    oper-state down
    oper-state-reason dcbx-admin-disabled
    pfc-priority 0 {
        oper-state down
        remote-state remote-down
    }
    pfc-priority 1 {
        oper-state down
        remote-state remote-down
    }
...
    pfc-priority 7 {
        oper-state down
        remote-state remote-down
    }

--{ state }--[  ]--
# info from state qos interfaces interface ethernet-2/3 dcbx
    admin-state enable
    oper-state down
    oper-state-reason remote-dcbx-down
    pfc-priority 0 {
        oper-state up
        remote-state remote-down
    }
```

```
...
    pfc-priority 7 {
        oper-state up
        remote-state remote-down
    }


--{ state }--[  ]--
# info from state qos interfaces interface ethernet-1/3 dcbx
    admin-state enable
    oper-state down
    oper-state-reason lldp-oper-state-down
    pfc-priority 0 {
        oper-state up
        remote-state remote-down
    }
...
    pfc-priority 7 {
        oper-state up
        remote-state remote-down
    }
```

If an interface has DCBX enabled, but it does not receive a DCBX capability message from the peer (DCBX is disabled on the remote node), the DCBX state is shown as **oper-down**, with a reason code of `remote-dcbx-down`.

Customer document and product support

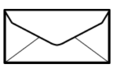


Customer documentation

Customer documentation welcome page

Technical support

Product support portal



Documentation feedback

Customer documentation feedback