Nokia Service Router Linux
7730 Service Interconnect Router
Release 25.7

7730 SXR Quality of Service Guide

3HE 21401 AAAB TQZZA
Edition: 01
July 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) on 7730 SXR series platforms.

Note:
This guide describes QoS feature support on 7730 SXR series platforms only. For information about QoS feature support on 7220 IXR and 7250 IXR platforms, see the *7220 IXR and 7250 IXR 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 about 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
Pre-classification	How QoS works on 7730 SXR platforms Pre-classification

3 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**.

4 Named queues and forwarding classes

7730 SXR platforms provide support for both named queues and named forwarding classes. Packets that require a similar treatment (per-hop behavior) are grouped into the same forwarding class, also known as a behavior aggregate. 7730 SXR platforms differentiate up to 16 forwarding classes.

By default, forwarding classes have system-reserved names, `fc0` to `fc15`, that map to system-reserved queues, `queue-0` to `queue-11`.

As part of the forwarding class and queue configurations, 7730 SXR platforms provide 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.
- Even though they do not appear as named forwarding classes in the configuration, the default forwarding class names `fc0` to `fc15` always exist and are reserved names.
- Even though they do not appear as named queues in the configuration, the default queue names `queue-0` to `queue-11` always exist and are 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 (`queue-0` to `queue-11`) with default parameters.
- If you configure a named forwarding class (for example, `forwarding-class-A`) and assign it a forwarding class index of 3, any subsequent configuration that references the default forwarding class associated with index 3 (`fc3`) fails. You must always reference the named forwarding class when it is configured.

Each queue of an egress interface is associated with a scheduler node. The mapping of queues to scheduler nodes is platform-dependent.

Related topics

Egress queue mapping

4.1 Configuring named queues

About this task

Procedure

Use the **qos queues queue** command to configure a name and index for a queue.

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

Example: Configure queue name

```
--{ candidate shared default }--[  ]--
# info with-context qos queues queue queue-1
    qos {
        queues {
            queue queue-1 {
                queue-index 1
            }
        }
    }
```

4.2 Configuring forwarding class names and queue associations

About this task

On an interface, packets are assigned to egress queues based on FC-to-queue mapping. To change the default mapping on interfaces, configure a custom FC-to-queue mapping using the **qos forwarding classes** command.

On a subinterface, by default all traffic uses the same interface-level queues. To associate queues and forwarding classes to a subinterface, configure an output class map using the **qos output-class-map** command and assign it to the subinterface (see Egress queue mapping).

Procedure

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

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

Example: Configure forwarding class name and queue association

```
--{ candidate shared default }--[  ]--
# info with-context qos forwarding-classes
    qos {
        forwarding-classes {
            forwarding-class fc1 {
                forwarding-class-index 1
                output {
                    queue queue-1
                }
            }
        }
    }
```

You can reference the named forwarding class in policies including DSCP classification and marking, dot1p classification and marking, MPLS traffic-class classification and marking, and the ingress policer policy.

5 How QoS works for router-originated traffic

This section describes how QoS applies to traffic that originates on 7730 SXR platforms.

Default forwarding class and profile for router-originated traffic

The marking of system-generated-traffic on 7730 SXR platforms follows egress remarking policies as they are attached to the respective subinterface, using forwarding class and profile values defined in the following table.

LACP and LLDP packets do not have a subinterface context as they are interface-based protocols. As a consequence, the 7730 SXR forwarding path is unable to account for these packets under interface queue statistics, as the statistics object is fetched from the subinterface record.

Table 1: Default forwarding class and profile for router-originated traffic

Protocol/Message	forwarding-class-index	profile	resource-priority
IPv4 ARP request/reply	6	out	defined by FC rsource priority table
ICMPv4 including echo-request[1], echo-reply[2], dest-unreachable, redirect, time-exceeded, parameter-problem	0	in	defined by FC rsource priority table
ICMPv4 echo-request with ToS/DSCP override = *X*	Look up *X* in system-generated-traffic-mapping table	Look up *X* in system-generated-traffic-mapping table	defined by FC rsource priority table
ICMPv4 echo-reply to echo-request with non-zero DSCP *X*	Look up *X* in system-generated-traffic-mapping table	Look up *X* in system-generated-traffic-mapping table	defined by FC rsource priority table
UDP traceroute	0	out	defined by FC rsource priority table
IPv6 neighbor solicitation	6	out	defined by FC rsource priority table
IPv6 neighbor advertisement	6	out	defined by FC rsource priority table
All other ICMPv6 including dest unreachable, packet-	0	in	defined by FC rsource priority table

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/Message	forwarding-class-index	profile	resource-priority
too-big, time-exceeded, parameter-problem, echo-request[1], echo-reply[2], router-solicitation, redirect			
ICMPv6 echo-request with DSCP override = *X*	Look up *X* in system-generated-traffic-mapping table	Look up *X* in system-generated-traffic-mapping table)	defined by FC rsource priority table
ICMPv6 echo-reply to echo-request with non-zero DSCP *X*	Look up *X* in system-generated-traffic-mapping table)	Look up *X* in system-generated-traffic-mapping table	defined by FC rsource priority table
BFD	6	out	defined by FC rsource priority table
BGP	6	out	defined by FC rsource priority table
OSPF	6	out	defined by FC rsource priority table
DHCP/DHCPv6	4	out	defined by FC rsource priority table
DNS query	4	out	defined by FC rsource priority table
FTP/TFTP	4	out	defined by FC rsource priority table
gNMI	4	out	defined by FC rsource priority table
JSON RPC	4	out	defined by FC rsource priority table
LACP	6	out	defined by FC rsource priority table
LLDP	4	out	defined by FC rsource priority table
NTP	4	out	defined by FC rsource priority table
RADIUS	4	out	defined by FC rsource priority table
sflow	0	out	defined by FC rsource priority table

Protocol/Message	forwarding-class-index	profile	resource-priority
SNMP	4	out	defined by FC rsource priority table
SSH	4	out	defined by FC rsource priority table
Syslog	4	out	defined by FC rsource priority table
TACACS+	4	out	defined by FC rsource priority table
IS-IS	6	out	defined by FC rsource priority table

Default DSCP to forwarding class index and profile values

On 7730 SXR platforms, you can specify a DSCP value for ICMP messages. The following table shows the mapping of DSCP to corresponding forwarding class and profile values. This mapping is also available in state using the **info from state qos system-generated-traffic** command, to determine which DSCP value to use to generate traffic for a given forwarding class and profile combination.

Table 2: Default DSCP to forwarding class index and profile values

DSCP value	Forwarding class index	Profile
0	0	out
1	0	in
2	0	in-plus
3	0	exceed
4	8	out
5	8	in
6	8	in-plus
7	8	exceed
8	1	out
9	13	out
10	2	in-plus
11	13	in
12	2	in
13	13	in-plus
14	2	out
15	2	exceed

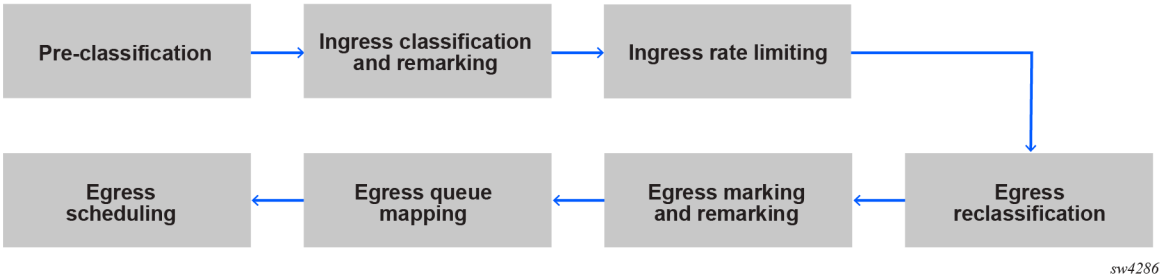
DSCP value	Forwarding class index	Profile
16	1	in
17	13	exceed
18	3	in-plus
19	14	out
20	3	in
21	14	in
22	3	out
23	3	exceed
24	1	in-plus
25	14	in-plus
26	4	in-plus
27	14	exceed
28	4	in
29	15	out
30	4	out
31	4	exceed
32	1	exceed
33	15	in
34	5	in-plus
35	15	in-plus
36	5	in
37	12	out
38	5	out
39	5	exceed
40	7	out
41	7	exceed
42	12	in
43	12	in-plus
44	6	in
45	6	out

DSCP value	Forwarding class index	Profile
46	6	in-plus
47	6	exceed
48	7	in
49	7	in-plus
50	9	out
51	9	in
52	9	in-plus
53	9	exceed
54	12	exceed
55	11	exceed
56	7	in-plus
57	10	out
58	10	in
59	10	in-plus
60	10	exceed
61	11	out
62	11	in
63	11	in-plus

6 How QoS works on 7730 SXR platforms

The following chapters describe QoS functionality on 7730 SXR platforms. These descriptions break down the QoS packet processing into their individual stages as shown in the following figure.

Figure 1: How QoS works (7730 SXR)



7 Pre-classification

Pre-classification provides drop protection for ingress packets accessing the 7730 SXR chipset to prevent low priority packets from starving high priority packets before the SR Linux software can perform ingress classification.

The system-wide settings under **qos resource-management pre-classification** define which packet types the pre-classifier protects from being dropped. During periods of congestion, unlisted packet types may be dropped, in which case the interface drop counters are incremented.

The following table describes the supported pre-classifier types.

Table 3: Pre-classifier types

Pre-classifier type	Description
dscp	Provides protection for tagged or untagged IPv4 and IPv6 packets (with outer Ethertype of IPv4 or IPv6)
mpls-traffic-class	Provides protection for tagged or untagged MPLS packets (with outer Ethertype of MPLS)
dot1p	Provides protection for tagged packets that are not IPv4, IPv6, or MPLS

No protection is available for untagged packets that are not IPv4, IPv6, or MPLS.

Protected control packets

In addition to the custom-defined protected packet types, the pre-classifier automatically protects all control packets from the following protocols:

- ARP
- PPPoE
- LLDP
- MACSEC
- PBB
- CFM
- ICMPv6
- IGMP
- RSVP
- EIGRP
- VRRP
- IPSec
- ISIS
- LACP

- ESMC
- PIM
- LDP
- OSPF
- BGP
- PIM-Register
- PIM-RegisterStop
- ELMI
- Dot1x
- EFM-OAM
- PTP

Default pre-classifier settings

The following are the default pre-classification settings.

```
--{ candidate shared default }--[  ]--
# info with-context from state qos resource-management pre-classification
    qos {
        resource-management {
            pre-classification {
                dot1p 6 {
                    action protect
                }
                dot1p 7 {
                    action protect
                }
                dscp 48 {
                    action protect
                }
                dscp 56 {
                    action protect
                }
                mpls-traffic-class 6 {
                    action protect
                }
                mpls-traffic-class 7 {
                    action protect
                }
            }
        }
    }
```

7.1 Configuring the pre-classifier

Procedure

To configure the pre-classifier, use the **qos resource-management pre-classification** command.

Example: Configure the pre-classifier

The following example configures the pre-classifier to protect packets with a dot1p value of 6, a DSCP value of CS0, or an MPLS traffic-class value of 6.

```
--{ candidate shared default }--[  ]--
# info with-context qos resource-management pre-classification
    qos {
        resource-management {
            pre-classification {
                dot1p 6 {
                    action protect
                }
                dscp CS0 {
                    action protect
                }
                mpls-traffic-class 6 {
                    action protect
                }
            }
        }
    }
```

8 Ingress classification: dot1p, MPLS traffic-class, and DSCP

7730 SXR platforms support ingress classification using classifier policies that are applied at the subinterface level. Three types of ingress classifier policies are supported: dot1p, MPLS traffic-class, and DSCP policies. All classifier policies contain two basic elements: classification rules and associated actions.

The classification rule can match against the incoming packet's dot1p, MPLS traffic-class, or DSCP value. When a packet matches an ingress classification rule, the classifier policy can assign one or more of the following values to the packet.

- forwarding class
- profile (for color-aware policing)
- discard eligibility bit
- IP rewrite policy

Classifier policy order of precedence

7730 SXR platforms apply the ingress classifier policies in the following order:

1. Default forwarding class and profile settings for the subinterface
2. Dot1p policy
3. MPLS traffic-class policy
4. DSCP policy

If a packet matches a classification rule in each of the above policies, the last matching rule (for example, from the DSCP policy) is applied.

8.1 Actions

When a packet matches an ingress classification rule, the classifier policy can assign one or more of the following values to the packet.

- **forwarding class**

  The forwarding class is identified by the forwarding class name.

- **profile**

  The profile can be one of the following values:

  – **exceed**

  – **in**

  – **in-low**

  – **in-plus**

  - **out**
  - **out-low**

  Nokia recommends the use of the **exceed**, **in**, **in-plus**, or **out** profiles, which are used as input for color-aware policing at ingress. Post-ingress policing profile values are limited to these four values.

- **IP rewrite policy**

  The IP rewrite policy provides rules to remark DSCP or IP precedence values based on the profile of the packet. The rewrite policy applies only to IP packets on a Layer 3 interface.

  By default, no IP rewrite policy is applied.

- **de-out-profile**

  The discard eligiblity (DE) bit setting determines whether the packet is marked as lower priority and can therefore be dropped first when the network experiences congestion.

  If a packet's dot1p field has the discard eligibility indicator (DEI) bit set to 1 (true), and the packet matches a classification rule that has the **de-out-profile** parameter set to **true**, the profile state for the packet is set to out-low instead of the profile value defined by the rule. The DEI state is maintained through the classification pipeline that allows each rule evaluation to determine the current DEI state for the packet being classified.

  Note: A lower precedence level classification rule that sets the profile to out-low based on the **de-out-profile true** setting is overwritten by any higher precedence matching classification rule that specifies a profile setting.

- **default action**

  If a packet does not match any of the classification rules defined in the classifier policies, the default forwarding class and profile settings under **qos interfaces interface** <*name*> **input classifiers default** are applied to the packet. Unless the default settings are configured otherwise, the default forwarding class is **fc0**, and the default profile is **out**.

8.2 Dot1p policy

The IEEE 802.1p (dot1p) classifier policy classifies packets based on the 3-bit Priority Code Point (PCP) field in the VLAN tag. The dot1p policy can match traffic based on the dot1p priority value from 0 to 7.

The following table describes which dot1p bits the policies use for classification, depending on the type of encapsulation applied to the subinterface and packet.

Table 4: Dot1p policy classification criteria

Subinterface encapsulation	Packet encapsulation	Dot1p bits used for classification
Null	No VLAN tag	None
Null	Dot1Q	Dot1Q—P-bits
Null	TopQ/BottomQ	TopQ—P-bits
Dot1Q	No VLAN tag (default subinterface)	None

Subinterface encapsulation	Packet encapsulation	Dot1p bits used for classification
Dot1Q	Do1Q	Dot1Q—P-bits
QinQ/TopQ	Dot1Q	Dot1Q—P-bits
QinQ/TopQ	TopQ/BottomQ	TopQ—P-bits
QinQ/QinQ	TopQ/BottomQ	Determined by the **match-qinq-dot1p** parameter (under **qos interfaces interface** *<name>* **input classifiers**), which defines whether classification is based on the **outer** or **inner** P-bits (default: **outer**)

8.2.1 Configuring dot1p classifiers

Prerequisites

Before referencing a forwarding class in any QoS policy, the forwarding class must first be explicitly mapped to an output queue. For more information, see Named queues and forwarding classes.

Procedure

To configure a dot1p classifier policy, use the **qos classifiers dot1p-policy** command.

Example: Configure dot1p classifiers

```
--{ candidate shared default }--[  ]--
# info with-context qos classifiers dot1p-policy dot1p-policy-name dot1p 0
    qos {
        classifiers {
            dot1p-policy dot1p-policy-name {
                dot1p 0 {
                    forwarding-class fc0
                    profile in
                    de-out-profile true
                    ip-rewrite-policy ip-rewrite-name
                }
            }
        }
    }
```

8.3 MPLS traffic-class policy

MPLS traffic-class policies classify incoming packets based on the EXP bits of the outer MPLS label.

To provide support for the short-pipe model for MPLS-based aggregation networks, the **ler-use-dscp** option can overrule the MPLS traffic-class policy. If **ler-use-dscp** is set to **true** under the **qos interfaces interface** *<interface-id>* **input classifiers** context, it forces DSCP-based classification for all terminating LSP bindings. This corresponds to the so-called short-pipe model for MPLS based aggregation networks.

8.3.1 Configuring MPLS traffic-class policies

Prerequisites

Before referencing a forwarding class in any QoS policy, the forwarding class must first be explicitly mapped to an output queue. For more information, see Named queues and forwarding classes.

Procedure

To configure an MPLS traffic-class policy, use the **qos classifiers mpls-traffic-class-policy** command.

Example: Configure MPLS traffic-class policies

```
--{ + candidate shared default }--[  ]--
# info with-context qos classifiers mpls-traffic-class-policy mpls-policy-name
    qos {
        classifiers {
            mpls-traffic-class-policy mpls-policy-name {
                traffic-class 0 {
                    forwarding-class fc0
                    profile in
                    de-out-profile true
                    ip-rewrite-policy ip-rewrite-name
                }
            }
        }
    }
```

8.4 DSCP policy

DSCP policies classify packets based on the DSCP value of incoming IP packets. If the datapath cannot determine whether the incoming packet is an IP packet (for example on L2 interfaces after parsing subinterface delimiting encapsulation), the DSCP-based classification is skipped.

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 action.

8.4.1 Configuring DSCP classifiers

Prerequisites

Before referencing a forwarding class in any QoS policy, the forwarding class must first be explicitly mapped to an output queue. For more information, see Named queues and forwarding classes.

Procedure

To configure a DSCP classifier policy, use the **qos classifiers dscp-policy** command.

Example: Configure DSCP classifiers

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

```
        classifiers {
            dscp-policy dscp-policy-name {
                dscp AF11 {
                    forwarding-class fc0
                    profile in
                    de-out-profile true
                    ip-rewrite-policy ip-rewrite-name
                }
            }
        }
    }
```

8.5 Policy application to subinterfaces

7730 SXR platforms perform ingress classification at subinterface level. As a result, classifier policies are applied to individual subinterfaces.

To enable QoS features on a subinterface, you must first configure a custom 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.

All ingress classifier policies are created in the **qos classifiers** container. To apply a policy to a subinterface, reference the required policy name under the **qos interfaces interface** <*interface-id*> **input classifiers** context.

Default subinterface policy

By default when a subinterface is created, no classifier policies are assigned. In this case, the default forwarding class and profile values apply (default: **fc0** and **out**).

8.5.1 Applying classifier policies to subinterfaces

Procedure

To apply classifier policies to a subinterface, use the **qos interfaces interface input classifiers** command.

If the **ler-use-dscp** parameter is set to **true**, it overrules the MPLS traffic-class policy and forces DSCP-based classification for all terminating LSP bindings. The **match-qinq-dot1p** parameter defines whether classification is based on the outer or inner P-bits (default: **outer**). And the **tos-rewrite-state** parameter defines whether a subinterface is considered as trusted or untrusted for the purpose of DSCP remarking.

Example: Apply classifier policies to subinterfaces

```
--{ + candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-1/1.1
    qos {
```

```
        interfaces {
            interface ethernet-1/1.1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
                input {
                    classifiers {
                        ler-use-dscp true
                        match-qinq-dot1p outer
                        tos-rewrite-state trusted
                        dscp-policy dscp-policy-name
                        dot1p-policy dot1p-policy-name
                        mpls-traffic-class-policy mpls-policy-name
                    }
                }
            }
        }
    }
```

8.6 Configuring the default forwarding class and profile

Prerequisites

Before referencing a forwarding class in any QoS policy, the forwarding class must first be explicitly mapped to an output queue. For more information, see Named queues and forwarding classes.

Procedure

You can configure the default forwarding class and profile for input packets arriving on a subinterface that do not match any classification rule. Unless the default settings are configured otherwise, the default forwarding class is **fc0**, and the default profile is **out**.

Example: Configure default forwarding class and profile for a subinterface

```
--{ candidate shared default }--[ ]--
# info with-context qos interfaces interface eth-1/1.1 input classifiers default
    qos {
        interfaces {
            interface eth-1/1.1 {
                input {
                    classifiers {
                        default {
                            forwarding-class test-fc
                            profile out
                        }
                    }
                }
            }
        }
    }
```

9 Ingress policing

To perform ingress policing, 7730 SXR platforms support two two-rate-three-color marker (trTCM) algorithms using the following **algorithm-type** parameters:

- **trtcm1**: RFC 2698 trTCM
- **trtcm2** (default): RFC 4115 trTCM

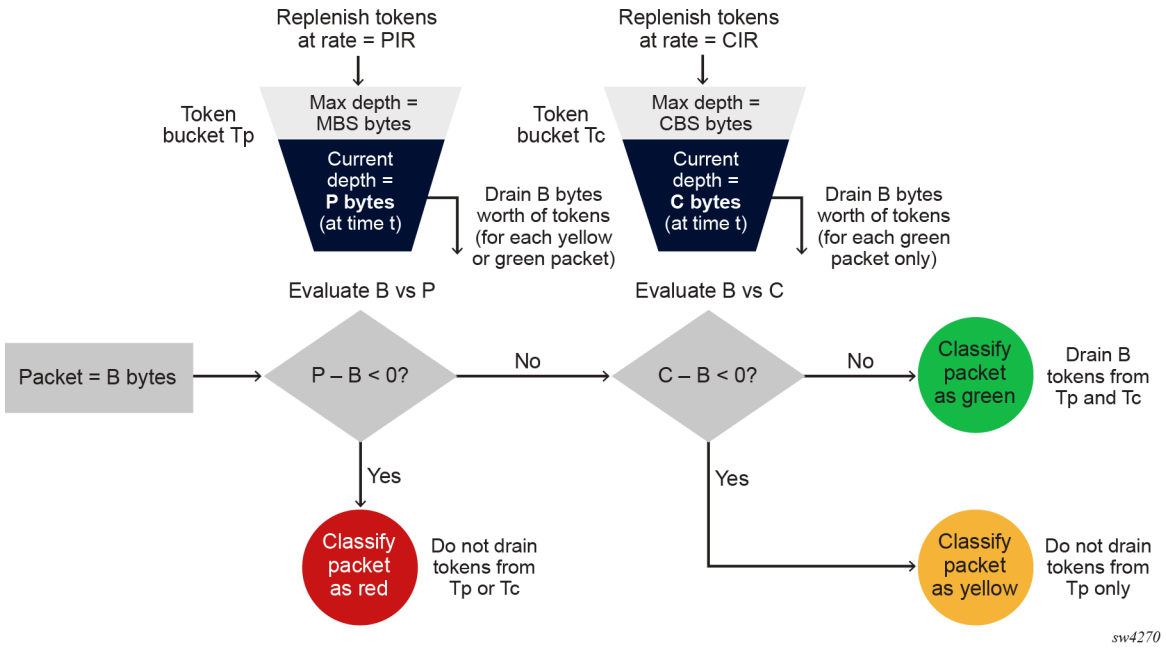
trtcm1

The **trtcm1** option refers to the policing algorithm defined in RFC 2698. With **trtcm1**, packets are marked green only after evaluation against both the CIR bucket (Tc) depth and PIR bucket depth (Tp).

Two token buckets are used, the CBS bucket and the MBS bucket. Tokens are added to the buckets based on the CIR and PIR rates. The algorithm deducts tokens from both the CBS and the MBS buckets to determine a profile for the packet.

The following diagram shows the token bucket operation for packets classified as green or uncolored at ingress.

Figure 2: RFC 2698 (trtcm1)



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.
- If the packet is precolored as yellow or if $C - B < 0$, the packet is yellow (exceeding), and *B* bytes are drained from Tp.

- Otherwise, the packet is green (conforming), and *B* bytes are drained from Tp and Tc.

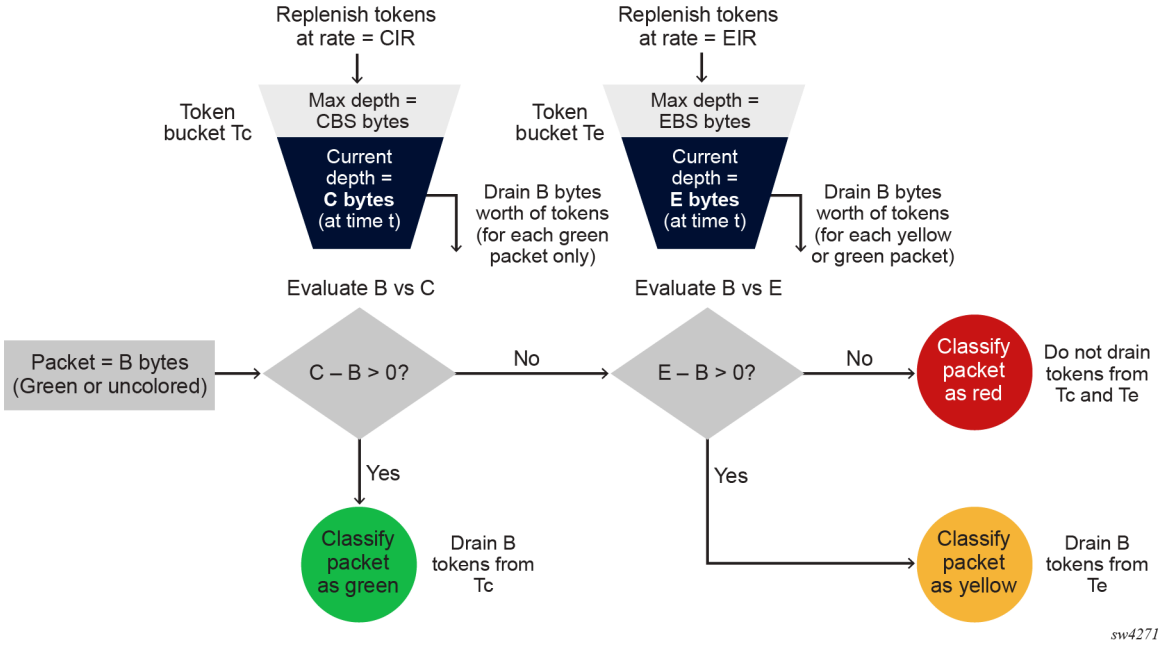
Note: A drop probability of medium or high increases the chance that the packet is discarded when it enters the egress queue, if that egress queue has a WRED slope applied.

trtcm2

The **trtcm2** option refers to the policing algorithm defined in RFC 4115. With **trtcm2**, packets that are below the CIR bucket (Tc) depth are immediately colored green, without the need to evaluate the packets against a second bucket depth. This behavior is the principal differentiator between the RFC 4115 and RFC 2698 algorithms.

Two token buckets are used, the CBS bucket and the EBS bucket. Tokens are added to the buckets based on the CIR and EIR rates.

Figure 3: RFC 2698 (trtcm2): green packets



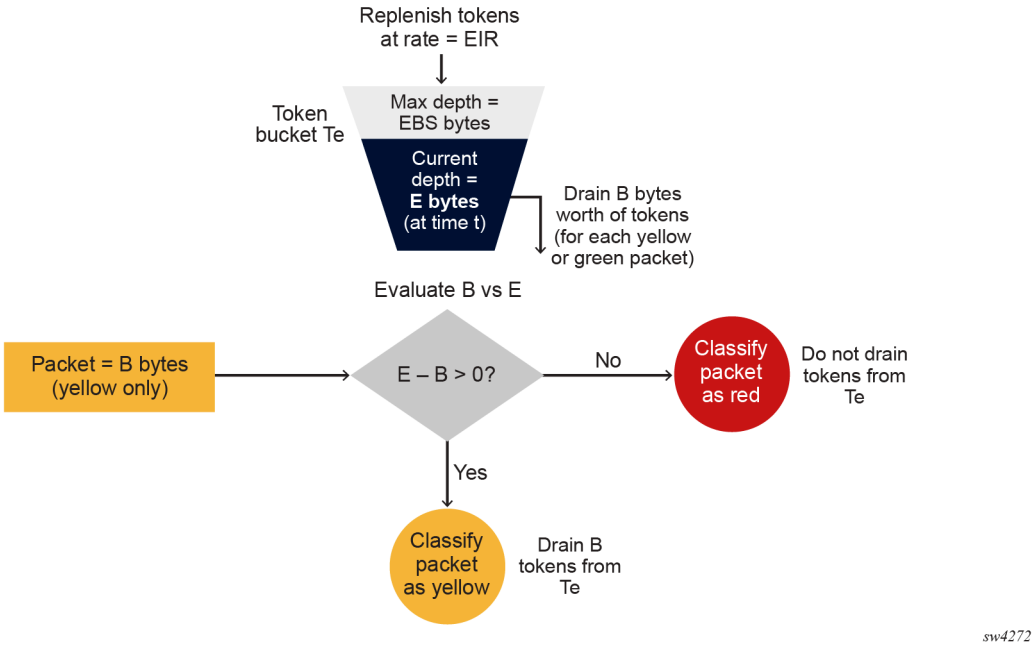
When a green or uncolored packet of size *B* bytes arrives at time *t*, the algorithm processes the packet as follows:

- If $C - B > 0$, the packet is green (conforming), and *B* bytes are drained from Tc.
- If $E - B > 0$, the packet is yellow (exceeding), and *B* bytes are drained from Te.
- Otherwise, the packet is red (violating) and no tokens are drained from Tc or Te.

The algorithm deducts tokens from either the CBS bucket (when the algorithm identifies the packet as in-profile or green) or the EBS bucket (when the algorithm identifies the packet as out-of-profile or yellow).

For yellow packets, the token bucket operation differs as shown in the following diagram. In this case, only the EBS bucket is used.

Figure 4: RFC 2698 (trtcm2): yellow packets



When a yellow packet arrives at ingress, the algorithm deducts tokens only from the EBS bucket.

When a yellow packet of size *B* bytes arrives at time *t*, the algorithm processes the packet as follows:

- If *E* – *B* > 0, the packet is yellow (exceeding), and *B* bytes are drained from Te.
- Otherwise, the packet is red (violating) and no tokens are drained from Te.

Color aware policing

By default, trTCM policing on 7730 SXR platforms operates in color-aware mode based on RFC 4115. In this mode, the trTCM algorithm processes ingress packets based on the profile that is previously applied to the packets by the input classifier. Nokia recommends to classify ingress traffic to one of the following profiles: **in-plus**, **in**, **out**, or **exceed**.

Policer profile marking

The following tables describe the color aware operation of both trTCM policers.

Table 5: trtcm1 (RFC 2698)

input-profile	CIR conform	PIR conform	output-profile	CIR bucket decrement	PIR bucket decrement
in-plus	conform	conform	in-plus	true	true
	conform	non-conform	exceed	false	false
	non-conform	conform	out	false	true
	non-conform	non-conform	exceed	false	false

input-profile	CIR conform	PIR conform	output-profile	CIR bucket decrement	PIR bucket decrement
in/in-low	conform	conform	in	true	true
	conform	non-conform	exceed	false	false
	non-conform	conform	out	false	true
	non-conform	non-conform	exceed	false	false
out/out-low	conform	conform	out	false	true
	conform	non-conform	exceed	false	false
	non-conform	conform	out	false	true
	non-conform	non-conform	exceed	false	false
exceed	conform	conform	exceed	false	true
	conform	non-conform	exceed	false	false
	non-conform	conform	exceed	false	true
	non-conform	non-conform	exceed	false	false

Table 6: trtcm2 (RFC 4115)

input-profile	CIR conform	EIR conform	output-profile	CIR bucket decrement	EIR bucket decrement
in-plus	conform	conform	in-plus	true	false
	conform	non-conform	in-plus	true	false
	non-conform	conform	out	false	true
	non-conform	non-conform	exceed	false	false
in/in-low	conform	conform	in	true	false
	conform	non-conform	in	true	true
	non-conform	conform	out	false	false
	non-conform	non-conform	exceed	false	false
out/out-low	conform	conform	out	false	true
	conform	non-conform	exceed	false	false
	non-conform	conform	out	false	true
	non-conform	non-conform	exceed	false	false
exceed	conform	conform	exceed	false	true
	conform	non-conform	exceed	false	false
	non-conform	conform	exceed	false	true

input-profile	CIR conform	EIR conform	output-profile	CIR bucket decrement	EIR bucket decrement
	non-conform	non-conform	exceed	false	false

CIR vs PIR or EIR

The CIR value must not be greater than PIR (**trtcm1**) or EIR (**trtcm2**). Otherwise, the system adjusts the CIR value to the respective PIR or EIR value. The operational values for these rates are available in state for a specified policer instance.

Note: With **trtcm2**, to obtain the full rate from the policer, the packets must be marked as **in** or **in-plus**.

CIR vs PIR bucket sizes

A packet can be non-conforming to a PIR bucket, while still conforming to a CIR bucket. This seemingly contradictory condition can occur when the CBS is quite large relative to MBS, while the difference between CIR and PIR is very small (or for example, when CIR = PIR).

Color-blind emulation

To emulate color-blind mode, the input classifier can mark all incoming packets to the same profile, for example **in**. In this case, all incoming packets have the same color and the policer operation is the same for all.

Single-rate emulation

To emulate a single-rate algorithm using **trtcm1**, set the CIR value to 0. This causes null-CIR to be set to TRUE for child policers. (A non-zero rate sets this flag to FALSE). This CIR setting is supported only for **trtcm1** policers.

9.1 Policing policies

There are four groups of policies describing policer parameters:

- Policer policies (**policer-policy**)
- CIR threshold separation policies (**cir-threshold-separation-policy**)
- PIR/EIR threshold separation policies (**pir-threshold-separation-policy**)
- Parent policer threshold policy (**parent-policer-threshold-policy**)

The CIR and PIR/EIR policies are defined based on the **algorithm-type** and **threshold-separation** parameters configured at the policer level. The policer policies and parent policer threshold policy are explicitly configured under the **qos policer-policies** context.

9.2 Policer policy

The policer policy is the basic policy defining individual policers used at subinterface level and their respective parameters. Every policer policy is identified by a unique name. Up to 64 policer policies can be defined on the system.

Each policy contains multiple policers, which are defined by algorithm type (**trtcm1** or **trtcm2**), CIR/PIR/EIR and CBS/MBS/EBS values, and a violate action for red packets.

Every policer policy instance at subinterface level is allocated 32 policers, regardless of the number of policers explicitly defined within the policer policy.

Input class map

While the policer policy defines the parameters for policers, the input class map defines the mapping of FCs to those defined policers. The input class map may not map all FCs to policers. In this case, mapping from the default input class map applies. As a consequence, the state for subinterfaces always displays statistics for all policers, even though they may not be specified explicitly.

Parent policer

The individual policers can be aggregated by a parent policer enforcing an aggregate rate. The parent policer is defined by its PIR and its inputs. The parent policer is also defined by a parent policer threshold policy defining the bucket depths and their separation between different profiles.

If a policer policy includes policers that are not declared as inputs to a parent policer, such policers are considered as orphaned and their rate is not included in the aggregate rate of the parent policer.

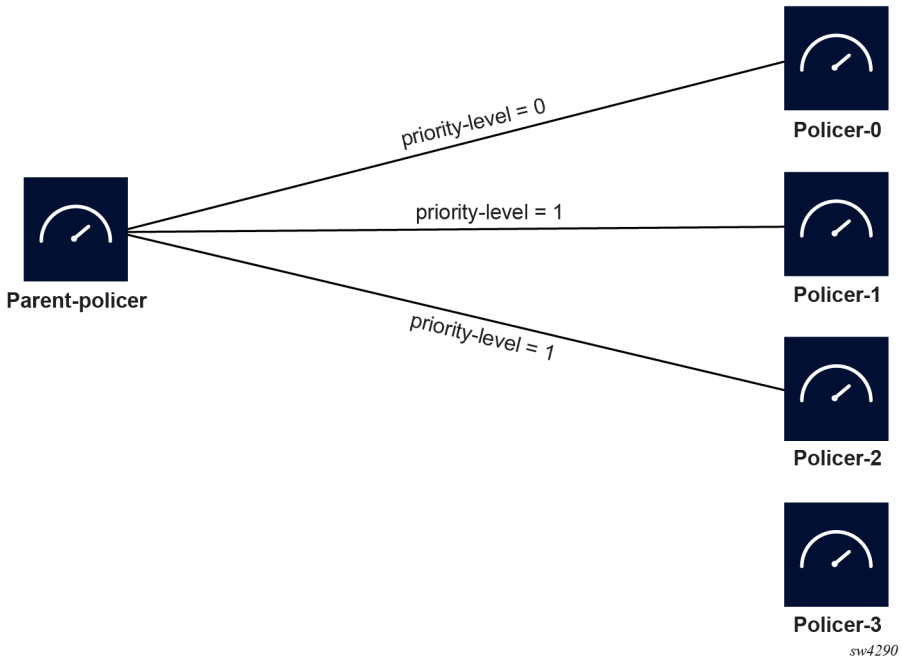
In addition, the parent policer configuration does not accept as input a policer that is not explicitly defined. Policers that are not explicitly defined in the policer policy inherit default parameters, equal to the maximum values. In other words, they do not affect traffic and they are by definition orphaned.

When the policers are parented by a parent policer, they can be assigned to one of six priority levels (0 to 5), with priority level 0 having the lowest priority. If two or more policers are assigned the same priority level, the bandwidth of the parent policer is distributed equally, provided all policers are receiving equal amounts of traffic.

Policer policy example

The following figure shows a policer policy example where the parent policer aggregates policers 0, 1, and 2, and where policer 3 is an orphaned policer.

Figure 5: Parent policer



The following output shows the configuration for the example policer policy.

```
--{ candidate shared default }--[  ]--
# info with-context qos policer-policies
    qos {
        policer-policies {
            parent-policer-threshold-policy example-parent {
                threshold-separation 18000
            }
            policer-policy example-policer {
                policer 0 {
                    algorithm-type trtcm1
                    peak-rate-kbps 1000
                    committed-rate-kbps 1000
                    maximum-burst-size 40000
                    violate-action drop
                    statistics-mode extended
                    adaptation-rules {
                        peak-rate closest
                        maximum-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 1 {
                    algorithm-type trtcm1
                    peak-rate-kbps 500
                    committed-rate-kbps 500
                    maximum-burst-size 50000
                    violate-action drop
                    statistics-mode extended
                    adaptation-rules {
                        peak-rate closest
```

```
                        maximum-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 2 {
                    algorithm-type trtcm1
                    peak-rate-kbps 250
                    committed-rate-kbps 250
                    maximum-burst-size 25000
                    violate-action drop
                    statistics-mode extended
                    adaptation-rules {
                        peak-rate closest
                        maximum-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 3 {
                    algorithm-type trtcm2
                    excess-rate-kbps 20000
                    excess-burst-size 40000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        peak-rate closest
                        committed-rate lower
                        maximum-burst-size closest
                        committed-burst-size higher
                    }
                    pir-threshold-separation {
                        inplus-separated true
                    }
                }
                parent-policer {
                    parent-policer-threshold-policy example-parent
                    rate {
                        peak-rate-kbps 1500
                        adaptation-rule closest
                        burst-allowance 40000
                    }
                    inputs {
                        policer 0 {
                            priority-level 0
                        }
                        policer 1 {
                            priority-level 1
                        }
                        policer 2 {
                            priority-level 1
                        }
                    }
                }
            }
        }
    }
```

9.3 Threshold separation policy

Policer operation is determined in part by how the policer bucket depths (PIR, EIR, and CIR) are partitioned and available for use by individual packet profiles.

On 7730 SXR platforms, the bucket depths are defined using policer threshold separation policies, defined in relation to the configured CBS, MBS, or EBS values. However, rather than enable custom definitions of policer threshold separation policies, 7730 SXR platforms support system-defined threshold separation policies, as defined in the following table.

Table 7: Default threshold separation policies

Policer bucket	Threshold separation policy name (system-reserved)	Definition
CIR	`cir-default`	Default (and only) CIR threshold separation policy
PIR (trtcm1)	`trtcm1-pir-default`	PIR policy for trtcm1 with no inplus threshold
	`trtcm1-pir-inplus`	PIR policy for trtcm1 including a threshold for the inplus profile
EIR (trtcm2)	`trtcm2-eir-default`	EIR policy for trtcm2 with no inplus threshold (the default PIR/EIR policy)
	`trtcm2-eir-inplus`	EIR policy for trtcm2 including a threshold for the inplus profile

The following sections provide additional details about these default threshold separation policies.

CIR default policy

The following table shows the settings for the `cir-default` threshold separation policy.

Table 8: CIR default policy

Input profiles	Threshold CBS value
in-plus	1.33 x CBS
in	1 x CBS
in-low	0.67 x CBS
out, out-low, exceed	0 x CBS

PIR threshold policies (trtcm1-pir-default)

The following tables describe the system-defined PIR threshold separation policies for trtcm1.

Table 9: trtcm1-pir-default

Input profiles	Threshold MBS value
in-plus, in, out	1 x MBS
in-low, out-low	0.75 x MBS
exceed	0 x MBS

Table 10: trtcm1-pir-inplus

Input profiles	Threshold MBS value
in-plus	1 x MBS
in, out	0.75 x MBS
in-low, out-low	0.5 x MBS
exceed	0 x MBS

EIR threshold policies (trtcm2)

The following tables describe the system-defined EIR trtcm2 threshold separation policies.

Table 11: trtcm2-eir-default

Input profiles	Threshold EBS value
in-plus, in, out	1 x EBS
in-low, out-low	0.75 x EBS
exceed	0 x EBS

Table 12: trtcm2-eir-inplus

Input profiles	Threshold EBS value
in-plus	1 x EBS
in, out	0.75 x EBS
in-low, out-low	0.5 x EBS
exceed	0 x EBS

Threshold separation policy settings

The selected trTCM algorithm is determined per-policer using the **algorithm-type** command: either **trtcm1** or **trtcm2**.

By default, no in-plus profile is defined for either PIR or EIR thresholds. To enable in-plus thresholds, use the **pir-threshold-separation inplus-separated** command in the following context:

qos policer-policies policer-policy policer pir-threshold-separation

Note: In this context, the keyword **pir-threshold-separation** refers either to PIR or EIR threshold separation, depending on the selected algorithm type.

State information

Information related to the operational CIR, PIR, and EIR threshold separation policies and to the operational separation threshold values per profile is available for every policer using the **info from state** command under the **qos interfaces interface** context.

Example: Display operational threshold separation policy values

```
--{ running }--[  ]--
# info with-context from state qos interfaces interface eth1.10 input policer-policies
 policer 1
    qos {
        interfaces {
            interface eth1.10 {
                input {
                    policer-policies {
                        policer 1 {
                            pir-policer-threshold-separation-policy trtcm1-pir-default
                            cir-policer-threshold-separation-policy cir-default
                            peak-rate-kbps 8000000
                            committed-rate-kbps 0
                            maximum-burst-size 180000
                            committed-burst-size 180000
                            forwarding-class af1 {
                                forwarding-type [
                                    unicast
                                ]
                            }
                            operational-separation-thresholds in {
                                pir-operational-separation-threshold 180000
                                cir-operational-separation-threshold 180000
                            }
                            operational-separation-thresholds out {
                                pir-operational-separation-threshold 180000
                                cir-operational-separation-threshold 0
                            }
                            operational-separation-thresholds exceed {
                                pir-operational-separation-threshold 0
                                cir-operational-separation-threshold 0
                            }
                            operational-separation-thresholds in-plus {
                                pir-operational-separation-threshold 180000
                                cir-operational-separation-threshold 239400
                            }
                            operational-separation-thresholds in-low {
                                pir-operational-separation-threshold 135000
                                cir-operational-separation-threshold 120600
                            }
                            operational-separation-thresholds out-low {
                                pir-operational-separation-threshold 135000
                                cir-operational-separation-threshold 0
                            }
                            policer-statistics {
                                aggregate-statistics {
                                    accepted-packets 0
                                    accepted-octets 0
                                    accepted-inplus-packets 0
```

```
                                    accepted-inplus-octets 0
                                    accepted-in-packets 0
                                    accepted-in-octets 0
                                    accepted-out-packets 0
                                    accepted-out-octets 0
                                    exceed-packets 0
                                    exceed-octets 0
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

9.4 Parent policer threshold policy

The parent policer threshold policy defines the basic parameters for the parent policer. Only one parameter is supported:

- **threshold-separation**: defines the separation (in bytes) between different priority levels of child policers. This is the amount of headroom the higher-priority level child policer gets in relation to its lower-priority peers. The ultimate MBS of the parent policer is then calculated as follows: MBS= burst-allowance + threshold-separation * 6; where the **burst-allowance** is configured under **qos policer-policies policer-policy parent-policer rate burst-allowance**. For any intermediate level child policer, the multiplication factor corresponds to its configured **priority-level**.

The operation of counters for the parent policer is as follows:

- A packet that arrives from the child policer with an **exceed** profile retains the **exceed** setting and does not consume tokens from the PIR bucket of the parent policer. The exceed counter is incremented at the child policer level instead.
- A packet that arrives from the child policer with any other profile, but that violates the PIR bucket size of the parent policer is classified with the **exceed** profile. The exceed counter is incremented at the child policer level.
- A packet that arrives from the child policer with any other profile but does not violate the PIR of the parent policer keeps the profile and it increments the corresponding counter at the child policer level.

The following shows the default **parent-policer-threshold-policy**:

```
--{ candidate shared default }--[  ]--
# info with-context from state qos policer-policies parent-policer-threshold-policy default
    qos {
        policer-policies {
            parent-policer-threshold-policy default {
                threshold-separation 18000
            }
        }
    }
```

9.5 LAG operation and statistics

If subinterfaces are created on a LAG, one policer is always instantiated per forwarding complex. The settings of the individual policer instances are set to the administrative values configured in the corresponding policer policy. The state information displays aggregate statistics (per subinterface) as well as per LAG member using the following commands:

- **info from state qos interfaces interface input policer-policies policer policer-statistics per-lag-member-statistics**
- **info from state qos interfaces interface input policer-policies policer policer-statistics aggregate-statistics**

9.6 Default policing policy

While the policer policy defines the parameters for policers, the input class map defines the mapping of FCs to those defined policers. To apply a policer to a subinterface, both a policer policy and an input class map are assigned to the subinterface. However, if a mismatch exists such that the input class map references a policer that is not explicitely defined in the corresponding policer policy, the system applies the default policer with the corresponding ID.

For every policer policy, the system always reserves a block of 32 policers. The policers that are not defined within a custom defined policer policy inherit the parameters set according to the definition in the default policer policy.

Default policer policy and input class map

The system provides a default policer policy named **default** that defines 32 policers with maximum rates. Similarly, a default input class map named **default** maps each forwarding class to a corresponding unicast policer and each broadcast/multicast/unknown-broadcast traffic to a corresponding policer per forwarding class. These default policies allow ingress statistics to be collected, as the statistics are dependent on the existence of a policer policy. Both default policies are attached to every subinterface created on the system, with no requirement for an explicit subinterface declaration under the **qos interfaces** context. Similarly, state-related information is available without an explicit subinterface declaration under the **qos interfaces** tree.

If a default policer policy is attached to a subinterface, the system allocates a single counter set per policer, but no actual policing action takes place.

When a new custom policy is created, every parameter is set to the default value.

The configuration of the default policer policy and input class map is as follows:

```
--{ running }--[  ]--
# info with-context from state qos policer-policies policer-policy default
    qos {
        policer-policies {
            policer-policy default {
                policer 0 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
```

```
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 1 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 2 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 3 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 4 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
```

```
                policer 5 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 6 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 7 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 8 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 9 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
```

```
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 10 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 11 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 12 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 13 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 14 {
                    algorithm-type trtcm2
```

```
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 15 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 16 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 17 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 18 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
```

```
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 19 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 20 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 21 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 22 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 23 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
```

```
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 24 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 25 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 26 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 27 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
```

```
                    }
                }
                policer 28 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 29 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 30 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
                policer 31 {
                    algorithm-type trtcm2
                    excess-rate-kbps 800000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated false
                    }
                }
            }
        }
    }
--{ running }--[  ]--
# info with-context from state qos input-class-map default
    qos {
```

```
        input-class-map default {
            forwarding-class fc0 {
                policers {
                    unicast-policer 0
                    multicast-policer 16
                    broadcast-policer 16
                    unknown-unicast-policer 16
                }
            }
            forwarding-class fc1 {
                policers {
                    unicast-policer 1
                    multicast-policer 17
                    broadcast-policer 17
                    unknown-unicast-policer 17
                }
            }
            forwarding-class fc10 {
                policers {
                    unicast-policer 10
                    multicast-policer 26
                    broadcast-policer 26
                    unknown-unicast-policer 26
                }
            }
            forwarding-class fc11 {
                policers {
                    unicast-policer 11
                    multicast-policer 27
                    broadcast-policer 27
                    unknown-unicast-policer 27
                }
            }
            forwarding-class fc12 {
                policers {
                    unicast-policer 12
                    multicast-policer 28
                    broadcast-policer 28
                    unknown-unicast-policer 28
                }
            }
            forwarding-class fc13 {
                policers {
                    unicast-policer 13
                    multicast-policer 29
                    broadcast-policer 29
                    unknown-unicast-policer 29
                }
            }
            forwarding-class fc14 {
                policers {
                    unicast-policer 14
                    multicast-policer 30
                    broadcast-policer 30
                    unknown-unicast-policer 30
                }
            }
            forwarding-class fc15 {
                policers {
                    unicast-policer 15
                    multicast-policer 31
                    broadcast-policer 31
                    unknown-unicast-policer 31
                }
```

```
            }
            forwarding-class fc2 {
                policers {
                    unicast-policer 2
                    multicast-policer 18
                    broadcast-policer 18
                    unknown-unicast-policer 19
                }
            }
            forwarding-class fc3 {
                policers {
                    unicast-policer 3
                    multicast-policer 19
                    broadcast-policer 19
                    unknown-unicast-policer 19
                }
            }
            forwarding-class fc4 {
                policers {
                    unicast-policer 4
                    multicast-policer 20
                    broadcast-policer 20
                    unknown-unicast-policer 20
                }
            }
            forwarding-class fc5 {
                policers {
                    unicast-policer 5
                    multicast-policer 21
                    broadcast-policer 21
                    unknown-unicast-policer 21
                }
            }
            forwarding-class fc6 {
                policers {
                    unicast-policer 6
                    multicast-policer 22
                    broadcast-policer 22
                    unknown-unicast-policer 22
                }
            }
            forwarding-class fc7 {
                policers {
                    unicast-policer 7
                    multicast-policer 23
                    broadcast-policer 23
                    unknown-unicast-policer 23
                }
            }
            forwarding-class fc8 {
                policers {
                    unicast-policer 8
                    multicast-policer 24
                    broadcast-policer 24
                    unknown-unicast-policer 24
                }
            }
            forwarding-class fc9 {
                policers {
                    unicast-policer 9
                    multicast-policer 25
                    broadcast-policer 25
                    unknown-unicast-policer 25
                }
```

```
            }
        }
    }
```

9.7 Configuring ingress policer policies

About this task

Each policer policy specifies a policer ID (from 0 to 31), algorithm type (**trtcm1** or **trtcm2**), and the following additional policer parameters:

- **adaptation-rules**: defines how individual configured values for PIR, CIR, EIR, MBS, EBS and CBS are mapped to hardware values: **closest**, **lower**, or **higher**
- **committed-burst-size**: defines the CBS in bytes.

  Note: When the **trtcm1** algorithm is enabled, the system does not force the CBS value to be below the MBS value. CBS is a burst tolerance on CIR, which means that if the offered rate is smaller than the PIR but higher than the CIR, the CBS bucket depth is consumed, even if it is nominally larger than MBS.

- **committed-rate-kbs**: defines the CIR in kilobits per second
- **excess-burst-size** (**trtcm2** only): defines the EBS in bytes
- **excess-rate-kbs** (**trtcm2** only): defines the EIR in kilobits per second
- **maximum-burst-size** (**trtcm1** only): defines the MBS in bytes
- **packet-length-adjustment**: defines what number of bytes to **add** to or **subtract** from the packet length for the policer calculation
- **peak-rate-kbs** (**trtcm1** only): defines the PIR in kilobits per second
- **pir-threshold-separation inplus-separated**: determines whether the in-plus profile is defined for either PIR or EIR thresholds
- **statistics-mode**:
  - **extended** (the only mode supported): counts forwarded packets/octets (accepted and exceeded) per profile and per policer. The discarded packets are accounted for at aggregate level per policer.

    Whenever a new policer policy is assigned to a subinterface, all counters of policer instances using the given policer policy are reset.
- **violate-action**: (applicable for either **trtcm1** or **trtcm2**): defines the action for packets that violate the PIR (or EIR) of the policer: **drop** or **mark-exceed**.

Procedure

Configure ingress policer policies using the **qos policier-policies policer-policy** command.

Example: Configure ingress policer policy

The following example shows an ingress policer policy specifying the **trctm2** algorithm, a violate action of **mark-exceed**, and in-plus **pir-threshold-separation** enabled.

```
--{ * candidate shared default }--[  ]--
# info with-context qos policer-policies policer-policy policer-policy-test
```

```
    qos {
        policer-policies {
            policer-policy policer-policy-test {
                policer 0 {
                    algorithm-type trtcm2
                    excess-rate-kbps 8000000
                    excess-burst-size 180000
                    violate-action mark-exceed
                    statistics-mode extended
                    adaptation-rules {
                        excess-rate closest
                        excess-burst-size closest
                    }
                    pir-threshold-separation {
                        inplus-separated true
                    }
                }
            }
        }
    }
```

9.8 Configuring input class maps

Procedure

Configure input class maps using the **qos input-class-map** command.

Example: Configure input class map

The following example shows an input class map that maps a forwarding class to policers for unicast, multicast, broadcast, and unknown unicast traffic.

```
--{ +* candidate shared default }--[  ]--
# info with-context qos input-class-map input-class-map-test
    qos {
        input-class-map input-class-map-test {
            forwarding-class fc-test {
                policers {
                    unicast-policer 0
                    multicast-policer 16
                    broadcast-policer 16
                    unknown-unicast-policer 16
                }
            }
        }
    }
```

9.9 Applying ingress policer policies and input class maps to a subinterface

Procedure

Apply ingress policier policies and input class maps to a subinterface using the **input-class-map** and **policer policies** commands:

Example: Apply ingress policer policies and input class maps to a subinterface

```
--{ +* candidate shared default }--[  ]--
# info with-context qos interfaces interface eth-1/1.1 input
    qos {
        interfaces {
            interface eth-1/1.1 {
                input {
                    input-class-map input-class-map-test
                    policer-policies {
                        policer-policy policer-policy-test
                    }
                }
            }
        }
    }
```

10 Egress DSCP reclassification

In many core networks, the number of different per-hop behaviors (forwarding class and profile combinations) can be limited by the need for a common denominator of per-hop behaviors among the different aggregation networks that the core network interconnects. To restore per-hop-behavior with finer granularity at the exit from the core network to an aggregation network, 7730 SXR platforms support egress DSCP reclassification.

The DSCP reclassification policy maps a DSCP value to a forwarding class and profile. The policy can be assigned to any type of subinterface (Layer 2 or Layer 3).

By default, no DSCP reclassification policy is assigned to a subinterface.

10.1 Configuring DSCP reclassification policies

Procedure

To configure egress DSCP reclassification, use the **qos classifiers dscp-reclassify-policy** command to map a DSCP value to a forwarding class and profile.

The supported values for **profile** at egress are as follows:

- **exceed**
- **in**
- **in-low**
- **in-plus**
- **out**
- **out-low**

Example: Configure DSCP reclassification policies

```
--{ + candidate shared default }--[ ]--
# info with-context qos classifiers dscp-reclassify-policy reclassify-policy-name dscp
 AF11
    qos {
        classifiers {
            dscp-reclassify-policy reclassify-policy-name {
                dscp AF11 {
                    forwarding-class fc0
                    profile out
                }
            }
        }
    }
```

10.2 Applying a DSCP reclassification policy to a subinterface

Procedure

To apply a DSCP reclassification policy to a subinterface, use the **qos interfaces interface output dscp-reclassify-policy** command.

Example: Apply a DSCP reclassification policy to a subinterface

```
--{ + candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/2.1 output dscp-reclassify-policy
    qos {
        interfaces {
            interface ethernet-1/2.1 {
                output {
                    dscp-reclassify-policy reclassify-policy-name
                }
            }
        }
    }
```

11 Egress marking and remarking: dot1p, MPLS traffic-class, and DSCP

At egress, packet marking and remarking allow 7730 SXR platforms to signal the per-hop-behavior (forwarding class and profile combinations) to the downstream node.

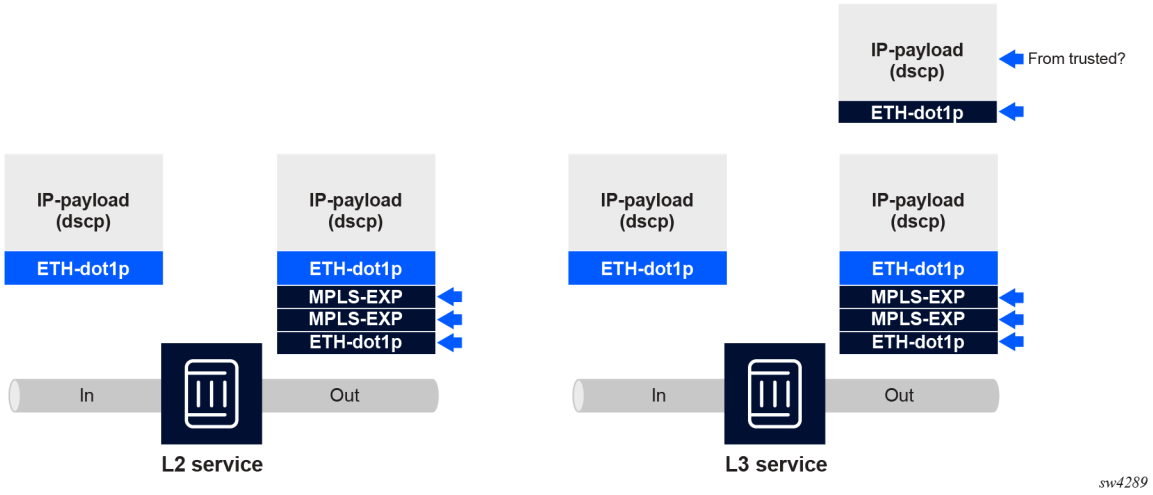
7730 SXR platforms perform egress marking and remarking based on rewrite-rule policies applied at the subinterface level. These platforms support three types of rewrite-rule policies, which are applied in the following order:

1. Dot1p policy (including inner dot1p, outer dot1p, and discard eligibility bit)
2. MPLS traffic-class policy (including optional profile settings)
3. DSCP policy (including optional profile settings)

Service edge – ingress

The following figure shows the supported packet marking capabilities at service edge ingress.

Figure 6: Service edge – ingress

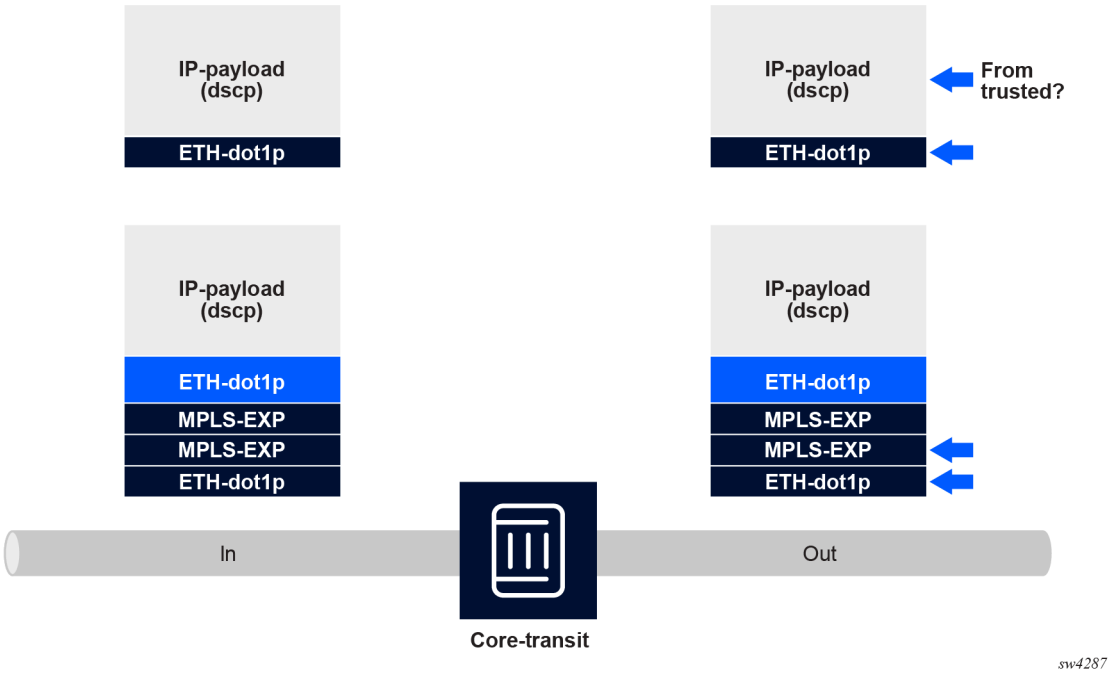


There are two options, depending on the type of service (Layer 2 or Layer 3). In both cases, the service related encapsulation is pushed on the packet and marking is indicated by arrows. Only the outer label is marked according to the MPLS traffic-class policy, and all inner labels pushed are marked with the EXP bit set to 0.

Network core – transit

The following figure shows the supported packet marking capabilities in a network core transit scenario.

Figure 7: Network core – transit



There are two types of packets to be expected:

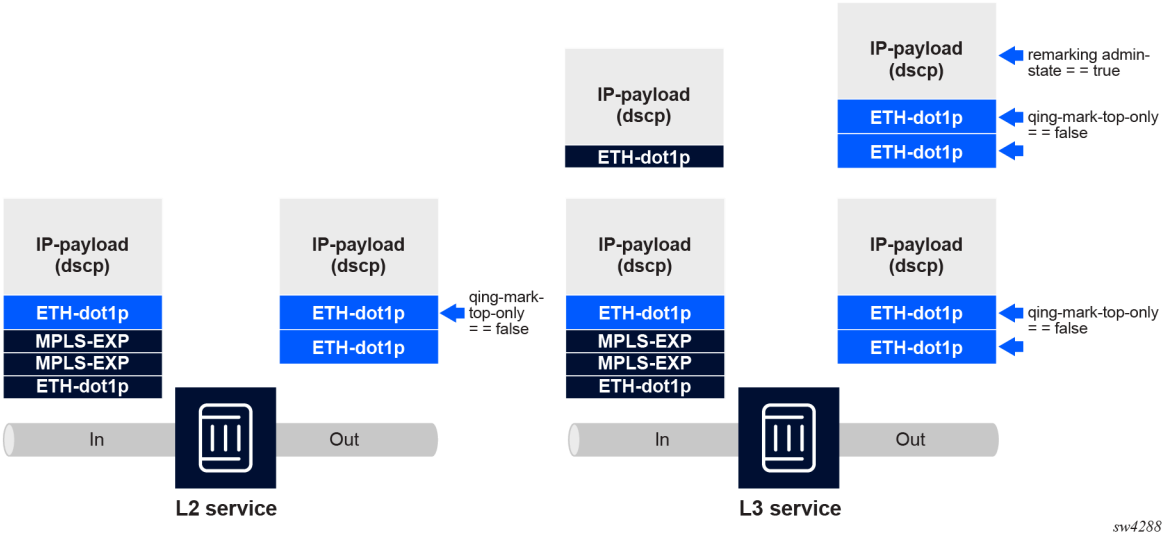
- MPLS encapsulated packets
- plain IP packets

The preceding figure indicates which encapsulation levels are subject to marking and remarking.

Service edge – egress

The following figure shows the supported packet remarking capabilities at service edge egress.

Figure 8: Service edge – egress



11.1 Dot1p rewrite-rule policy

The dot1p rewrite-rule policy maps a forwarding class and optionally profile to a dot1p value and discard-eligibility setting. Marking of inner and outer dot1p values is also supported for relevant cases. In addition, the policy can also specify marking of the discard-eligibility bit.

Dot1p marking refers to rewriting of the PCP value in the inner and outer VLAN tags. 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 16 possible internal forwarding classes to a PCP value (0 to 7) and profile.

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.

Default dot1p rewrite-rule policy

The following output shows the default dot1p rewrite-rule policy, with system-reserved name of **default**. This policy is assigned by default to any newly created subinterface.

Note: When you create custom forwarding class names, the default rewrite-rule policy is automatically updated to match the new forwarding-class naming scheme.

```
qos {
    rewrite-rules {
        dot1p-policy default {
            map fc0 {
                inner-dot1p 0
                outer-dot1p 0
```

```
                }
                map fc1 {
                    inner-dot1p 1
                    outer-dot1p 1
                }
                map fc2 {
                    inner-dot1p 2
                    outer-dot1p 2
                }
                map fc3 {
                    inner-dot1p 3
                    outer-dot1p 3
                }
                map fc4 {
                    inner-dot1p 4
                    outer-dot1p 4
                }
                map fc5 {
                    inner-dot1p 5
                    outer-dot1p 5
                }
                map fc6 {
                    inner-dot1p 6
                    outer-dot1p 6
                }
                map fc7 {
                    inner-dot1p 7
                    outer-dot1p 7
                }
                map fc8 {
                    inner-dot1p 0
                    outer-dot1p 0
                }
                map fc9 {
                    inner-dot1p 0
                    outer-dot1p 0
                }
                map fc10 {
                    inner-dot1p 0
                    outer-dot1p 0
                }
                map fc11 {
                    inner-dot1p 0
                    outer-dot1p 0
                }
                map fc12 {
                    inner-dot1p 0
                    outer-dot1p 0
                }
                map fc13 {
                    inner-dot1p 0
                    outer-dot1p 0
                }
                map fc14 {
                    inner-dot1p 0
                    outer-dot1p 0
                }
                map fc15 {
                    inner-dot1p 0
                    outer-dot1p 0
                }
            }
        }
    }
```

11.1.1 Configuring dot1p rewrite-rule policies

Procedure

To configure dot1p rewrite-rule policies, use the **qos rewrite-rules dot1p-policy** command to map a forwarding class and optionally profile values to inner dot1p, outer dot1p, and discard-eligibility settings.

Example: Configure dot1p rewrite-rule policies

```
--{ + candidate shared default }--[  ]--
# info with-context qos rewrite-rules dot1p-policy dot1p-rewrite-name
    qos {
        rewrite-rules {
            dot1p-policy dot1p-rewrite-name {
                map fc0 {
                    inner-dot1p 0
                    outer-dot1p 0
                    inner-de true
                    outer-de false
                    profile in {
                    }
                }
            }
        }
    }
```

11.1.2 Applying a dot1p rewrite-rule policy to a subinterface

Procedure

To apply a dot1p rewrite-rule policy to a subinterface, use the **qos interfaces interface output rewrite-rules** command.

Example: Apply a dot1p rewrite-rule policy to a subinterface

In this example, the **qinq-rewrite-outer-only** parameter is set to **true**, which restricts remarking to only the outer dot1p bits on the subinterface.

```
--{ + candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/2.1
    qos {
        interfaces {
            interface ethernet-1/2.1 {
                interface-ref {
                    interface ethernet-1/2
                    subinterface 1
                }
                output {
                    rewrite-rules {
                        dot1p-policy dot1p-rewrite-name
                        qinq-rewrite-outer-only true
                    }
                }
            }
        }
    }
```

11.2 MPLS traffic-class rewrite-rule policy

The MPLS traffic-class rewrite-rule policy maps a packet's forwarding class and optionally profile settings to a traffic class (EXP) value. The policy applies in the following scenarios:

- Service edge ingress scenario: the marking is applied only for the outer label. All inner labels are marked with the EXP bit set to 0.
- Core transit scenario: only the outer label EXP bits are remarked.

The policy can specify one traffic class that applies for all packets matching an FC, or it can map multiple traffic classes to one or more profiles associated with an FC.

Default MPLS traffic-class rewrite-rule policy

The following output shows the default MPLS traffic-class rewrite-rule policy, with system-reserved name of **default**. This policy is assigned by default to any newly created subinterface.

Note: When you create custom forwarding class names, the default rewrite-rule policy is automatically updated to match the new forwarding class naming scheme.

```
qos {
    rewrite-rules {
        mpls-traffic-class-policy default {
            map fc0 {
                traffic-class 0
            }
            map fc1 {
                traffic-class 1
            }
            map fc2 {
                traffic-class 2
                profile in {
                    traffic-class 3
                }
                profile in-plus {
                    traffic-class 3
                }
            }
            map fc3 {
                traffic-class 2
                profile in {
                    traffic-class 3
                }
                profile in-plus {
                    traffic-class 3
                }
            }
            map fc4 {
                traffic-class 4
            }
            map fc5 {
                traffic-class 5
            }
            map fc6 {
                traffic-class 6
            }
            map fc7 {
                traffic-class 7
            }
```

```
            map fc8 {
                traffic-class 0
            }
            map fc9 {
                traffic-class 0
            }
            map fc10 {
                traffic-class 0
            }
            map fc11 {
                traffic-class 0
            }
            map fc12 {
                traffic-class 0
            }
            map fc13 {
                traffic-class 0
            }
            map fc14 {
                traffic-class 0
            }
            map fc15 {
                traffic-class 0
            }
        }
    }
}
```

11.2.1 Configuring MPLS traffic-class rewrite-rule policies

Procedure

To configure MPLS traffic-class rewrite-rule policies, use the **qos rewrite-rules mpls-traffic-class-policy** command to map forwarding classes, and optionally profiles, to an MPLS traffic class.

Example: Configure MPLS traffic-class rewrite-rule policies

```
--{ candidate shared default }--[  ]--
# info with-context qos rewrite-rules mpls-traffic-class-policy mpls-traffic-class-policy-
name
    qos {
        rewrite-rules {
            mpls-traffic-class-policy mpls-traffic-class-policy-name {
                map fc0 {
                    traffic-class 1
                    profile in-plus {
                        traffic-class 1
                    }
                }
            }
        }
    }
```

11.2.2 Applying an MPLS traffic-class rewrite-rule policy to a subinterface

Procedure

To apply an MPLS traffic-class rewrite-rule policy to a subinterface, use the **qos interfaces interface output rewrite-rules** command.

Example: Apply an MPLS traffic-class rewrite-rule policy to a subinterface (7730 SXR)

```
--{ + candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-1/2.1 output rewrite-rules mpls-
traffic-class-policy
    qos {
        interfaces {
            interface ethernet-1/2.1 {
                interface-ref {
                    interface ethernet-1/2
                    subinterface 1
                }
                output {
                    rewrite-rules {
                        mpls-traffic-class-policy mpls-traffic-class-policy-name
                    }
                }
            }
        }
    }
```

11.3 DSCP rewrite-rule policy

The DSCP rewrite-rule policy maps a forwarding class and optionally profile to a DSCP value.

When a DSCP rewrite-rule policy is applied to a subinterface, the policy attempts to match the forwarding class (and optionally the profile) 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.

To enable changes to the DSCP value in the original IP packet (considered true remarking), you must perform the following:

- Assign a DSCP rewrite-rule policy to the subinterface, using the following command:

  qos interfaces interface <*name*> **output rewrite-rules dscp-policy** <*name*>

- If the input subinterface is trusted (the default setting), enable the **force-rewrite-trusted** option using the following command:

  qos interfaces interface <*name*> **output rewrite-rules dscp-rewrite force-rewrite-trusted**

  If the input subinterface is untrusted, **force-rewrite-trusted** has no effect, and remarking is applied as defined by the **rewrite-rules dscp-policy** command.

The following table describes the rewrite-rule behavior based on the policy configuration and subinterface flag.

Table 13: DSCP rewrite-rule action based on policy configuration and subinterface flag

Trusted at input	force-rewrite-trusted setting at output	DSCP action
false	false	remark
true	false	preserve
false	true	remark
true	true	remark

Default DSCP rewrite-rule policy

The following output shows the default DSCP rewrite-rule policy, with system-reserved name of **default**. This default policy is assigned to any subinterface at creation time.

Note: When you create custom forwarding class names, the default rewrite-rule policy is automatically updated to match the new forwarding-class naming scheme.

```
qos {
    rewrite-rules {
        dscp-policy default {
            map fc0 {
                dscp LE
            }
            map fc1 {
                dscp CS1
            }
            map fc2 {
                dscp AF12
                profile in {
                    dscp AF11
                }
                profile in-plus {
                    dscp AF11
                }
            }
            map fc3 {
                dscp AF22
                profile in {
                    dscp AF21
                }
                profile in-plus {
                    dscp AF21
                }
            }
            map fc4 {
                dscp AF42
                profile in {
                    dscp AF41
                }
                profile in-plus {
                    dscp AF41
                }
            }
            map fc5 {
                dscp EF
            }
```

```
                map fc6 {
                    dscp CS6
                }
                map fc7 {
                    dscp CS7
                }
                map fc8 {
                    dscp LE
                }
                map fc9 {
                    dscp LE
                }
                map fc10 {
                    dscp LE
                }
                map fc11 {
                    dscp LE
                }
                map fc12 {
                    dscp LE
                }
                map fc13 {
                    dscp LE
                }
                map fc14 {
                    dscp LE
                }
                map fc15 {
                    dscp LE
                }
            }
        }
    }
```

11.3.1 Configuring DSCP rewrite-rule policies

Procedure

To configure DSCP rewrite-rule policies, use the **qos rewrite-rules dscp-policy** command to map forwarding classes, and optionally profiles, to a DSCP value.

Example: Configure DSCP rewrite-rule policies

```
--{ + candidate shared default }--[  ]--
# info with-context qos rewrite-rules dscp-policy dscp-policy-name
    qos {
        rewrite-rules {
            dscp-policy dscp-policy-name {
                map fc0 {
                    dscp AF11
                    profile out {
                        dscp AF11
                    }
                }
            }
        }
    }
```

11.3.2 Applying a DSCP rewrite-rule policy to a subinterface

Procedure

To apply a DSCP rewrite-rule policy to a subinterface, use the **qos interfaces interface output rewrite-rules** command.

When the input subinterface is trusted (the default setting), remarking is only applied if the **output rewrite-rules dscp-rewrite force-rewrite-trusted** parameter is set to **true** on the output subinterface. When the input subinterface is untrusted, remarking is applied regardless of the **force-rewrite-trusted** setting. (To configure an input subinterface as trusted or untrusted, use the **qos interfaces interface input classifiers tos-rewrite-state** command.)

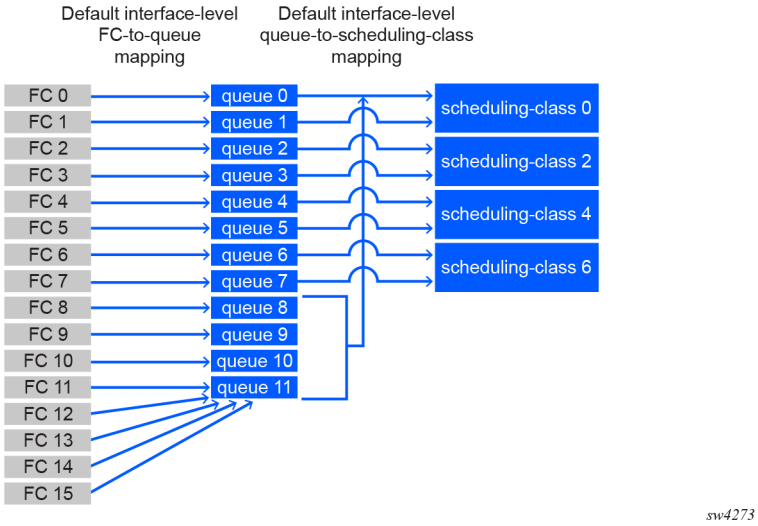
Example: Apply a DSCP rewrite-rule policy to a subinterface

```
--{ + candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/2.1
    qos {
        interfaces {
            interface ethernet-1/2.1 {
                interface-ref {
                    interface ethernet-1/2
                    subinterface 1
                }
                output {
                    rewrite-rules {
                        dscp-policy dscp-policy-name
                        dscp-rewrite {
                            force-rewrite-trusted true
                        }
                    }
                }
            }
        }
    }
```

12 Egress queue and scheduling class mapping and scheduling

By default on 7730 SXR platforms, each interface is associated with 12 configurable egress queues. Every interface always has a full set of egress queues; only the names of the queues are variable. By default, these queues also map to a set of scheduling classes, as shown in the following figure.

Figure 9: Default FC to queue mapping



On an interface, packets are assigned to egress queues based on FC-to-queue mapping. To change the default mapping on interfaces, configure a custom FC-to-queue mapping using the **qos forwarding-classes** command.

On a subinterface, by default all traffic uses the same interface-level queues as shown in the preceding figure. To associate queues and forwarding classes to a subinterface, configure an output class map using the **qos output-class-map** command and assign it to the subinterface.

Egress schedulers

To manage the distribution of scheduling resources among queues and scheduling classes, the traffic management unit (TMU) uses egress schedulers. Two configurable egress scheduling policies are supported to manage the queue and scheduling class schedulers:

- Queue scheduling policy (at subinterface [CVLAN] and interface-queue level)
  The queue scheduling policy also defines the queue to scheduling class mapping.
- Scheduling class scheduling policy (at interface level)

The chapters that follow provide more detail about egress queue mapping and egress scheduling.

13 Egress queue mapping

Treatment of egress packets is governed by the mapping of forwarding classes to egress queues. Two egress queue mappings are supported, at interface-level and at subinterface-level:

- The interface-level forwarding class to queue mapping is configured using the following command:

  qos forwarding-classes forwarding-class <*name*> **output queue** <*value*>

- The subinterface-level forwarding class to queue mapping is configured via an output class map using the following command:

  qos output-class-map forwarding-class <*name*> **queue name** <*value*>

Packets tagged with a forwarding class at egress of the interface or subinterface are forwarded to the associated queue.

No default queues are created at the subinterface level. As a result, all subinterface traffic by default uses interface-level queues. Subinterface level queues are created only after an output class map is associated with a subinterface.

13.1 Interface-level queues

On 7730 SXR platforms, 12 egress queues (default name: **queue-0** to **queue-11**) are always available on every interface.

Default interface queues

The following table shows the default settings applied to the interface-level queues.

Table 14: Default interface queues

default-queue-name	queue-index	committed-burst-size	maximum-burst-size	peak-rate-percent	weight	scheduling-class
queue-0	0	default	max	100	1	0
queue-1	1	default	max	100	1	0
queue-2	2	default	max	100	1	2
queue-3	3	default	max	100	1	2
queue-4	4	default	max	100	1	4
queue-5	5	default	max	100	1	4
queue-6	6	default	max	100	1	6
queue-7	7	default	max	100	1	6
queue-8	8	default	max	100	1	0

default-queue-name	queue-index	committed-burst-size	maximum-burst-size	peak-rate-percent	weight	scheduling-class
queue-9	9	default	max	100	1	0
queue-10	10	default	max	100	1	0
queue-11	11	default	max	100	1	0

Interface queue settings include the following parameters:

- **committed-burst-size**: defines the guaranteed portion of the queue length based on the global **qos buffer-management committed-burst-size-table** configuration.
- **maximum-burst-size**: defines the maximum queue length. A value of **max** corresponds to 10 ms buffering capacity at the peak rate.
- **peak-rate-percent**: defines the peak queue rate as a percentage of the interface rate.

Custom queue names

If the default queue names are updated to custom-defined queue names, each custom queue inherits the default values originally associated with its **queue-index** value.

Related topics

Configuring named queues

13.2 Subinterface-level queues

No default queues are created at the subinterface level. As a result, the default interface-level queue settings as described in the preceding section are applied to the subinterface-level traffic by default. To create custom subinterface-level queues, define an output class map and associate it with a subinterface. The output class map defines only the forwarding class to queue mapping (and is typically based on custom-defined forwarding class and queue names). No default output class map is provided.

Queue attributes are defined separately using a buffer allocation profile, while all queue scheduling is defined using egress scheduling parameters for the subinterface.

Related topics

Buffer allocation profile

Egress scheduling

13.3 Egress queue mapping configuration

To configure egress queue mapping, perform the following high-level steps:

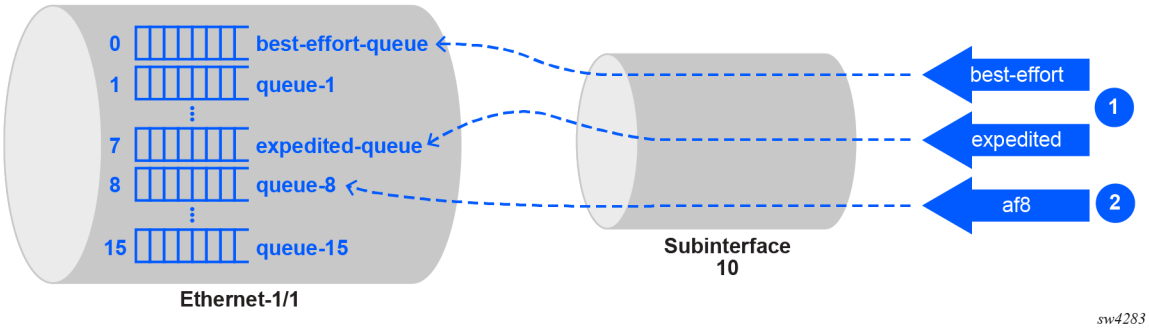
1. Configure the queue names and indexes.
2. Configure the forwarding class names, index values, and interface-level FC to queue mapping.
3. Configure an output class map to define the subinterface-level FC to queue mapping.
4. Apply the queue settings to the interface QoS configuration.
5. Apply the output class map to the subinterface QoS configuration.

Note: For the queue mapping to function properly, a classifier (DSCP, dot1p, or MPLS) must also be assigned to the ingress interfaces to classify incoming packets with an FC value.

13.4 Example 1: Mapping without output class map

By default, no output class map exists on a subinterface. The following figure shows such an example, in which the subinterface has no output class map associated. As a result, no queue mapping occurs on the subinterface and the queues are mapped using the interface-level mapping instead.

Figure 10: Egress queue mapping example without an output class map



Based on input classification, packets can arrive at egress tagged with one of the following three forwarding-classes:

1. **best-effort** or **expedited** as defined by the DSCP policy
2. **af8** as defined by the **default forwarding-class** setting defined at subinterface level

Because no output class map exists, all forwarding classes are directed to interface-level queues.

The following examples show the configurations applied in this example.

Example: Queue configuration (example 1)

```
--{ + candidate shared default }--[ ]--
# info with-context qos queues
    qos {
        queues {
            queue best-effort-queue {
                queue-index 0
            }
            queue expedited-queue {
                queue-index 7
            }
            queue queue-8 {
                queue-index 8
            }
        }
    }
```

Example: Forwarding class configuration (example 1)

```
--{ + candidate shared default }--[ ]--
# info with-context qos forwarding-classes
```

```
    qos {
        forwarding-classes {
            forwarding-class af8 {
                forwarding-class-index 8
                output {
                    queue queue-8
                }
            }
            forwarding-class best-effort {
                forwarding-class-index 0
                output {
                    queue best-effort-queue
                }
            }
            forwarding-class expedited {
                forwarding-class-index 7
                output {
                    queue expedited-queue
                }
            }
        }
    }
```

Example: DSCP classifiers configuration (example 1)

```
--{ + candidate shared default }--[ ]--
# info with-context qos classifiers
    qos {
        classifiers {
            dscp-policy example-dscp-policy {
                dscp 0 {
                    forwarding-class best-effort
                    profile out
                }
                dscp 50 {
                    forwarding-class expedited
                    profile in-plus
                }
            }
        }
    }
```

Example: QoS interface configuration (example 1)

```
--{ + candidate shared default }--[ ]--
# info with-context qos interfaces interface ethernet-1/1
    qos {
        interfaces {
            interface eth-1/1 {
                interface-ref {
                    interface ethernet-1/1
                }
                output {
                    queues {
                        queue best-effort-queue {
                        }
                        queue expedited-queue {
                        }
                    }
                }
            }
        }
```

```
    }
```

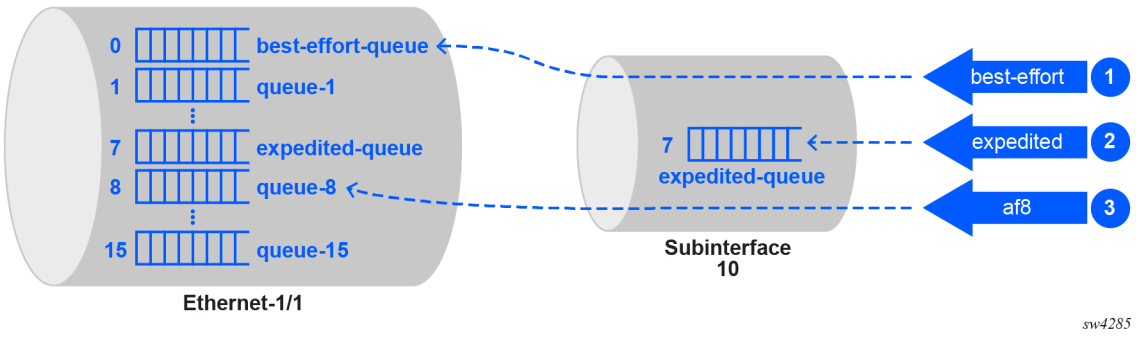
Example: QoS subinterface configuration (example 1)

```
--{ + candidate shared default }--[ ]--
# info with-context qos interfaces interface eth-1/1.10
    qos {
        interfaces {
            interface eth-1/1.10 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 10
                }
                input {
                    classifiers {
                        dscp-policy example-dscp-policy
                        default {
                            forwarding-class af8
                        }
                    }
                }
            }
        }
    }
```

13.5 Example 2: Mapping with output class map

In the following example, a subinterface has an associated output class map that redirects forwarding class **best-effort** to an interface-level queue and maps forwarding class **expedited** to a local subinterface queue.

Figure 11: Egress queue mapping example with output class map



The example shows three possibilities for traffic flows. The output class map defines the following mappings:

1. Forwarding class **best-effort** is explicitly redirected to the interface level queue (using the **re-direct-to remote** parameter).
2. Forwarding class **expedited** maps to local subinterface queue **expedited-queue**.
3. Forwarding class **af8** has no explicit output queue mapping; therefore, matching traffic is redirected to the interface level queue.

Note: As a best practice, always define an explicit output queue mapping for the forwarding class. This item is included in the example only to show the behavior when the output queue mapping is omitted.

The following outputs show the configurations applied in this example.

Example: Queue configuration (example 2)

```
--{ + candidate shared default }--[  ]--
# info with-context qos queues
    qos {
        queues {
            queue best-effort-queue {
                queue-index 0
            }
            queue expedited-queue {
                queue-index 7
            }
            queue queue-8 {
                queue-index 8
            }
        }
    }
```

Example: Forwarding class configuration (example 2)

```
--{ + candidate shared default }--[  ]--
# info with-context qos forwarding-classes
    qos {
        forwarding-classes {
            forwarding-class af8 {
                forwarding-class-index 8
                output {
                    queue queue-8
                }
            }
            forwarding-class best-effort {
                forwarding-class-index 0
                output {
                    queue best-effort-queue
                }
            }
            forwarding-class expedited {
                forwarding-class-index 7
                output {
                    queue expedited-queue
                }
            }
        }
    }
```

Example: Output class map configuration (example 2)

```
--{ + candidate shared default }--[  ]--
# info with-context qos output-class-map example-output-class-map
    qos {
        output-class-map example-output-class-map {
            forwarding-class best-effort {
                queue {
```

```
                    name best-effort-queue
                    re-direct-to remote
                }
            }
        }
    }
```

Example: DSCP classifiers configuration (example 2)

```
--{ + candidate shared default }--[  ]--
# info with-context qos classifiers
    qos {
        classifiers {
            dscp-policy example2-dscp-policy {
                dscp 0 {
                    forwarding-class best-effort
                    profile out
                }
                dscp 50 {
                    forwarding-class expedited
                    profile in-plus
                }
            }
        }
    }
```

Example: QoS interface configuration (example 2)

```
--{ + candidate shared default }--[  ]--
# info with-context qos interfaces interface eth-1/1
    qos {
        interfaces {
            interface eth-1/1 {
                interface-ref {
                    interface ethernet-1/1
                }
                output {
                    queues {
                        queue best-effort-queue {
                        }
                        queue expedited-queue {
                        }
                    }
                }
            }
        }
    }
```

Note: The configuration above is shown for illustrative purposes only. Given that interfaces inherit the queue configurations by default, the queues do not need to be explicitly associated with the interface.

Example: QoS subinterface configuration (example 2)

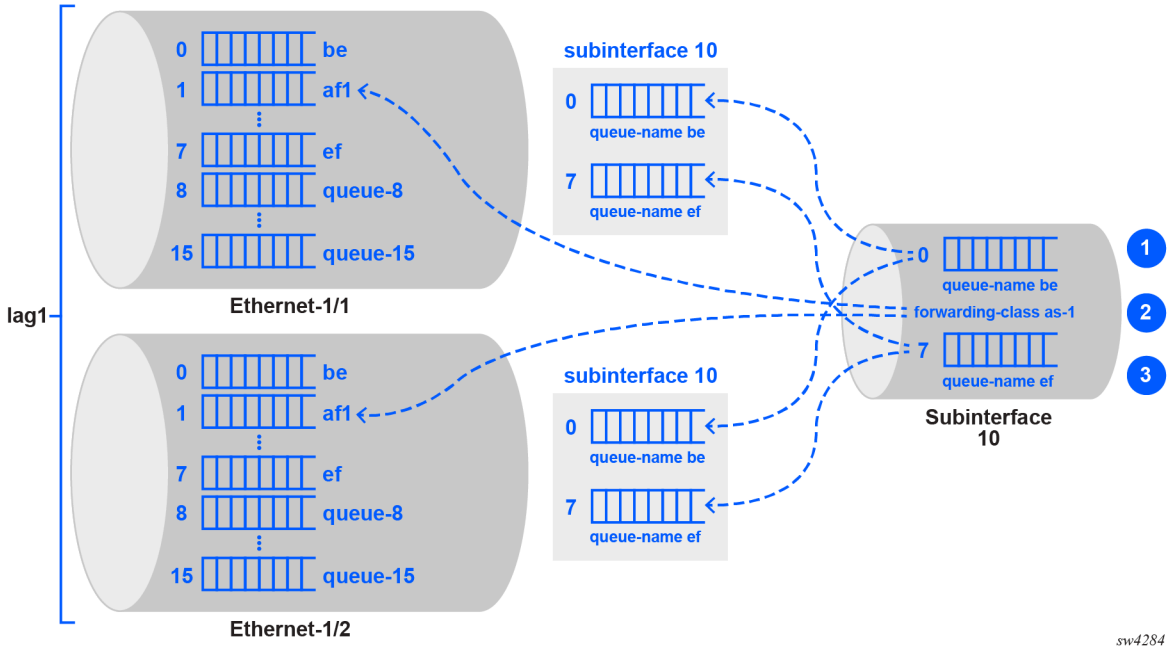
```
--{ + candidate shared default }--[  ]--
# info with-context qos interfaces interface eth-1/1.10
    qos {
        interfaces {
            interface eth-1/1.10 {
                interface-ref {
```

```
                    interface ethernet-1/1
                    subinterface 10
                }
                input {
                    classifiers {
                        dscp-policy example2-dscp-policy
                        default {
                            forwarding-class af8
                        }
                    }
                }
                output {
                    output-class-map example-output-class-map
                }
            }
        }
    }
```

13.6 Example 3: Mapping with output class map on a LAG

In the following example, a LAG subinterface has an associated output class map that redirects forwarding class **af-1** to an interface-level queue, and maps forwarding classes **0** and **7** to local subinterface queues.

Figure 12: Egress queue mapping example with LAG



The example shows three possibilities for traffic flows. The output class map defines the following mappings:

1. Forwarding class **0** (**best-effort**) maps to local subinterface queue **be**.
2. Forwarding class **af-1** is explicitly redirected to the interface level queue (using the **re-direct-to remote** parameter).

3. Forwarding class **7** (**expedited**) maps to local subinterface queue **ef**.

Forwarding class **af-1** on the LAG subinterface maps to queue **af1** on each of the LAG member interfaces. And queue names **be** and **ef** on LAG subinterface 10 similarly map to each interface's equivalent queues (**be** and **ef**) on each member interface's subinterface 10.

The following examples show the configurations applied in this example.

Example: Queue configuration (example 3)

```
--{ + candidate shared default }--[  ]--
# info with-context qos queues
    qos {
        queues {
            queue af1 {
                queue-index 1
            }
            queue be {
                queue-index 0
            }
            queue ef {
                queue-index 7
            }
        }
    }
```

Example: Forwarding class configuration (example 3)

```
--{ + candidate shared default }--[  ]--
# info with-context qos forwarding-classes
    qos {
        forwarding-classes {
            forwarding-class af-1 {
                forwarding-class-index 1
                output {
                    queue af1
                }
            }
            forwarding-class best-effort {
                forwarding-class-index 0
                output {
                    queue be
                }
            }
            forwarding-class expedited {
                forwarding-class-index 7
                output {
                    queue ef
                }
            }
        }
    }
```

Example: Output class map configuration (example 3)

```
--{ + candidate shared default }--[  ]--
# info with-context qos output-class-map example3-output-class-map
    qos {
        output-class-map example3-output-class-map {
            forwarding-class af-1 {
                queue {
                    name af1
```

```
                    re-direct-to remote
                }
            }
            forwarding-class best-effort {
                queue {
                    name be
                }
            }
            forwarding-class expedited {
                queue {
                    name ef
                }
            }
        }
```

Example: DSCP classifiers configuration (example 3)

```
--{ + candidate shared default }--[  ]--
# info with-context qos classifiers dscp-policy example3-dscp-policy
    qos {
        classifiers {
            dscp-policy example3-dscp-policy {
                dscp 0 {
                    forwarding-class best-effort
                    profile out
                }
                dscp 20 {
                    forwarding-class af-1
                    profile in
                }
               dscp 50 {
                    forwarding-class expedited
                    profile in-plus
                }
            }
        }
    }
```

Example: LAG configuration (example 3)

```
--{ + candidate shared default }--[  ]--
# info with-context interface lag1
    interface lag1 {
        admin-state enable
        vlan-tagging true
        subinterface 10 {
            vlan {
                encap {
                    single-tagged {
                        vlan-id 10
                    }
                }
            }
        }
        lag {
            lag-type static
            member-speed 100G
        }
    }
--{ + candidate shared default }--[  ]--
# info with-context interface ethernet-1/1
    interface ethernet-1/1 {
```

```
        admin-state enable
        ethernet {
            aggregate-id lag1
            port-speed 100G
        }
    }
--{ + candidate shared default }--[  ]--
# info with-context interface ethernet-1/2
    interface ethernet-1/2 {
        admin-state enable
        ethernet {
            aggregate-id lag1
            port-speed 100G
        }
    }
```

Example: QoS LAG subinterface configuration (example 3)

```
--{ + candidate shared default }--[  ]--
# info with-context qos interfaces interface lag1-10
    qos {
        interfaces {
            interface lag1-10 {
                interface-ref {
                    interface lag1
                    subinterface 10
                }
                output {
                    output-class-map example3-output-class-map
                }
            }
        }
    }
```

13.7 LAG queue statistics

For interface-level queues on interfaces that are members of a LAG, every LAG member (interface) has a separate set of 12 queues and the queue statistics per interface are available in state.

For subinterface queues, the output class map is assigned to the LAG interface ID. However, the queues are instantiated per interface or per subinterface (as shown in the preceding example), while the statistics for these queues are maintained per queue and per interface.

13.7.1 Displaying LAG queue statistics

Procedure

To display the aggregate LAG statistics in this example, use the following command:

info from state qos interfaces interface lag1-10 output queues queue be queue-statistics aggregate-statistics

To display the per-LAG member statistics in this example, use the following command:

info from state qos interfaces interface lag1-10 output queues queue be queue-statistics per-lag-member-statistics

13.8 Interface and subinterface queue statistics

For every interface or subinterface-level queue, the system collects the following statistics:

- transmitted-inplus-packets
- transmitted-inplus-octets
- dropped-inplus-packets
- dropped-inplus-octets
- transmitted-in-packets
- transmitted-in-octets
- dropped-in-packets
- dropped-in-octets
- transmitted-out-packets
- transmitted-out-octets
- dropped-out-packets
- dropped-out-octets
- transmitted-exceed-packets
- transmitted-exceed-octets
- dropped-exceed-packets
- dropped-exceed-octets

At subinterface level, the queue statistics as well as the other queue state are maintained only for the local queues.

13.8.1 Displaying queue statistics

Procedure

To display queue statistics at interface and subinterface levels, use the following command:

info from state qos interfaces interface <*name*> **output queues queue * queue-statistics**

The command output displays statistics for dropped and transmitted traffic by profile.

Example: Display queue statistics

```
--{ candidate shared default }--[  ]--
# info with-context from state qos interfaces interface eth1.10 output queues queue be-
queue
    qos {
        interfaces {
            interface eth1.10 {
                output {
                    queues {
                        queue be-queue {
                            queue-type local
                            forwarding-class [
                                be
```

```
                        ]
                        queue-statistics {
                            aggregate-statistics {
                                in-profile {
                                    transmitted-packets 0
                                    transmitted-octets 0
                                    dropped-packets 0
                                    dropped-octets 0
                                }
                                in-plus-profile {
                                    transmitted-packets 0
                                    transmitted-octets 0
                                    dropped-packets 0
                                    dropped-octets 0
                                }
                                out-profile {
                                    transmitted-packets 0
                                    transmitted-octets 0
                                    dropped-packets 0
                                    dropped-octets 0
                                }
                                exceed-profile {
                                    transmitted-packets 0
                                    transmitted-octets 0
                                    dropped-packets 0
                                    dropped-octets 0
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

13.8.2 Clearing queue statistics

Procedure

To clear queue statistics for an individual queue, use the following **tools** command:

tools qos interfaces interface <*name*> **output queues queue** <*name*> **queue-statistics clear**

To clear queue statistics at interface and subinterface level for all queues at once, use the following **tools** command:

tools qos interfaces interface <*name*> **output queues clear-statistics**

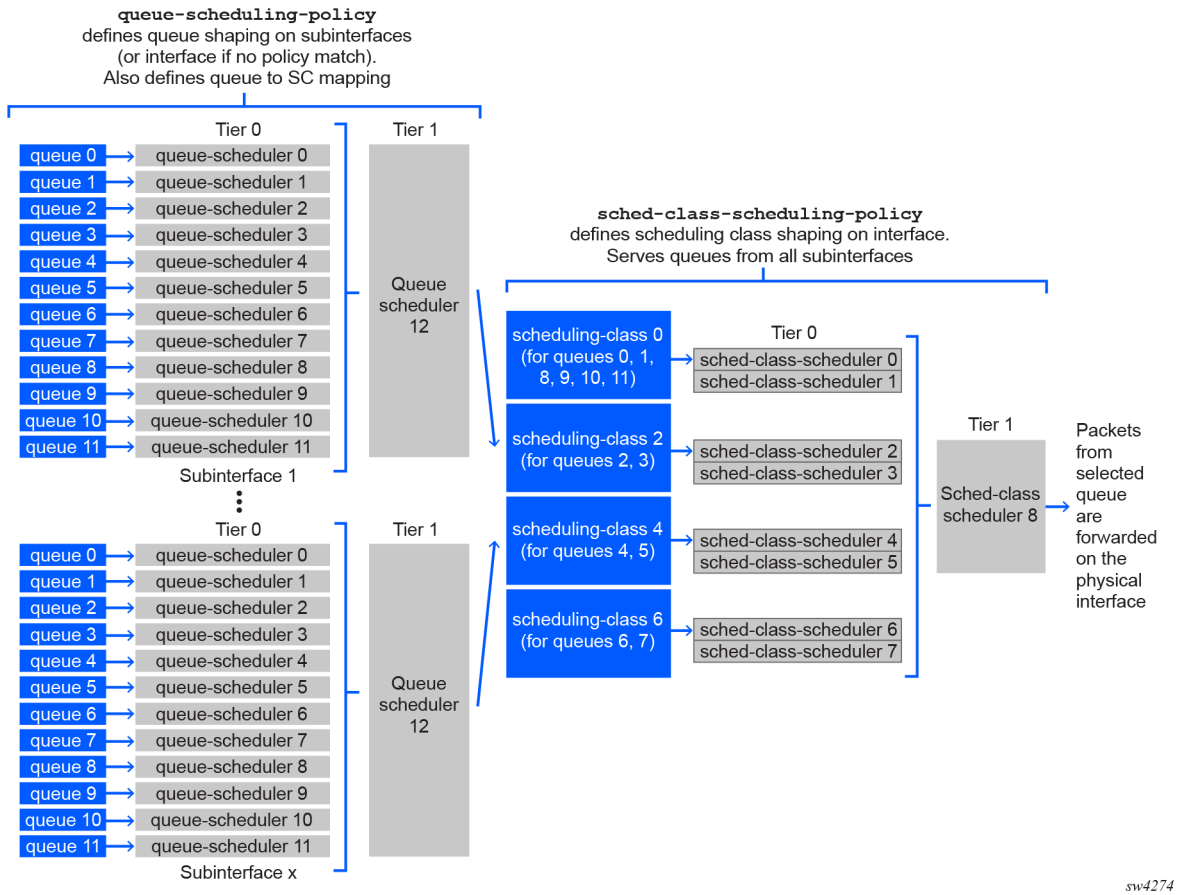
14 Egress scheduling

On 7730 SXR platforms, the traffic management unit (TMU) uses egress schedulers to manage the distribution of scheduling resources available to process egress traffic. Two configurable egress scheduling policies are supported:

- Queue scheduling policy (at subinterface [CVLAN] and interface-queue level)
- Scheduling class scheduling policy (at interface level)

The following diagram shows the flow of traffic from queues through example queue schedulers and scheduling class schedulers.

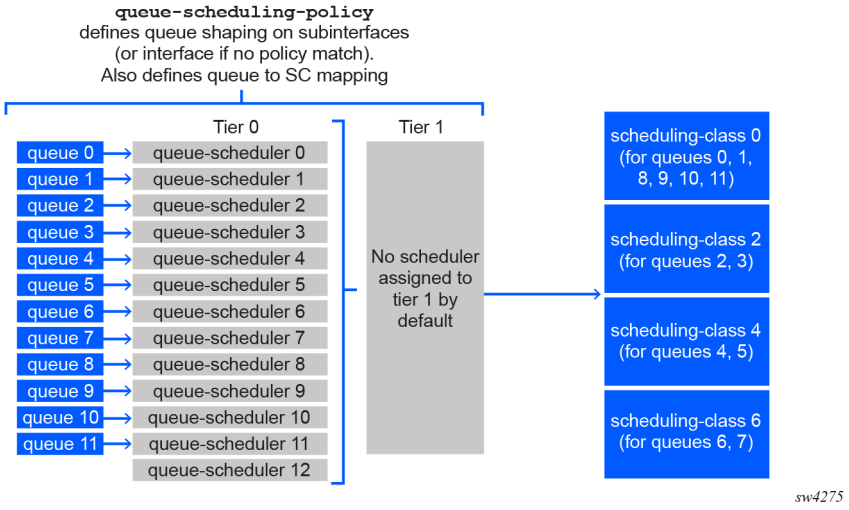
Figure 13: Queue scheduling policy and scheduling class scheduling policy



14.1 Queue scheduling policy (at subinterface [CVLAN] and interface-queue level)

Queue scheduling policies manage the distribution of scheduling resources available to process traffic in egress queues. The following figure illustrates the default queue scheduling policy.

Figure 14: Default queue scheduling policy



The queue scheduling policies define the following hierarchy of scheduling tiers:

- Tier 0: Up to 12 schedulers, each of which can provide shaping for single or multiple queues. The egress queues are processed initially by tier 0 schedulers, which in turn feed into the tier 1 scheduler.
- Tier 1: One scheduler, which provides aggregate-level shaping for all queues.

A total of 13 tier 0 and tier 1 schedulers can be defined. Within a queue scheduling policy, each defined queue must be mapped as an input to one of the tier 0 schedulers. The tier 0 schedulers are in turn mapped as inputs to the tier 1 scheduler. For each queue scheduler, shaper bucket thresholds determine how the scheduling resources are distributed by the scheduler.

The queue scheduling policies are applied at subinterface-level (such as CVLANs) to manage the defined subinterface queues. They are also applied at interface-level for any traffic that does not match any policies defined for subinterfaces.

The queue scheduling policies also support the mapping of each of the 12 egress queues to one of four scheduling classes: 0, 2, 4, and 6. If multiple queues are assigned to the same tier 0 scheduler, they must also be assigned to the same scheduling class.

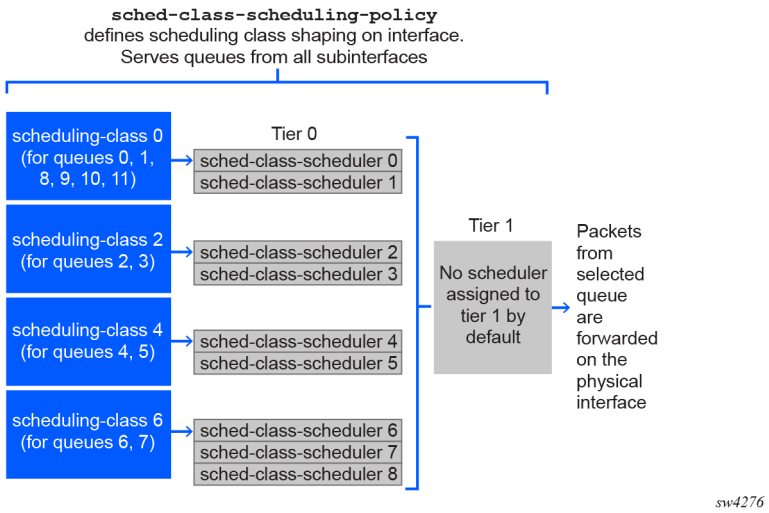
The queues within a tier 0 scheduler draw bandwidth proportional to their weights. The tier 0 scheduler draws bandwidth based on the scheduling class. The tier 0 schedulers with higher scheduling classes are scheduled ahead of the lower scheduling classes and are capped by their rate. The scheduler rate can be defined as an absolute value (from 64Kb/s to 400Gb/s), or as a percentage value of the line rate.

The interface-level scheduling class schedulers provide further shaping.

14.2 Scheduling class scheduling policy (at interface level)

Scheduling class scheduling policies manage the distribution of scheduling resources available to process traffic in scheduling classes. The following figure illustrates the default scheduling class scheduling policy.

Figure 15: Default scheduling class scheduling policy



Similar to queue scheduling policies, the scheduling class scheduling policies define the following hierarchy of scheduling tiers:

- Tier 0: Up to 8 schedulers, each of which can provide shaping for single or multiple scheduling classes exiting the queue schedulers. The scheduling classes are processed initially by tier 0 schedulers, which in turn feed into the tier 1 scheduler.
- Tier 1: One scheduler, which provides aggregate-level shaping for all scheduling classes.

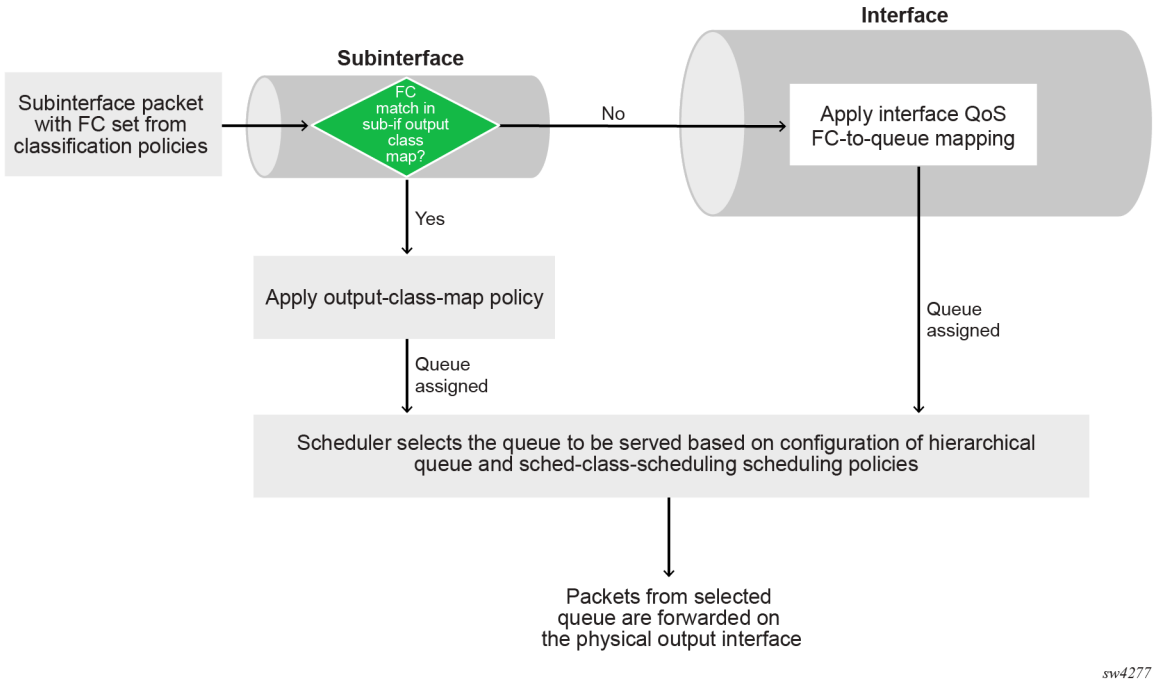
A total of 9 tier 0 and tier 1 schedulers can be defined. Also similar to queue schedulers, shaper bucket thresholds determine how the scheduling resources are distributed by the scheduler within each scheduling class scheduler.

The scheduling class scheduling policies are applied at interface level.

14.3 Queue and scheduling class policies applied to interface and subinterface

The following diagram shows the flow of packet processing through the queue and scheduling class scheduler policies applied to a subinterface and to an interface.

Figure 16: Applied queue and scheduling class policies



When a packet arrives on a subinterface for queueing at egress, the system performs the following steps.

1. Compare the packet FC (derived from one of the ingress classification policies) against the subinterface queue policies.
   - If a match is found, assign a queue for the packet based on the output class map policy.
   - If no match is found, assign the packet to a queue based on the interface-level FC-to-queue mapping.
2. The queues are scheduled based on the configured queue and scheduling class policies. The tier-1 scheduler allocates available bandwidth to its children serving the higher scheduling classes first and allocating the remaining bandwidth until none remains. And within a scheduling class, the available bandwidth is allocated to children in a weighted fair manner.

14.4 Queue scheduling policy example

The queue scheduling hierarchy consists of queues feeding multiple tier 0 schedulers which in turn feed into a single tier 1 scheduler that aggregates all queues from tier 0.

The following example shows the queue scheduling policy configuration for three queues, where **scheduler 4** in tier 1 is configured with the **inputs auto-input** parameter, which indicates that all tier 0 schedulers are inputs to scheduler 4. (The **auto-input** option cannot be set for a tier 0 scheduler.)

```
--{ candidate shared default }--[  ]--
# info with-context qos scheduler-policies queue-scheduling-policy test-sub-interface
    qos {
        scheduler-policies {
```

```
            queue-scheduling-policy test-sub-interface {
                queue af1-queue {
                    scheduling {
                        scheduling-class 4
                        weight 2
                        packet-length-adjustment {
                            add 0
                        }
                    }
                }
                queue af2-queue {
                    scheduling {
                        scheduling-class 4
                        weight 4
                        packet-length-adjustment {
                            add 0
                        }
                    }
                }
                queue be-queue {
                    scheduling {
                        scheduling-class 0
                        packet-length-adjustment {
                            add 0
                        }
                    }
                }
                queue ef-queue {
                    scheduling {
                        scheduling-class 6
                        packet-length-adjustment {
                            add 0
                        }
                    }
                }
                scheduler 0 {
                    tier 0
                    rate {
                        peak-rate-percentage 100
                    }
                    inputs {
                        queue [
                            be-queue
                        ]
                    }
                }
                scheduler 1 {
                    tier 0
                    rate {
                        peak-rate-percentage 80
                    }
                    inputs {
                        queue [
                            af1-queue
                            af2-queue
                        ]
                    }
                }
                scheduler 2 {
                    tier 0
                    rate {
                        peak-rate-percentage 20
                    }
                    inputs {
```

```
                        queue [
                            expedite-queue
                        ]
                    }
                }
                scheduler 4 {
                    tier 1
                    rate {
                        peak-rate-kbps 7000
                    }
                    inputs {
                        inputs auto-input
                    }
                }
            }
        }
    }
```

14.5 Scheduling class scheduling policy example

At interface-level, the scheduling hierarchy consists of scheduling classes feeding multiple tier 0 schedulers which in turn feed into a single tier 1 scheduler that aggregates all scheduling-classes from tier 0.

The following example shows the scheduling class scheduling policy configuration for three scheduling classes.

Similar to the queue policy, the tier 1 **scheduler 2** is configured with the **inputs auto-input** parameter, which indicates that all tier 0 schedulers are inputs to this scheduler. (If the **auto-input** option is set for a tier 0 scheduler, all defined queues are inputs to this scheduler. In that case, none of other tier 0 schedulers can have queues defined as explicit inputs.)

```
--{ candidate shared default }--[  ]--
# info with-context qos scheduler-policies sched-class-scheduling-policy sched-class-
scheduling-policy-name
    qos {
        scheduler-policies {
            sched-class-scheduling-policy sched-class-scheduling-policy-name {
                scheduler 1 {
                    tier 0
                    rate {
                        peak-rate-kbps 10000
                        pir-adaptation-rule closest
                    }
                    inputs {
                        scheduling-class [
                            0
                            4
                        ]
                    }
                }
                scheduler 2 {
                    tier 1
                    rate {
                        peak-rate-kbps 10000
                        pir-adaptation-rule closest
                    }
                    inputs {
                        inputs auto-input
```

```
                }
            }
            scheduler 3 {
                tier 0
                rate {
                    peak-rate-kbps 2000
                    pir-adaptation-rule closest
                }
                inputs {
                    scheduling-class [
                        6
                    ]
                }
            }
        }
    }
}
```

14.6 Default scheduling policy settings

Default interface queues

The following table shows the default settings applied to the interface-level queues. At interface level, all 12 queues are created by default. The table lists default settings for all queues, including the scheduling-related parameters.

Table 15: Default interface queues

default-queue-name	queue-index	committed-burst-size	maximum-burst-size	peak-rate-percent	weight	scheduling-class
queue-0	0	default	max	100	1	0
queue-1	1	default	max	100	1	0
queue-2	2	default	max	100	1	2
queue-3	3	default	max	100	1	2
queue-4	4	default	max	100	1	4
queue-5	5	default	max	100	1	4
queue-6	6	default	max	100	1	6
queue-7	7	default	max	100	1	6
queue-8	8	default	max	100	1	0
queue-9	9	default	max	100	1	0
queue-10	10	default	max	100	1	0
queue-11	11	default	max	100	1	0

To change the queue scheduling parameters, assign a custom queue scheduling policy to the interface. Similarly, to modify default scheduling behavior between scheduling classes at the interface level, assign a custom scheduling class scheduling policy to an interface.

Default subinterface queues

At subinterface level, local queues are created only after an output class map is assigned to the subinterface. By default, the queue parameters are set as described in the interface table above, using the queue-index as a key. To change queue scheduling parameters, assign a custom queue scheduling policy to the subinterface.

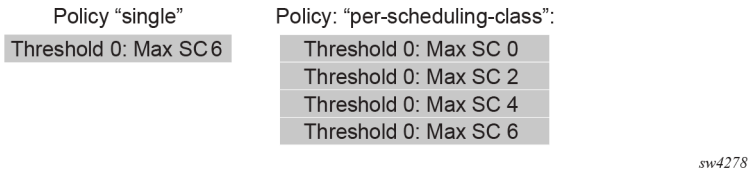
14.7 Default scheduling threshold policies

By default, the system provides the following two non-configurable scheduling threshold policies:

- `per-scheduling-class`
- `single`

The following diagram shows the default scheduling threshold policies. Both tier 0 schedulers are assigned the default scheduling-threshold-policy `single`, while both tier 1 schedulers are assigned the default scheduling-threshold-policy `per-scheduling-class`.

Figure 17: Default scheduling threshold policies



Custom-defined scheduling threshold policies are not supported.

Scheduling threshold policy: per-scheduling-class

The `per-scheduling-class` policy ensures that each scheduling class is assigned a distinct priority level within a shaper bucket.

The shaping of scheduling classes occurs in relation to the shaper token level. This process can be illustrated by considering scheduling class 3, which is associated with threshold level 3. If the token level is below threshold level 3, but above threshold level 2, queues associated with scheduling class 3 are shaped to enforce the overall rate of the bucket. Queues associated with scheduling class 3 are not shaped by this bucket if the shaper token level is at or above threhold level 3, or if the token level drops below threshold level 2.

The `per-scheduling-class` policy is assigned to tier 1 schedulers by default, both in the queue scheduling policies and the scheduling class scheduling policies. These settings are non-configurable.

The default `per-scheduling-class` policy is as follows:

```
qos {
   scheduling-policies {
      scheduling-threshold-policy "per-scheduling-class" {
         threshold 0 {
```

```
            maximum-scheduling-class 0
         }
         threshold 2 {
            maximum-scheduling-class 2
         }
         threshold 4 {
            maximum-scheduling-class 4
         }
         threshold 6 {
            maximum-scheduling-class 6
         }
      }
   }
}
```

Scheduling threshold policy: single

The `single` scheduling threshold policy places all scheduling classes on the first threshold level, avoiding multiple priority levels and the corresponding increases in burst tolerance.

The `single` policy is assigned to tier 0 schedulers by default, in both the queue scheduling policy and scheduling class scheduling policy. This setting is non-configurable.

```
qos {
   scheduling-policies {
      scheduling-threshold-policy "single" {
         threshold 0 {
            maximum-scheduling-class 6
         }
      }
   }
}
```

Modifying threshold level bucket depths

The size of the shaper buckets for all threshold levels used in the default scheduling threshold policies are defined in relation to the **burst-allowance** and **threshold-separation** parameters. These parameters are defined for queues and scheduling classes using queue scheduling policies and scheduling class scheduler policies.

14.8 Configuring queue scheduling policies

Procedure

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

Example: Configure a tier 0 queue scheduling policy

The following example shows a custom configuration for one queue and one scheduler in a tier 0 queue scheduling policy. The queue is assigned a scheduling class and weight, while the scheduler is assigned tier, burst, threshold-separation, and rate parameters and associated queues.

```
--{ * candidate shared default }--[  ]--
# info with-context qos scheduler-policies queue-scheduling-policy custom-tier-0-queue-
policy
    qos {
```

```
        scheduler-policies {
            queue-scheduling-policy custom-tier-0-queue-policy {
                queue queue-01 {
                    scheduling {
                        scheduling-class 0
                        weight 10
                    }
                }
                scheduler 0 {
                    tier 0
                    burst-allowance 10000
                    threshold-separation 28672
                    rate {
                        peak-rate-percentage 80
                    }
                    inputs {
                        queue [
                            queue-01
                        ]
                    }
                }
            }
        }
    }
```

Example: Configure a tier 1 queue scheduling policy

The following example shows a custom configuration for all queues in a tier 1 queue scheduling policy. The scheduler is assigned a tier, burst, threshold-separation, and rate parameters. All queues egressing the tier 0 schedulers feed into this single tier 1 scheduler using the **inputs auto-input** option.

```
--{ candidate shared default }--[  ]--
# info with-context qos scheduler-policies queue-scheduling-policy custom-tier-1-
scheduling-policy
    qos {
        scheduler-policies {
            queue-scheduling-policy custom-tier-1-scheduling-policy {
                scheduler 14 {
                    tier 1
                    burst-allowance 11000
                    threshold-separation 28672
                    rate {
                        peak-rate-percentage 100
                    }
                    inputs {
                        inputs auto-input
                    }
                }
            }
        }
    }
```

14.9 Applying queue scheduling policies to subinterfaces

Procedure

To apply queue scheduling policies to egress traffic on a subinterface, specify the required policy using the **qos interfaces interface output scheduler** command.

Example: Apply queue scheduling policies to a subinterface

The following example shows a tier 0 queue scheduling policy applied to a subinterface.

```
--{ * candidate shared default }--[  ]--
# info with-context qos interfaces
    qos {
        interfaces {
            interface eth-1/1.1 {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
                output {
                    scheduler {
                        queue-scheduling-policy custom-tier-0-queue-policy
                    }
                }
            }
        }
    }
```

14.10 Configuring scheduling class scheduling policies

Procedure

To configure scheduling class scheduling policies, use the **qos scheduler-policies sched-class-scheduling-policy** <*name*> command.

Example: Configure a tier 0 scheduling class scheduling policy

The following example shows a custom configuration for one scheduler included in a tier 0 scheduling class scheduling policy. The scheduler is assigned tier, burst, threshold-separation, and rate parameters, and associated scheduling classes.

```
--{ * candidate shared default }--[  ]--
# info with-context qos scheduler-policies sched-class-scheduling-policy custom-tier-0-
sched-class-policy
    qos {
        scheduler-policies {
            sched-class-scheduling-policy custom-tier-0-sched-class-policy {
                scheduler 0 {
                    tier 0
                    burst-allowance 10000
                    threshold-separation 28672
                    rate {
                        peak-rate-percentage 90
                    }
                    inputs {
                        scheduling-class [
                            0
                            2
                        ]
                    }
                }
            }
        }
    }
```

Example: Configure a tier 1 scheduling class scheduling policy

The following example shows a custom configuration for a tier 1 scheduling class scheduling policy. The scheduler is assigned tier, burst, threshold-separation, and rate parameters. All scheduling classes egressing the tier 0 schedulers feed into this single tier 1 scheduler using the **inputs auto-input** option.

```
--{ candidate shared default }--[  ]--
# info with-context qos scheduler-policies queue-scheduling-policy custom-tier-1-
scheduling-policy
    qos {
        scheduler-policies {
            queue-scheduling-policy custom-tier-1-scheduling-policy {
                scheduler 14 {
                    tier 1
                    burst-allowance 11000
                    threshold-separation 28672
                    rate {
                        peak-rate-percentage 100
                    }
                    inputs {
                        inputs auto-input
                    }
                }
            }
        }
    }
```

14.11 Applying scheduling class scheduling policies to interfaces

Procedure

To apply scheduling class scheduling policies to egress traffic on an interface, specify the required policy using the **qos interfaces interface output scheduler** command.

Example: Apply scheduling class scheduling policies to an interface

The following example shows a tier 0 scheduling class scheduling policy applied to a an interface.

```
--{ * candidate shared default }--[  ]--
# info with-context qos interfaces interface eth-1/1
    qos {
        interfaces {
            interface eth-1/1 {
                interface-ref {
                    interface ethernet-1/1
                }
                output {
                    scheduler {
                        sched-class-scheduling-policy custom-tier-0-sched-class-policy
                    }
                }
            }
        }
    }
```

14.12 Scheduling priority mapping table

The scheduling priority mapping table is a global table that maps individual scheduling classes into three scheduling priorities at the hardware level. Scheduling priority 0 is the lowest priority, and priority 2 is the highest.

The following table shows the default settings:

Scheduling class	Scheduling priority
0	0
2	0
4	1
6	2

14.12.1 Configuring the scheduling priority mapping table

Procedure

To configure the scheduling priority mapping table, use the **qos scheduler-policies scheduling-priority-mapping-table** command.

Example: Configure scheduling priority mapping table

```
--{ * candidate shared default }--[  ]--
# info with-context qos scheduler-policies scheduling-priority-mapping-table
    qos {
        scheduler-policies {
            scheduling-priority-mapping-table {
                scheduling-class 0 {
                    scheduling-priority 0
                }
                scheduling-class 2 {
                    scheduling-priority 0
                }
                scheduling-class 4 {
                    scheduling-priority 1
                }
                scheduling-class 6 {
                    scheduling-priority 2
                }
            }
        }
    }
```

14.13 Egress queue scheduling resource management

The resource management of output queues, tier 0 queue schedulers, and tier 1 queue schedulers is based on resource sets consisting of:

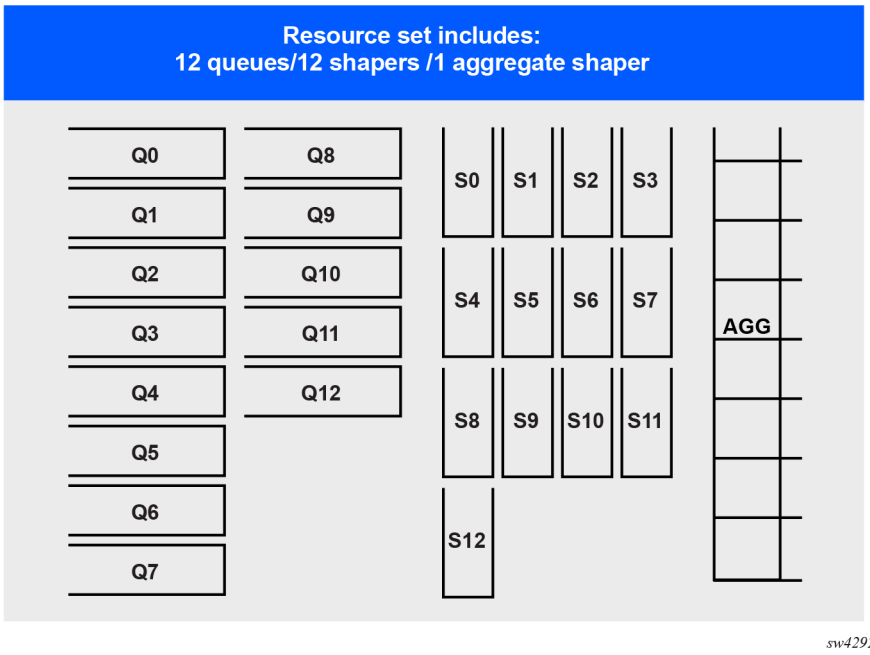
- 12 output queues

- 12 tier 0 queue schedulers
- one tier 1 queue scheduler

Every subinterface that is created is allocated a resource set regardless of the number of local queues configured in its associated output class map and regardless of the number of schedulers configured in its associated queue scheduling policy. This is to prevent changes in any of the scheduling policies from causing depletion of resources for the subinterface.

The following diagram shows a resource set consisting of queues, tier 0 schedulers, and a single tier 1 scheduler.

Figure 18: Resource set



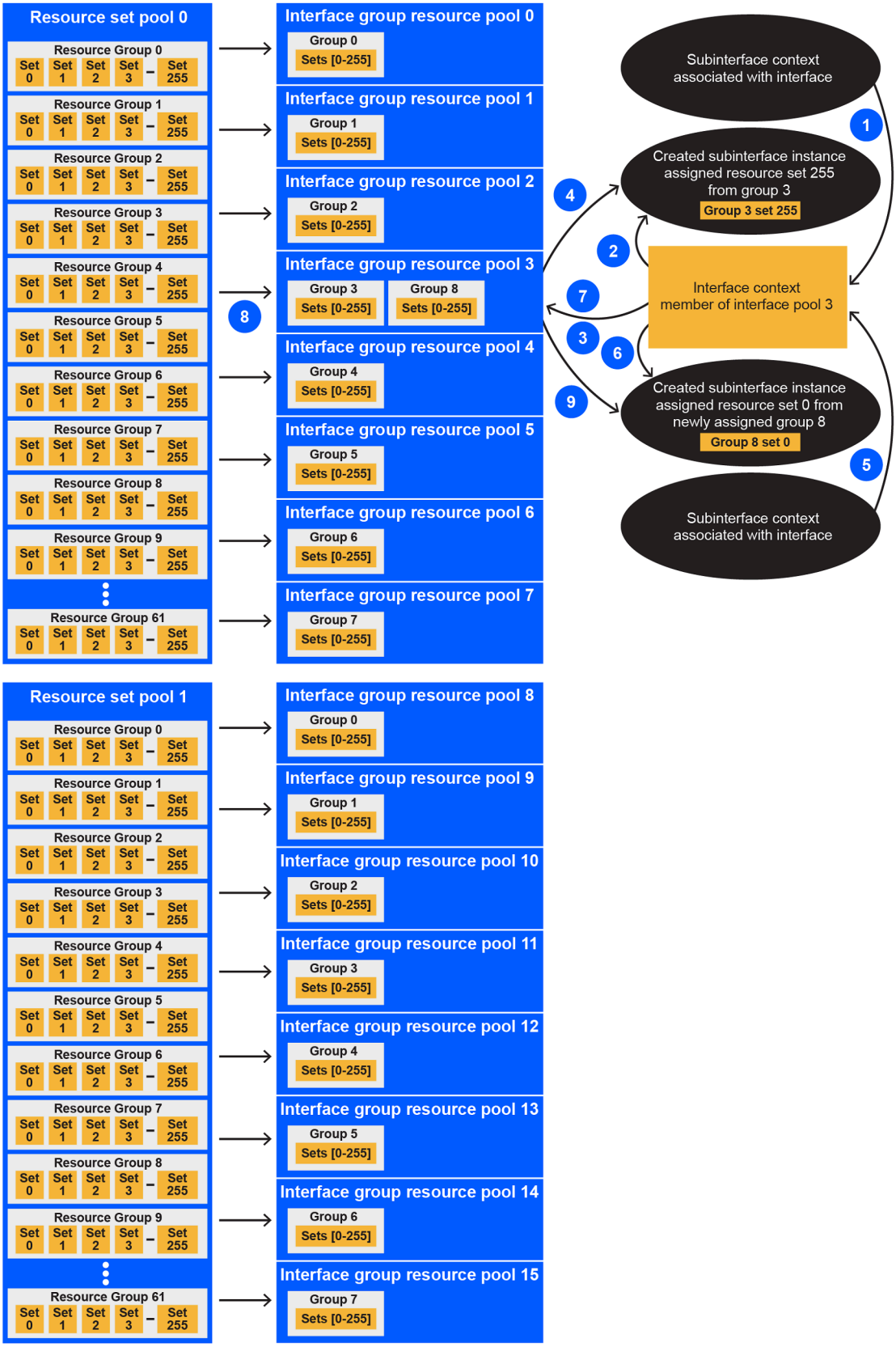
At the highest level, all resource sets are collected into two resource set pools. Each resource set pool contains 62 resource groups. And each resource groups contains 256 resource sets.

These resource groups are allocated to interface group resource pools, defining the interface-level resource sets that subinterfaces require to process queues.

At interface level, every interface is statically mapped to an interface group resource pool. The resource groups are dynamically allocated to the interface group resource pool based on demand. On an interface, all subinterfaces associated with the given interface group resource pool share the available resource sets.

The following figure shows the organization of resource sets to interfaces and subinterfaces.

Figure 19: Resource sets to interfaces and subinterfaces



sw4291

At system initialization, every interface group resource pool is allocated a single resource group from the corresponding resource set pool.

As the individual subinterfaces are created, the resource sets are allocated from the corresponding resource group. If a new subinterface is created and all resource sets in a group are already allocated, a new resource group is allocated to the interface group resource pool. This allocation is persistent, such that when a resource group is allocated to an interface group resource pool, it remains allocated even if all corresponding resource sets become free.

If subinterface queues are mapped to interface queues as a result of exhaustion of available resources, the `output-class-map-pending` flag indicates the interfaces that failed to allocate the queue resources. To view this flag, use the **info from state qos interfaces interface output output-class-map-pending** command.

In the system state information, a container is available under **scheduling-resources-pools** listing the scheduling resource pools individual interfaces are using resources from. This mapping is static, but it is platform dependent.

```
--{ candidate shared default }--[ ] --
# info from state qos interfaces interface eth1 output scheduler
    scheduling-resources-pools {
        resource-set-pool 1
        interface-group-resource-pool 12
    }
```

There is also a state table under **qos resource-set-pool** that tracks the utilization of the individual resource groups and their allocation to interface group resource pools.

```
--{ candidate shared default }--[ ] --
# info with-context from state platform linecard 1 forwarding-complex 1 qos
    qos {
        ...
        resource-set-pool 0 {
        ...
        }
        resource-set-pool 1 {
           interface-group-resource-pool 8 {
               resource-group 0 {
                   resource-set-used 253
                   resource-set-free 3
               }
           }
           interface-group-resource-pool 9 {
               resource-group 1 {
                   resource-set-used 0
                   resource-set-free 256
               }
           }
            interface-group-resource-pool 10 {
               resource-group 2 {
                   resource-set-used 0
                   resource-set-free 256
               }
           }
           interface-group-resource-pool 11 {
               resource-group 3 {
                   resource-set-used 255
                   resource-set-free 1
               }
               resource-group 8 {
                   resource-set-used 1
```

```
                resource-set-free 255
            }
        }
        ...
        interface-group-resource-pool 15 {
            resource-group 7 {
                resource-set-used 255
                resource-set-free 1
            }
        }
    }
}
```

Each configured subinterface requires one resource set at minimum. However, if the subinterface is created on a LAG, one resource set per LAG member is allocated. Depending on the configuration, this can lead to depletion of the resources. In such a case, the subinterface points to interface-based queues (remote-output-queue) overriding any output class map configuration. If the resource set in the corresponding interface group resource pool becomes available, the system automatically assigns it to the subinterface waiting for such a resource set.

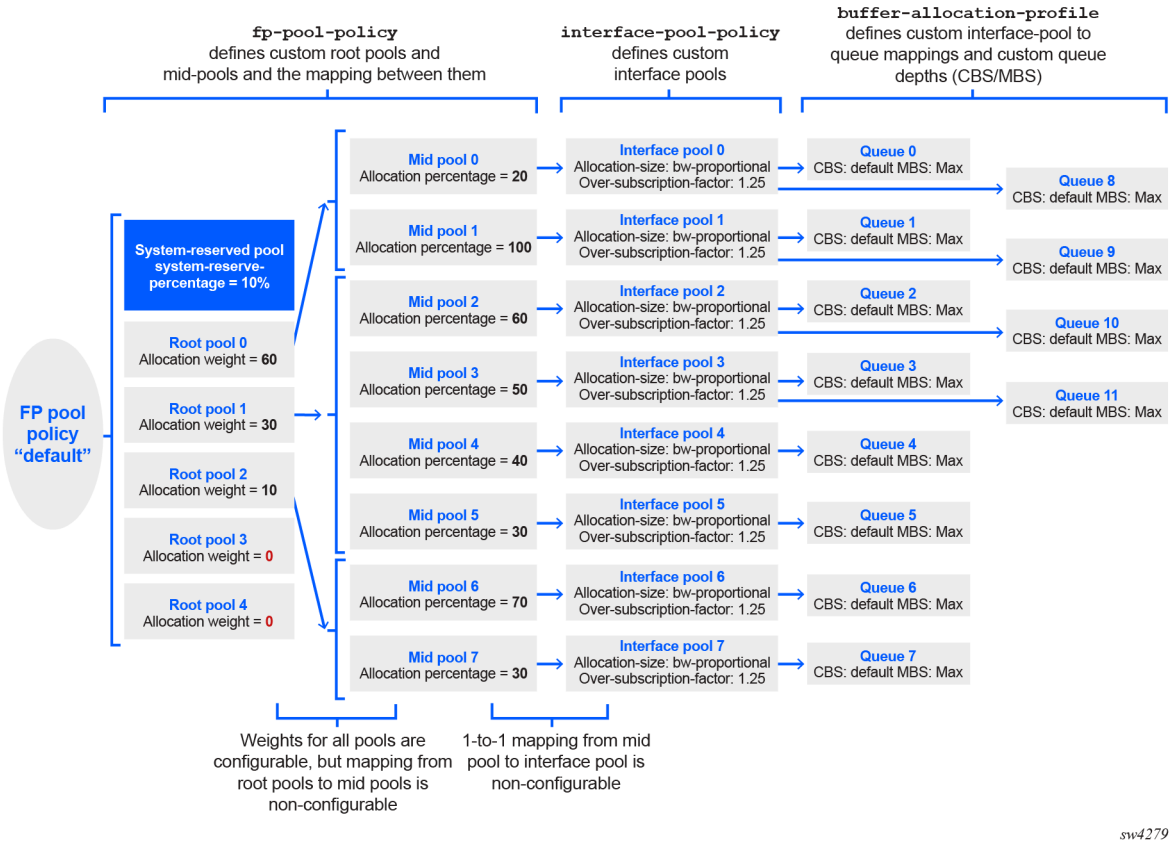
15 Egress buffer management

7730 SXR platforms support customized buffer allocation for egress queues beyond the default buffer pool behavior. The total system buffer allocation is divided into a system-reserved portion and a configurable portion. Within the configurable portion, the buffer hierarchy consists of the following levels:

- root pools
- mid-pools
- interface pools
- queues

The following diagram shows an overview of the default pools and default queue buffer allocations.

Figure 20: Default pools and default queue buffer allocations



Buffer management configuration elements

Buffer management on 7730 SXR platforms consists of the following four configurable elements:

- **FP pool policy (for root pool and mid-pool)**

The upper-level FP pool policy defines the buffer allocation for the root and mid-pools. On a linecard with two forwarding complexes, a different custom FP pool policy can be applied to each one of the forwarding complexes, as required.

- **Interface pool policy**

  The interface pool policy defines the interface-level buffer allocations.

  The mapping of interface pools to FP mid-pools is one-to-one and not configurable. The interface pool policy defines the interface buffer pool size as a percentage or proportion of the mid-pool size. All interfaces on an FP that are assigned to a particular interface pool share the buffer space of the associated mid-pool.

- **Buffer allocation profile**

  The buffer allocation profile associates the interface pool policy with individual queues. It also defines the queue depth parameters including maximum burst size (MBS), committed burst size (CBS) and the related adaptation rule parameter. The buffer allocation profile is applied to interfaces or subinterfaces as required.

  The 12 queues are always available on every interface at egress (queue names and parameters are configurable). On a subinterface, by default all subinterface traffic uses the same interface-level queues. Local queues are created on a subinterface only after an output class map is assigned to the subinterface.

- **WRED slope policy**

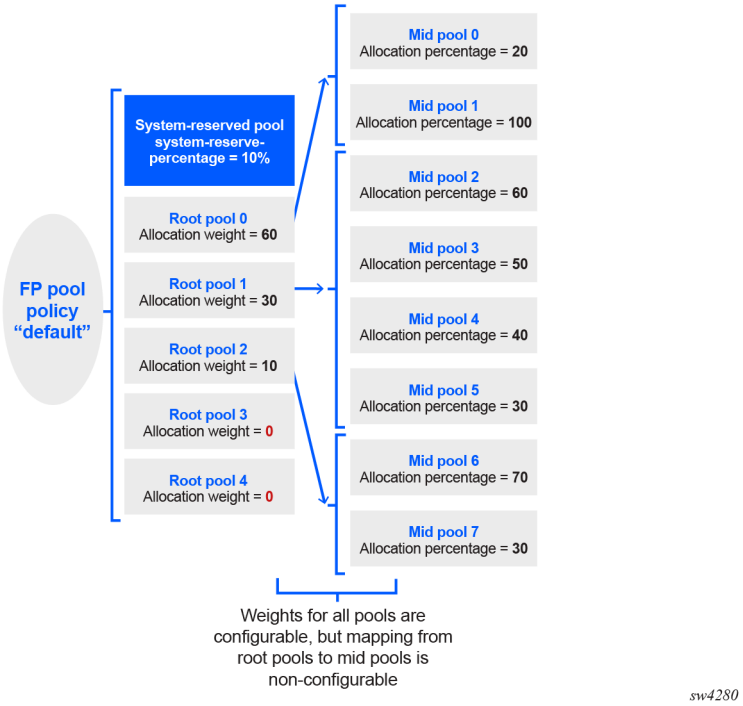
  A WRED slope policy can be applied to an FP pool policy, to an interface pool policy, or directly to queues to handle congestion when queue space is depleted.

15.1 FP pool policy for root pool and mid-pool

FP pool policies define the parameters for the system buffer root pools and mid-pools. All queues draw buffers from these pools as defined by the policies.

The following figure shows the default FP pool policy.

Figure 21: Default FP pool policy



In addition to the default FP pool policy shown, two additional configurable FP pool policies can be defined, to support up to two forwarding complexes. Within all three policies, the individual pool allocation weights are configurable, however the mapping of root pools to mid pools is static and non-configurable.

Root pools

The FP pool policy divides the FP buffer space into six root pools:

- One system-reserved pool for system generated traffic. The default value is 10% of total available buffer space.
- Five root pools, which divide the non-system-reserved FP buffer space using allocation weights relative to the total buffer space remaining.

The system-reserved pool is reserved for all system-generated traffic and for all traffic using queues created by the system (for example, failover queues).

A root pool can be used by different applications such as data, video, voice, or any other network application. The whole FP buffer space cannot be oversubscribed. Within each FP pool policy, all root pools (excluding the system-reserved pool) are assigned an **allocation-weight**, which determines the proportion of bandwith assigned to the pool in relation to the sum of all assigned pool weights in the policy.

Note: In the default FP pool policy, root pools 3 and 4 are not mapped to any mid-pools and are each assigned an allocation weight of 0, which means that they are not used.

Mid-pools

Mid-pools can represent different service classes within a single root pool.

Each forwarding complex supports a total of eight mid-pools (0 to 7), which are mapped to root pools 0, 1, and 2. The mid-pools can oversubscribe individual root pools, if required. If the mid-pool is not defined in the configuration, no buffer space is allocated to it. The mid-pool allocation percentages are configurable, however the mapping of mid-pools to root pools is static and non-configurable.

Default FP pool policy configuration

The following output shows the configuration for FP pool policy **default**, which is assigned to all forwarding complexes in the system by default.

```
--{ candidate shared default }--[  ]--
# info with-context from state qos buffer-management fp-pool-policy default
    qos {
        buffer-management {
            fp-pool-policy default {
                system-reserve-percentage 10
                root-tier {
                    root-pool 0 {
                        allocation-weight 60
                        mid-pool-members {
                            mid-pool-member 0 {
                            }
                            mid-pool-member 1 {
                            }
                        }
                    }
                    root-pool 1 {
                        allocation-weight 30
                        mid-pool-members {
                            mid-pool-member 2 {
                            }
                            mid-pool-member 3 {
                            }
                            mid-pool-member 4 {
                            }
                            mid-pool-member 5 {
                            }
                        }
                    }
                    root-pool 2 {
                        allocation-weight 10
                        mid-pool-members {
                            mid-pool-member 6 {
                            }
                            mid-pool-member 7 {
                            }
                        }
                    }
                    root-pool 3 {
                        allocation-weight 0
                    }
                    root-pool 4 {
                        allocation-weight 0
                    }
                }
                mid-tier {
                    mid-pool 0 {
                        allocation-percentage-size 20
                    }
                    mid-pool 1 {
                        allocation-percentage-size 100
                    }
```

```
                    mid-pool 2 {
                        allocation-percentage-size 60
                    }
                    mid-pool 3 {
                        allocation-percentage-size 50
                    }
                    mid-pool 4 {
                        allocation-percentage-size 40
                    }
                    mid-pool 5 {
                        allocation-percentage-size 30
                    }
                    mid-pool 6 {
                        allocation-percentage-size 70
                    }
                    mid-pool 7 {
                        allocation-percentage-size 30
                    }
                }
            }
        }
    }
```

Default settings for new FP pool policy

When a new custom defined FP pool policy is created, the following default settings apply:

- **system-reserve-percentage** = 10
- Mapping of the mid pools to root pools is the same as in the default FP pool policy, and is non-configurable.
- All mid-pool and root pool parameters are initially set the same as in the default FP pool policy.

15.2 Interface pool policy

Interface pools allow for a more precise allocation of the interface-level buffer space. Each interface can support up to eight interface pools, and the mapping from interface pools to FP-level mid-pools is one-to-one and not configurable. In other words, **interface-pool 0** maps to **mid-pool 0** and so on.

The default interface pool policy is assigned to every interface by default. A custom-defined interface pool policy can be attached to any interface at any time, although doing so is generally considered very exceptional.

Individual queues must still be mapped to corresponding interface pools using buffer allocation profiles, which are then assigned to interfaces and subinterfaces.

All interfaces that are assigned to a particular interface pool share the buffer space of the associated mid-pool. The size of the individual interface pools can be defined in one of two ways using the following parameters:

- **explicit-percentage**

  Sets the interface pool size as a percentage of the associated mid-pool size. The sum of participating interface pool percentages can be more than 100% (for oversubscription).
- **bw-proportional** and **over-subscription-factor**

The **bw-proportional** parameter divides the mid-pool size between participating interface pools based on their respective administrative interface rate (the nominal share). The **over-subscription-factor** defines how much the interface pool share is increased relative to its nominal share.

By default, the mid-pool buffer space is divided between interface pools using the bandwidth proportional approach, with an oversubscription factor of 1.25.

Default interface pool policy

The following figure illustrates the default interface pool policy and its mapping to the mid-pools.

Figure 22: Default interface pool policy

The following output shows the default interface pool policy configuration.

```
qos {
    buffer-management {
        interface-pool-policy default{
            interface-pool 0 {
                allocation-size {
                    bw-proportional {
                        over-subscription-factor 1.25
                    }
                }
            }
            interface-pool 1 {
                allocation-size {
                    bw-proportional {
                        over-subscription-factor 1.25
                    }
                }
            }
            interface-pool 2 {
```

```
                allocation-size {
                    bw-proportional {
                        over-subscription-factor 1.25
                    }
                }
            }
            interface-pool 3 {
                allocation-size {
                    bw-proportional {
                        over-subscription-factor 1.25
                    }
                }
            }
            interface-pool 4 {
                allocation-size {
                    bw-proportional {
                        over-subscription-factor 1.25
                    }
                }
            }
            interface-pool 5 {
                allocation-size {
                    bw-proportional {
                        over-subscription-factor 1.25
                    }
                }
            }
            interface-pool 6 {
                allocation-size {
                    bw-proportional {
                        over-subscription-factor 1.25
                    }
                }
            }
            interface-pool 7 {
                allocation-size {
                    bw-proportional {
                        over-subscription-factor 1.25
                    }
                }
            }
        }
    }
}
```

15.3 Buffer allocation profile

The buffer allocation profile defines the following for individual queues:

- The interface pool policy to associate with the queue
- The queue depth parameters including:
  - Maximum burst size (MBS)
  - Committed burst size (CBS)
  - MBS and CBS adaptation rules, which defines how the user-configured MBS and CBS values are adjusted to the available hardware values (**closest** [default], **higher**, or **lower**)

The buffer allocation profile is applied to interfaces and subinterfaces under the **qos interfaces interface output** context. It defines queue parameters for the interface or subinterface, depending on the context used.

The system provides default pools for queues that are not explicitly mapped to a pool in a buffer allocation profile.

Default buffer allocation profile

The following figure illustrates the default buffer allocation profile.

Figure 23: Default buffer allocation profile

The following output shows the default buffer allocation profile configuration.

```
--{  candidate shared default }--[  ]--
# info with-context from state qos buffer-management buffer-allocation-profile default
    qos {
        buffer-management {
            buffer-allocation-profile default {
                queues {
                    queue queue-0 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 0
                    }
                    queue queue-1 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 1
                    }
                    queue queue-10 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 2
                    }
                    queue queue-11 {
```

```
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 3
                    }
                    queue queue-2 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 2
                    }
                    queue queue-3 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 3
                    }
                    queue queue-4 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 4
                    }
                    queue queue-5 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 5
                    }
                    queue queue-6 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 6
                    }
                    queue queue-7 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 7
                    }
                    queue queue-8 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 0
                    }
                    queue queue-9 {
                        maximum-burst-size 0
                        mbs-adaptation-rule closest
                        interface-pool 1
                    }
                }
            }
        }
    }
```

15.3.1 Committed burst size table

7730 SXR platforms support only four configurable values for the committed burst size. A single global table defines the four possible values for the system. At the individual queue level, the explicit committed burst size is defined and the adaptation rule parameter defines how this value is rounded to one of the four possible values. After being adapted to one of the four committed burst size table values, the configured CBS values are then adapted to the available hardware values. This latter adaptation always uses the **closest** algorithm.

The actual CBS value can be viewed in state using the **info from state qos interfaces interface** <*id*> **output queues queue** <*name*> command for the subinterface.

The committed burst size table is defined under the **qos buffer-management** context. Below is the default configuration for this table.

You can modify the table settings at any time. The corresponding values are changed accordingly for all active queues.

Default committed burst size table

```
qos {
   buffer-management {
      committed-burst-size-table {
         alt-0 0
         alt-1 0
         alt-2 0
         alt-3 0
      }
   }
}
```

The 7730 SXR platforms also support disabling of CBS altogether by omitting the **committed-burst-size** parameter from the buffer allocation profile. If CBS is disabled, the datapath does not perform a lookup in the committed burst size table.

15.4 WRED slope policy

To implement WRED behavior, a slope policy must be attached to an object. Three slopes are defined in a slope policy, for in, out, and exceed profiles. Packets matching the in-plus profile are not subject to the WRED slope.

Without a WRED slope policy applied, when a queue reaches its maximum fill size, the queue discards any packets arriving at the queue (known as tail drop).

The WRED slope policy can be applied to root pools, mid-pools, and interface pools. Mid-pools and interface pools can each have their own slope policy attached. However, for root pools, only a default slope policy can be defined.

In addition, the WRED slope policy can be applied to a specified forwarding class for all associated interface or subinterface queues.

A WRED slope policy can also emulate RED behaviour by defining all profiles (in, out, and exceed) with the same values, or by mapping all packets to a single profile.

Default WRED slope policy

The default WRED slope policy is as follows.

```
qos {
    buffer-management {
        slope-policy default {
            wred-slope in {
                slope-enabled true
                min-threshold-percent 75
                max-threshold-percent 90
                max-probability 80
            }
            wred-slope out {
                slope-enabled true
```

```
                min-threshold-percent 50
                max-threshold-percent 75
                max-probability 80
            }
            wred-slope exceed {
                slope-enabled true
                min-threshold-percent 30
                max-threshold-percent 55
                max-probability 80
            }
        }
    }
}
```

Default settings for new WRED slope policy

When a new custom defined WRED slope policy is created, by default the following parameters are applied.

```
qos {
    buffer-management {
        slope-policy slope-policy-name {
            wred-slope in {
                slope-enabled false
                min-threshold-percent 85
                max-threshold-percent 100
                max-probability 80
            }
            wred-slope out {
                slope-enabled false
                min-threshold-percent 85
                max-threshold-percent 100
                max-probability 80
            }
            wred-slope exceed {
                slope-enabled false
                min-threshold-percent 85
                max-threshold-percent 100
                max-probability 80
            }
        }
    }
}
```

15.5 Buffer usage monitoring

For every non-reserved pool, the system maintains the operational size and actual usage of the pool. (For system reserved pools, only the operational size is maintained.) The platform collects these values at regular intervals and publishes them in the internal database, similar to other resource usage.

To display buffer usage monitoring, use the **info from state platform linecard forwarding-complex buffer-memory** command.

Example: Display buffer usage monitoring for root pool 1

```
--{ candidate shared default }--[  ]--
# info with-context from state platform linecard 1 forwarding-complex 1 buffer-memory
 root-pool 1
```

```
    platform {
        linecard 1 {
            forwarding-complex 1 {
                buffer-memory {
                    root-pool 1 {
                        operational-size 8704
                        used 0
                        mid-pool 2 {
                            operational-size 5248
                            used 0
                        }
                        mid-pool 3 {
                            operational-size 4352
                            used 0
                        }
                        mid-pool 4 {
                            operational-size 3504
                            used 0
                        }
                        mid-pool 5 {
                            operational-size 2624
                            used 0
                        }
                    }
                }
            }
        }
    }
```

Example: Display buffer usage monitoring for the system-reserved pool

```
--{ candidate shared default }--[  ]--
# info with-context from state platform linecard 1 forwarding-complex 1 buffer-memory
 system-reserved-pool
    platform {
        linecard 1 {
            forwarding-complex 1 {
                buffer-memory {
                    system-reserved-pool {
                        operational-size 1661952
                    }
                }
            }
        }
    }
```

15.6 Configuring FP pool policies for root pool and mid-pool

Procedure

To configure FP pool policies for root pool and mid-pool buffers, use the **qos buffer-management fp-pool-policy** command. The individual allocation weights are configurable for all pools, however the default mapping of root pools to mid-pools is not configurable.

Example: Configure FP pool policy

```
--{ candidate shared default }--[  ]--
# info with-context qos buffer-management fp-pool-policy fp-pool-policy-name
    qos {
```

```
        buffer-management {
            fp-pool-policy fp-pool-policy-name {
                system-reserve-percentage 15
                root-tier {
                    default-slope-policy default-slope-policy-name
                    root-pool 0 {
                        allocation-weight 65
                        mid-pool-members {
                            mid-pool-member 0 {
                            }
                            mid-pool-member 1 {
                            }
                        }
                    }
                    root-pool 1 {
                        allocation-weight 35
                        mid-pool-members {
                            mid-pool-member 2 {
                            }
                            mid-pool-member 3 {
                            }
                            mid-pool-member 4 {
                            }
                            mid-pool-member 5 {
                            }
                        }
                    }
                }
                mid-tier {
                    mid-pool 0 {
                        allocation-percentage-size 45
                        slope-policy slope-policy-name
                    }
                    mid-pool 1 {
                        allocation-percentage-size 30
                    }
                    mid-pool 2 {
                        allocation-percentage-size 25
                    }
                    mid-pool 3 {
                        allocation-percentage-size 20
                    }
                    mid-pool 4 {
                        allocation-percentage-size 70
                    }
                    mid-pool 5 {
                        allocation-percentage-size 30
                    }
                }
            }
        }
    }
```

15.7 Applying an FP pool policy to a forwarding complex

About this task

You can assign one FP pool policy per forwarding complex. For linecards with two forwarding complexes, you can assign different custom policies on each forwarding complex.

Procedure

To apply an FP pool policy to a forwarding complex, use the **qos linecard** <*slot*> **forwarding-complex** <*name*> **output fp-pool-policy** <*value*> command.

Example: Apply an FP pool policy to a forwarding complex

```
--{ + candidate shared default }--[  ]--
A:sxr1x44s# info with-context qos linecard 1
    qos {
        linecard 1 {
            forwarding-complex 0 {
                output {
                    fp-pool-policy fp-pool-policy-name
                }
            }
        }
    }
```

15.8 Configuring interface pool policies

Procedure

To configure interface pool polices, use the **qos buffer-management interface-pool-policy** <*name*> command.

Example: Configure interface pool policy

```
--{ candidate shared default }--[  ]--
# info with-context qos buffer-management interface-pool-policy interface-pool-policy-name
    qos {
        buffer-management {
            interface-pool-policy interface-pool-policy-name {
                interface-pool 0 {
                    allocation-size {
                        bw-proportional {
                            over-subscription-factor 1.30
                        }
                    }
                }
                interface-pool 1 {
                    allocation-size {
                        bw-proportional {
                            over-subscription-factor 1.20
                        }
                    }
                }
                interface-pool 2 {
                    allocation-size {
                        bw-proportional {
                            over-subscription-factor 1.10
                        }
                    }
                }
            }
        }
    }
```

15.9 Applying interface pool policies to an interface

Procedure

To apply interface pool policies to an interface, use the **interface-pool-policy** command in the **qos interfaces interface** <*name*> **output** context. Each interface can support up to eight interface pools.

An interface pool policy can be applied only to a QoS interface that references an interface, not a subinterface.

Note: To map individual queues into a corresponding interface pool, you must also configure a buffer allocation profile and assign it to an interface.

Example: Apply interface pool policies to an interface

```
--{ +* candidate shared default }--[  ]--
# info with-context qos interfaces interface eth-1/2
    qos {
        interfaces {
            interface eth-1/2 {
                interface-ref {
                    interface ethernet-1/2
                }
                output {
                    interface-pool-policy interface-pool-policy-name
                }
            }
        }
    }
```

15.10 Configuring buffer allocation profiles

Procedure

To configure buffer allocation profiles, use the **qos buffer-management buffer-allocation-profile** command.

Example: Configure buffer allocation profile

```
--{ * candidate shared default }--[  ]--
# info with-context qos buffer-management
    qos {
        buffer-management {
            buffer-allocation-profile buffer-allocation-profile-name {
                queues {
                    queue queue-0 {
                        maximum-burst-size 80
                        mbs-adaptation-rule closest
                        interface-pool 0
                    }
                    queue queue-1 {
                        maximum-burst-size 85
                        mbs-adaptation-rule closest
                        interface-pool 0
                    }
                    queue queue-2 {
```

```
                        maximum-burst-size 90
                        mbs-adaptation-rule closest
                        interface-pool 1
                    }
                }
            }
        }
    }
```

15.11 Applying buffer allocation profiles to an interface or subinterface

Procedure

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

Example: Apply buffer allocation profile to a subinterface

```
--{ * candidate shared default }--[  ]--
# info with-context qos interfaces interface ethernet-1/2.1
    qos {
        interfaces {
            interface ethernet-1/2.1 {
                interface-ref {
                    interface ethernet-1/2
                    subinterface 1
                }
                output {
                    buffer-allocation-profile test-buffer-profile
                }
            }
        }
    }
```

15.12 Configuring WRED slope policies

Procedure

To configure WRED slope policies, use the **qos buffer-management slope-policy** command.

Example: Configure WRED slope policy

```
--{ * candidate shared default }--[  ]--
# info with-context qos buffer-management
    qos {
        buffer-management {
            slope-policy slope-policy-name {
                wred-slope in {
                    slope-enabled true
                    min-threshold-percent 80
                    max-threshold-percent 95
                    max-probability 75
                }
                wred-slope out {
                    slope-enabled true
                    min-threshold-percent 55
```

```
                    max-threshold-percent 80
                    max-probability 85
                }
                wred-slope exceed {
                    slope-enabled true
                    min-threshold-percent 35
                    max-threshold-percent 60
                    max-probability 95
                }
            }
        }
    }
```

The configured WRED slope policy can be attached to the following objects:

- Interface forwarding-class under **qos forwarding-classes forwarding-class**.
  In this case the slope policy applies to all interface-level queues associated with the forwarding class.
- Subinterface forwarding-class under **qos output-class-map forwarding-class**.
  In this case the slope policy applies to all subinterface-level queues associated with the forwarding class.
- Root pools under **qos buffer-management fp-pool-policy root-tier**.
  In this case a single common slope policy applies to all root pools.
- Mid-pools under **qos buffer-management fp-pool-policy mid-tier mid-pool**.
  In this case a unique slope policy can be applied to each mid-pool.
- Interface pool under **qos buffer-management interface-pool-policy interface-pool**.
  In this case a unique slope policy can be applied to each interface pool.

15.12.1 Applying default WRED slope policies to root pools

Procedure

To apply a WRED slope policy to serve as the default policy for all root pools, use the following command.

qos buffer-management fp-pool-policy *<name>* **root-tier default-slope-policy** *<name>*

Example: Apply a default WRED slope policy to a root pool

```
--{ +* candidate shared default }--[  ]--
# info with-context qos buffer-management fp-pool-policy fp-pool-policy-name root-tier
 default-slope-policy
    qos {
        buffer-management {
            fp-pool-policy fp-pool-policy-name {
                root-tier {
                    default-slope-policy default-slope-policy-name
                }
            }
        }
    }
```

15.12.2 Applying WRED slope policies to mid-pools

Procedure

To apply a WRED slope policy to a mid-pool use the following command.

qos buffer-management fp-pool-policy <*name*> **mid-tier mid-pool** <*value*> **slope-policy** <*name*>

Example: Apply a WRED slope policy to mid-pool 0

```
--{ +* candidate shared default }--[ ]--
# info with-context qos buffer-management fp-pool-policy fp-pool-policy-name mid-tier mid-
pool 0
    qos {
        buffer-management {
            fp-pool-policy fp-pool-policy-name {
                mid-tier {
                    mid-pool 0 {
                        slope-policy slope-policy-name
                    }
                }
            }
        }
    }
```

15.12.3 Applying WRED slope policies to interface pools

Procedure

To apply a WRED slope policy to an interface pool use the following command.

qos buffer-management interface-pool-policy <*name*> **interface-pool** <*value*> **slope-policy** <*name*>

Example: Apply a WRED slope policy to interface pool 0

```
--{ +* candidate shared default }--[  ]--
# info with-context qos buffer-management interface-pool-policy interface-pool-name
 interface-pool 0 slope-policy
    qos {
        buffer-management {
            interface-pool-policy interface-pool-name {
                interface-pool 0 {
                    slope-policy slope-policy-name
                }
            }
        }
    }
```

15.12.4 Applying WRED slope policies to interface-level forwarding classes

Procedure

To apply a WRED slope policy to a forwarding class for interface-level queues, use the **slope-policy** command under the following context.

qos forwarding-classes forwarding-class <*name*> **output**

Example: Apply WRED slope policy to interface-level forwarding classes

The following example applies a WRED slope policy to all interface-level queues associated with the specified forwarding class.

```
--{ * candidate shared default }--[  ]--
# info with-context qos forwarding-classes forwarding-class fc-name
    qos {
        forwarding-classes {
            forwarding-class fc-name {
                output {
                    slope-policy slope-policy-name
                }
            }
        }
    }
```

15.12.5 Applying WRED slope policies subinterface-level forwarding classes

Procedure

To apply a WRED slope policy to a forwarding class for subinterface-level queues, use the **slope-policy** command under the following context.

qos output-class-map<*name*> **forwarding-class** <*name*>

Example: Apply WRED slope policy to subinterface-level forwarding classes

The following example applies a WRED slope policy to all subinterface-level queues associated with the specified forwarding class.

```
--{ * candidate shared default }--[  ]--
# info with-context qos output-class-map output-class-map-name
    qos {
        output-class-map output-class-map-name {
            forwarding-class fc-name {
                slope-policy slope-policy-name
            }
        }
    }
```

16 QoS resource management tables

On 7730 SXR platforms, resource management tables define how different internal resources are used in times of overload. The resource management tables are global, such that the same configuration applies to all forwarding complexes in the system.

Note: Because QoS resource management tables involve a very advanced level of configuration, consult first with Nokia personnel before making any changes to these tables.

The resource management tables govern the behavior of the following internal resources.

- **Buffer segments**
  These are 512 byte buffer segments. As packets arrive, the first 384 bytes of the packet are sent to the thread processing the packet. The remaining portion of the packet is assigned to buffer segments in memory. If the packet is forwarded without queuing, the portion in the thread buffer does not consume a buffer segment. If the packet is queued, the thread portion is moved into memory, consuming a buffer segment. When the assigned buffer segments are sent to the transmit pipeline for forwarding, the buffer segments are freed.
- **Packet IDs**
  Packet ID is the resource at each slice (400G or I/O) that tracks the ordering of in-flight packets. As packets are received, they are assigned a packet ID context in the receiving slice's Re-Order-Engine (ROE). The packet is then handed off to a thread for processing. The packet ID is initially placed in the main ROE context for the slice. Early in processing, the thread makes a request to the ROE to move the packet's ID from the main reordering queue to an ROE subcontext queue (one of 1024 per slice). The ROE context waits for the packet ID to reach the head of the main queue and then moves the packet ID to the specified subcontext queue. As the thread finishes processing the packet, it is either moved into a queue for future egress forwarding or sent directly to the target port's transmit pipeline. This only happens when the packet ID reaches the head of the subcontext's queue. This mechanism keeps packets in order based on the subcontext, preventing out-of-order forwarding.

  A limited number of packet IDs exist and the slice's ROE can exhaust its free list of packet IDs in the rare case where packets are arriving faster than being released. Longer processing times in the thread are a primary cause of packet ID exhaustion. To mitigate this issue, the ROE subcontexts are grouped based on expected processing time. This minimizes head-of-line blocking within the subcontexts.
- **Header buffers**
  Each thread has two header buffers used to process packets. When the system first comes up, each thread advertises one of the header buffers to an I/O slice and waits for a packet to be assigned by the slice's scatter block. Just before processing is finished for the first packet, the other header buffer is advertised to allow another packet to be assigned to the thread to allow the thread to minimize the amount of time the thread is idle between packets. While the thread is processing the new packet, the system works in the background to flush the older header buffer. When the older header buffer is flushed and the thread has neared the end of processing the newer packet, the old header buffer is advertised as available. The alternate advertising of the thread's header buffers continues in this fashion.

The following three tables control the logic related to resource management on 7730 SXR platforms:

- Forwarding class resource priority table
- Resource utilization zones table (non-configurable)

- Drop zone table (non-configurable)

16.1 Forwarding class resource priority table

The forwarding class resource priority table maps forwarding class and profile combinations for unicast and multicast traffic into one of four priorities (**0** to **3**); where **0** corresponds to the lowest priority and **3** is the highest.

The following shows the default settings for the forwarding class resource priority table:

Table 16: Default forwarding class resource priority table

Default FC name	FC index	Priority	in-plus	in	out	exceed	in-low	out-low
fc0	0	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc1	1	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc2	2	unicast priority	1	1	0	0	0	0
		multicast priority	1	1	0	0	0	0
fc3	3	unicast priority	1	1	0	0	0	0
		multicast priority	1	1	0	0	0	0
fc4	4	unicast priority	2	2	1	1	0	0
		multicast priority	2	2	1	1	0	0
fc5	5	unicast priority	2	2	1	1	0	0
		multicast priority	2	2	1	1	0	0
fc6	6	unicast priority	3	3	2	2	0	0

Default FC name	FC index	Priority	in-plus	in	out	exceed	in-low	out-low
		multicast priority	3	3	2	2	0	0
fc7	7	unicast priority	3	3	2	2	0	0
		multicast priority	3	3	2	2	0	0
fc8	8	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc9	9	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc10	10	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc11	11	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc12	12	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc13	13	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc14	14	unicast priority	0	0	0	0	0	0
		multicast priority	0	0	0	0	0	0
fc15	15	unicast priority	0	0	0	0	0	0

Default FC name	FC index	Priority	in-plus	in	out	exceed	in-low	out-low
		multicast priority	0	0	0	0	0	0

16.2 Configuring the forwarding class resource priority table

Procedure

To configure the forwarding class resource priority table, use the **qos resource-management forwarding-class-resource-priority** command.

Example: Configure the forwarding class resource priority table

The following example shows the configuration of priority values for unicast and multicast traffic for **forwarding-class fc1** with profile **out**.

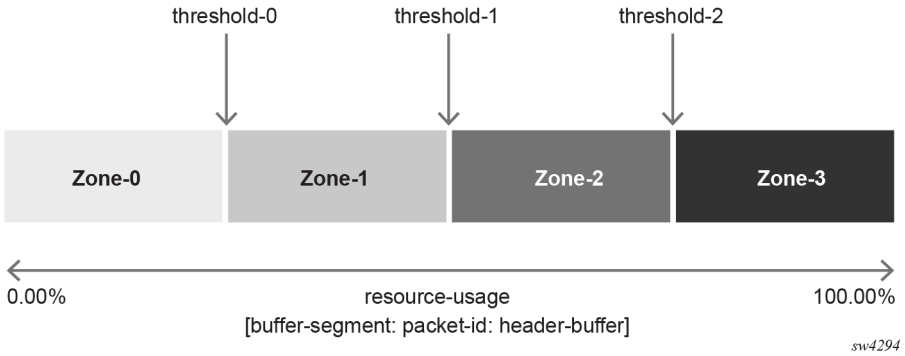
```
--{ + candidate shared default }--[  ]--
# info with-context qos resource-management
    qos {
        resource-management {
            forwarding-class-resource-priority {
                forwarding-class fc1 {
                    profile out {
                        unicast-resource-priority 1
                        multicast-resource-priority 0
                    }
                }
            }
        }
    }
```

16.3 Resource utilization thresholds table

The utilization of the individual resources (buffer segments, packet IDs, and header buffers) are divided into zones defined by the utilization thresholds for each resource. As packets arrive, the current zone for each resource is sent to the thread processing the packet. When the thread determines the resource priority for the current packet, it identifies the configured zone where that resource priority can cause an early discard of the packet. If the drop-zone has been exceeded, the thread discards the packet and increments the resource discard counter. If the drop-zone has not been exceeded, processing continues. As utilization increases and decreases, the current zone for each resource changes accordingly.

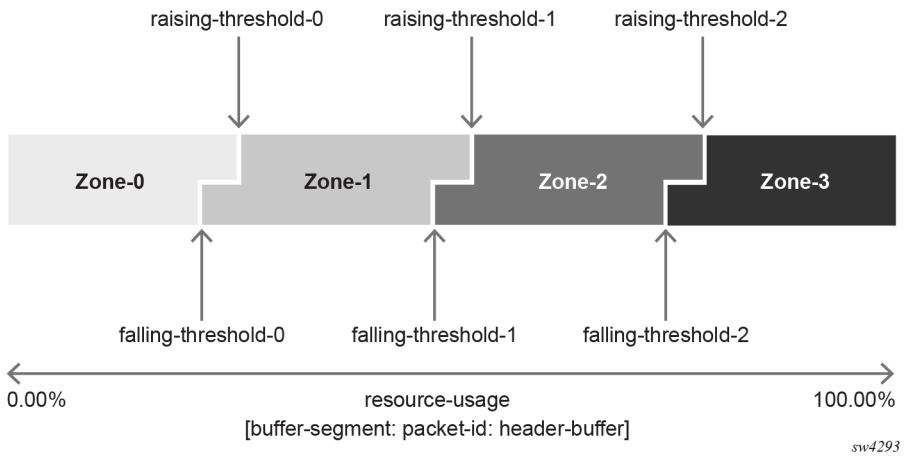
Conceptually, every resource is divided into four zones by configuring three thresholds, as shown in the figure below. Zone 0 indicates no congestion and therefore no drop is experienced while the resource utilization is within this zone.

Figure 24: Resource utilization thresholds



In practice, there are two thresholds defining each zone: the rising threshold and the falling threshold. Each threshold defines the boundaries between individual zones and are defined in term of percentage value (with granularity to 2 decimals) of the maximum resource value, as shown in the following figure.

Figure 25: Resouce utlization thresholds: rising and falling



The default, non-configurable values for the resource utilization zones table are defined as follows.

Table 17: Default resource utilization thresholds table

	threshold-0		threshold-1		threshold-2	
	rising	falling	rising	falling	rising	falling
segment-buffer	80	78	84	82	90	88
packet-id	80	78	84	82	90	88
header-buffer	88	86	92	90	96	94

16.4 Drop zone table

The drop zone table defines specific drop zones in which packets that correspond to a given resource priority are dropped. The drop zones are defined separately for unicast and multicast traffic. The resource priorities are defined by the mapping in the forwarding class resource priority table.

The default, non-configurable settings for the drop zone table are as follows.

Table 18: Default drop zone table

Drop zone	Unicast priority				Multicast priority			
	0	1	2	3	0	1	2	3
buffer-segment-drop-zone	1	2	2	3	2	2	3	3
packet-id-drop-zone	1	2	2	3	2	2	3	3
header-buffer-drop-zone	1	2	2	3	2	2	3	3

17 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
```

18 QoS profile resource usage

A QoS profile resource refers to the number of classifier and rewrite policies that are configured on a forwarding complex. Each classifier or rewrite policy that is created counts as one profile resource used, regardless of the number of subinterfaces it is applied to.

18.1 Displaying QoS profile resource usage

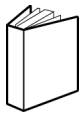
Procedure

To display QoS profile resource usage, 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

```
--{ candidate shared default }--[  ]--
# info with-context from state platform linecard 1 forwarding-complex 0 qos
    platform {
        linecard 1 {
            forwarding-complex 0 {
                qos {
                    resource classifier-profiles {
                    }
                    resource dot1p-rewrite-policies {
                        used 1
                        free 62
                    }
                    resource dscp-mpls-rewrite-policies {
                        used 1
                        free 62
                    }
                    resource dscp-rewrite-policies {
                        used 1
                        free 62
                    }
                    resource input-policers {
                        used 0
                        free 512000
                    }
                    resource output-class-maps {
                        used 0
                        free 256
                    }
                    resource rewrite-profiles {
                    }
                    resource-set-pool 0 {
                    }
                    resource-set-pool 1 {
                    }
                }
            }
        }
    }
```

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback