RBFS HQoS Configuration Guide

Version 20.9.2, 26 October 2020

Registered Address	Support	Sales
26, Kingston Terrace, Princeton, New Jersey 08540, United States		
		+91 80 4850 5445
http://www.rtbrick.com	support@rtbrick.com	sales@rtbrick.com

©Copyright 2020 RtBrick, Inc. All rights reserved. The information contained herein is subject to change without notice. The trademarks, logos and service marks ("Marks") displayed in this documentation are the property of RtBrick in the United States and other countries. Use of the Marks are subject to RtBrick's Term of Use Policy, available at https://www.rtbrick.com/privacy. Use of marks belonging to other parties is for informational purposes only.

Table of Contents

1. Introduction to Hierarchical Quality of Service (HQoS) . . . 4
1.1. MPLS HQoS . . . 5
1.1.1. Components of the RBFS QoS Solution . . . 5
1.2. Supported Hardware . . . 5
2. HQoS Features . . . 6
2.1. Priority Propagation . . . 6
2.1.1. Simple Priority Propagation Scheduling Example . . . 9
2.2. Behavior Aggregate (BA) Classifier . . . 10
2.3. Multifield (MF) Classifier . . . 11
2.4. Remarking . . . 12
2.5. Policer . . . 13
2.5.1. Class-Policer-Map . . . 15
2.6. Queueing . . . 16
2.6.1. Class-Queue-Map . . . 16
2.6.2. Queue-Group . . . 17
2.7. Scheduler . . . 17
2.8. Scheduler-Map . . . 22
2.9. Shaper . . . 22
2.10. Profiles . . . 23
2.11. Header Compensation . . . 23
2.11.1. Queue Compensation . . . 23
2.11.2. Port Compensation . . . 23
2.12. Multi-level H-QoS : Level-1 to Level-5 . . . 24
3. Configuring HQoS . . . 26
3.1. Behavior Aggregate (BA) Classifier Configuration . . . 30
3.1.1. BA Classifier to Profile Mapping . . . 30
3.1.2. BA Classifier to Global Mapping . . . 31
3.1.3. BA Classifier to Instance Mapping . . . 31
3.2. Multifield Classifier Configuration . . . 31
3.2.1. IPv4 Match Configuration . . . 32
3.2.2. IPv6 Match Configuration . . . 33
3.2.3. IPv4/IPv6 Action Configuration . . . 34
3.2.4. MF Classifier to Profile Mapping . . . 35
3.2.5. MF Classifier to Global Mapping . . . 35
3.3. Remark-Map Configuration . . . 36
3.3.1. Remark-map Profile Mapping . . . 36
3.3.2. Global Profile Mapping . . . 37
3.3.3. Instance Profile Mapping . . . 37

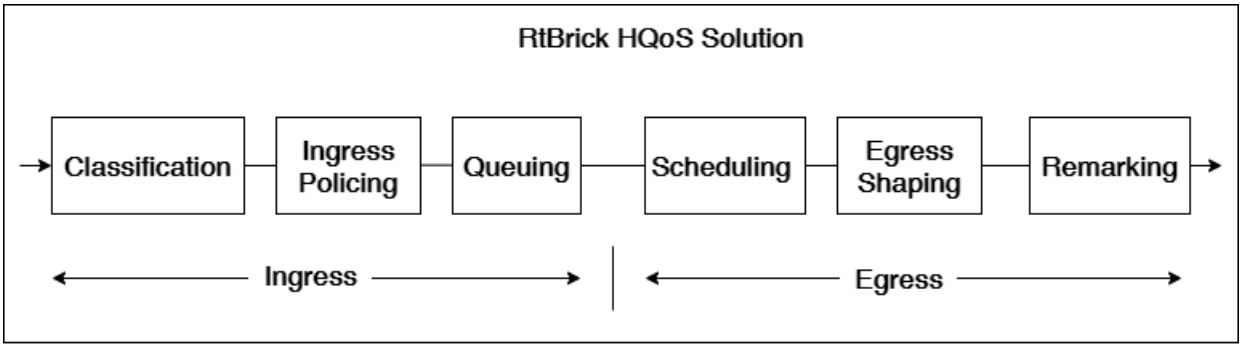
3.4. Policer Configuration . . . 37
3.5. Class Policer-Map Configuration . . . 40
3.6. Queue Configuration . . . 40
3.7. Class Queue-Map Configuration . . . 42
3.7.1. Class to Queue mapping . . . 42
3.8. Queue-Group Configuration . . . 43
3.9. Scheduler Configuration . . . 44
3.10. Scheduler-Map Configuration . . . 45
3.10.1. Scheduler to Port Configuration . . . 46
3.10.2. Scheduler to Scheduler (same Scheduler-Map) . . . 46
3.10.3. Scheduler to Scheduler (different Scheduler-Map) . . . 47
3.10.4. Queue to port . . . 47
3.10.5. Queue to scheduler . . . 47
3.11. Shaper Configuration . . . 48
3.12. Priority Propagation . . . 49
3.13. Profiles Configuration . . . 50
3.14. Interface Configuration . . . 53
3.14.1. Logical Interface QoS Profile . . . 53
3.14.2. Physical Interface Scheduler Map . . . 54
3.14.3. Physical Interface Shaper . . . 54
3.14.3.1. Physical Interface Header Compensation . . . 54
3.14.3.1.1. Ingress Header Compensation . . . 54
3.14.3.1.2. Egress Header Compensation . . . 55
4. HQoS Show Running-Configuration . . . 56
5. HQoS Show Commands . . . 62
5.1. Display Policer Counters . . . 62
5.2. Display Queue Counters . . . 63

1. Introduction to Hierarchical Quality of Service (HQoS)

Hierarchical Quality of Service (HQoS) is a technology that allows you to specify QoS (Quality of Service) behavior at multiple policy levels. It provides a high degree of granularity in traffic management. It can ensure that each network service gets the network resources it needs. This is achieved by classifying, policing, shaping, and scheduling the traffic based on service types. For example, in a simple QoS, you can differentiate between services (such as voice and video), but using H-QoS, you can apply QoS policies to different users, VLANs, maybe logical interfaces and so on.

The RtBrick Full Stack (RBFS) uses the following HQoS mechanisms:

- **Classifier:** Classifies each incoming packet as belonging to a specific class, based on packet contents. Supported classifiers are: Behavior Aggregate (BA) and Multifield (MF). In BA classifier, packets are classified according to the CoS field: IEEE 802.1p, IPv4/v6 ToS/TC, or MPLS EXP. In MF classifier, packets are classified using additional fields in IP header: source IPv4/IPv6 prefix, destination IPv4/IPv6 prefix, L4 source port, L4 destination port, and/or IP protocol.
- **Queuing:** Drop unqualified packets in advance using the Weighted Random Early Detection (WRED) technology in the case of congestion to ensure bandwidth for qualified services. This is performed at the egress.
- **Scheduler:** Manage traffic on a device using different algorithms for queue scheduling. Such algorithms include Fair Queuing (FQ), Weighted Round Robin (WRR), and Strict Priority (SP).
- **Policer:** Policer is implemented in the ingress to drop the unwanted traffic. Policer supports Committed Information Rate (CIR), the Committed Burst Size (CBS), Peak Information Rate (PIR), and Peak Burst Size (PBS). Drop behavior is to either mark traffic as green, yellow, or drop.
- **Shaper:** Shaper is implemented in egress to rate-limit the traffic.
- **Remarking:** Remarking allows you to rewrite the outgoing packet's codepoint. Remarking can be performed in the ingress or the egress side of the hardware pipeline.



1.1. MPLS HQoS

The MPLS HQoS has both UNIFORM and PIPE modes. These modes provides the following functionality:

- During MPLS Encapsulation, MPLS Mode is UNIFORM. MSB 3-bits from 8-bits IPv4-ToS or IPv6-TC are copied to the EXP bits of the newly added MPLS header(s).
- During MPLS Decapsulation, MPLS Mode is PIPE. 8-bits IPv4-ToS or IPv6-TC will be retained and hence it provides ToS/TC codepoint transparency.

For the Uniform MPLS mode mapping between IPv4-ToS or IPv6-TC to MPLS-EXP see the table below:

IPv4-TOS / IPv6-TC	EXP	DSCP
0-31	0	0-7
32-63	1	8-15
64-95	2	16-23
96-127	3	24-31
128-159	4	32-39
160-191	5	40-47
192-223	6	48-55
224-255	7	56-63

1.1.1. Components of the RBFS QoS Solution

The figure below shows the primary configuration components of RBFS QoS solution.

1.2. Supported Hardware

- Edgecore AS5916-XKS

2. HQoS Features

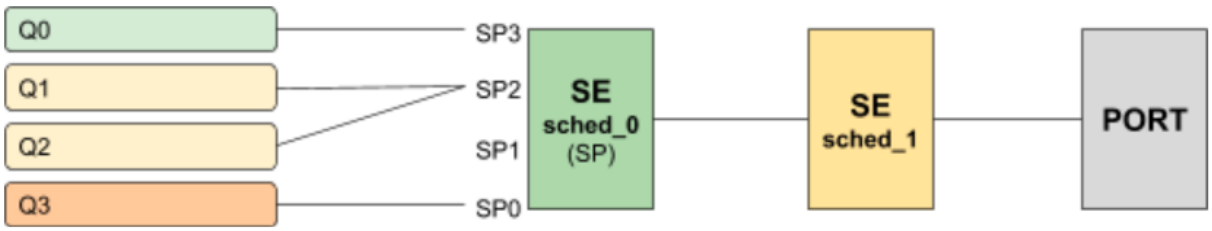
This chapter explains the following topics:

- Priority Propagation
- Behavior Aggregate (BA) Classifier
- Multifield (MF) Classifier
- Remarking
- Policer
  - Class-Policer-Map
- Queueing
  - Class-Queue-Map
  - Queue-Group
- Scheduler
- Scheduler-Map
- Shaper
- Profiles
- Multi-level H-QoS : Level-1 to Level-5

2.1. Priority Propagation

Hierarchical QOS (HQoS) on RBFS is implemented by connecting or chaining queues to scheduler elements (Q —> SE), scheduler elements to each other (SE —> SE) and scheduler elements to ports (SE —> PORT). Each scheduler element can have different child connection points based on types described in section Scheduler.

This means that sched_0 in the example below is not scheduling between the attached queues, but between the different child connection points SP0 to SP3. The scheduler element sched_0 cannot differentiate between Q1 and Q2 in this example because both are connected to SP2.



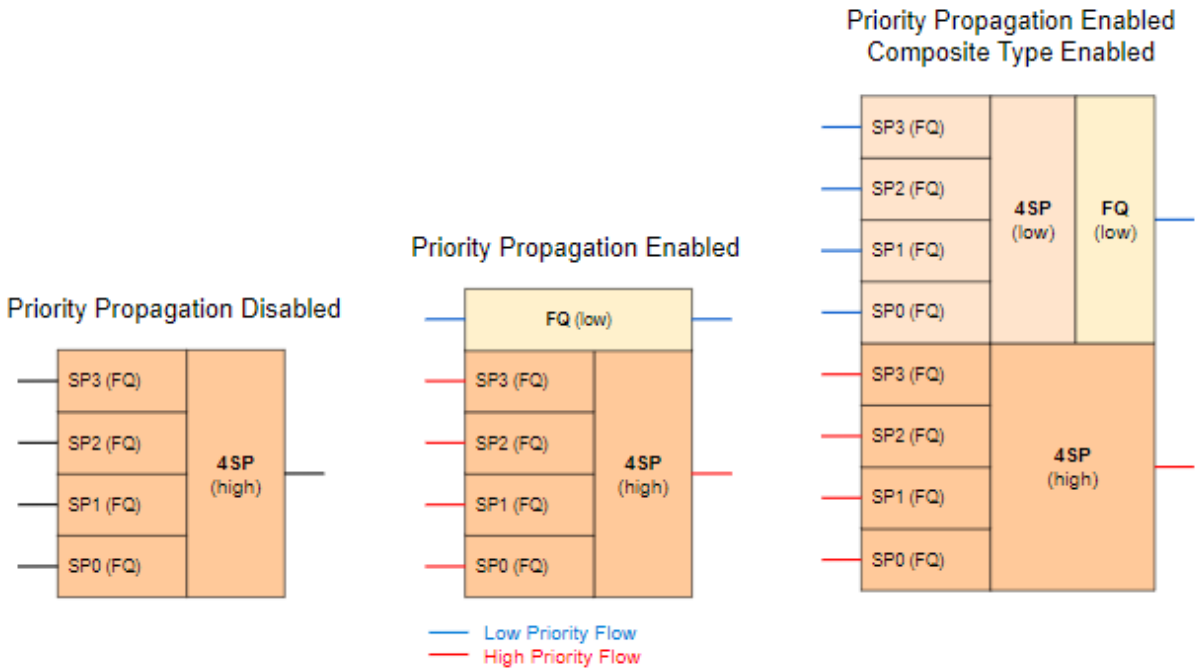
Without priority propagation each scheduler element can have multiple child connection points but just one parent connection point. Therefore traffic leaving a

scheduler element can't be differentiated by the parent scheduling element. The parent scheduler element sched_1 receives the traffic from sched_0 on the selected child connection point. As already mentioned scheduling within a scheduler element happens between child connection points. Second, a scheduler element has only one parent connection point which can be connected to a child connection point of another scheduler element (output of sched_0 input of sched_1). This results into the situation that all traffic from this SE is handled equally regardless of the queue. This may lead into the dropped priority traffic like voice or control traffic in case of congestion in parent elements. For example, if sched_1 has a shaping rate lower than the one of sched_0, it will drop traffic unaware of its original priority.

This problem is addressed with priority propagation which is enabled by default.

With priority propagation the scheduler elements operate in a dual-flow mode with high and low priority flows. The credits generated from the physical interface will be consumed by all attached high priority flows first and only remaining credits will be available for low priority flows. In this mode an implicit FQ element is created for each scheduler element. All queues assigned to low priority flow will be attached to this element.

An additional composite option of the scheduler element allows also the differentiation between multiple low priority queues if required. This composite type is created implicitly and does not need to be configured.

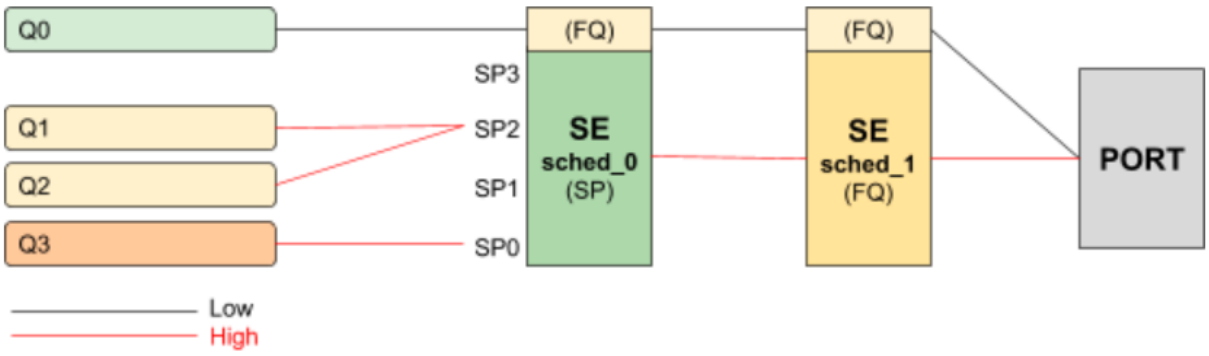


Without priority propagation enabled, each scheduler element consumes only one scheduler resource compared to two elements if enabled. The composite type consumes three scheduler elements.

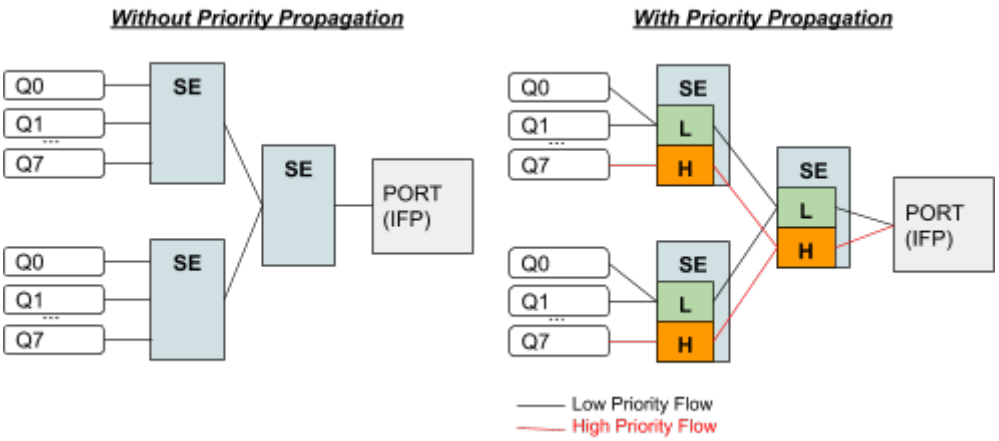
With priority propagation disabled, all traffic is considered as high priority flow.

Now for each queue we can select if connected to high priority or low priority flow where high priority flow is selected per default if not explicitly mentioned.

Assuming the example as before but with priority propagation and Q0 assigned to low priority flow and Q1 - Q3 assigned to high priority flow.



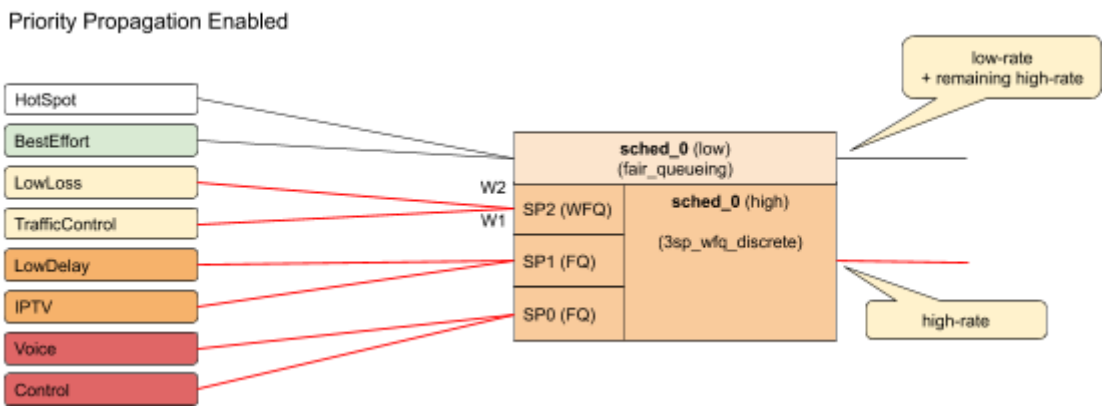
The figure below shows a typical multi level QoS configuration without priority propagation on the left and with priority propagation on the right side.

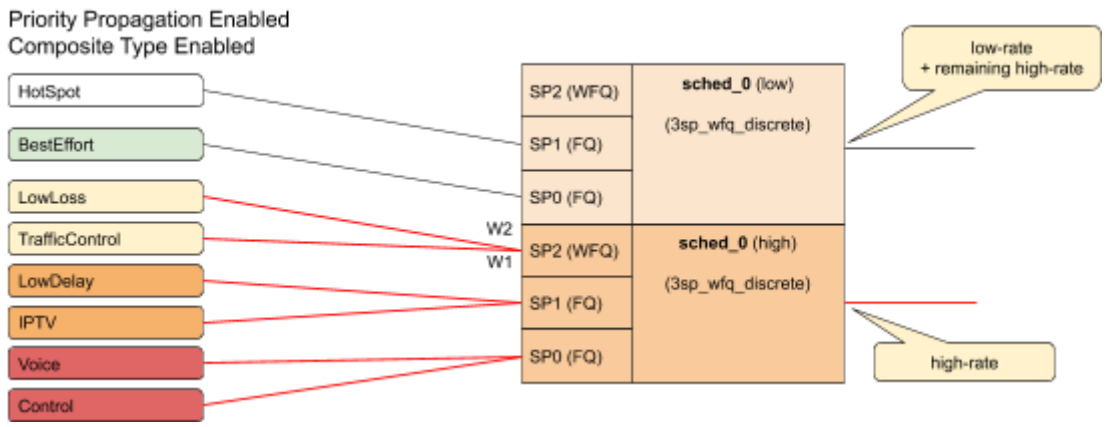


The credits generated from the physical interface will be consumed by high priority flow first and remaining credits will be available for low priority flow. The high flow traffic at any one element is scheduled based on type and connection point. Between schedulers it depends on how they are connected to the parent scheduling element. Per default all levels there is FQ for low and FQ for high priority flows. The port scheduler is also FQ.

In this mode each shaper supports two different rates for low and high priority where the actual shaper rate is the sum of low and high priority rate. If low priority rate is zero, this flow is only served if high priority flow is not consuming all credits. An example might be a high rate of 9Mbps and low rate of 1m which results in max 10Mbps for low priority flow if high priority flow is not consuming any packets but at least 1m is ensured.

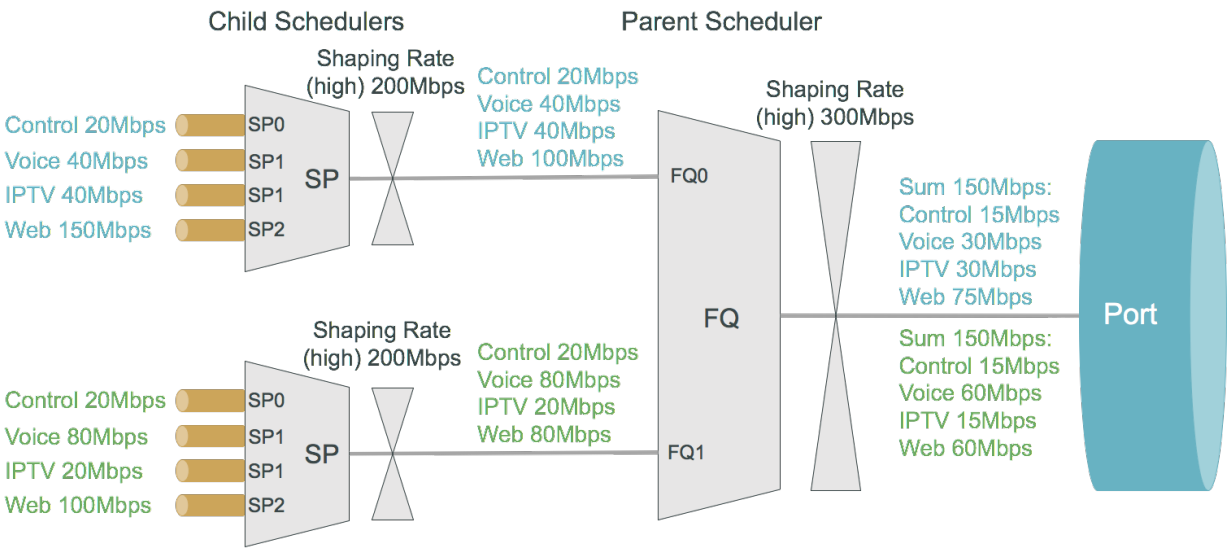
The following example shows a typical access service provider configuration with priority propagation enabled with and without composite type.



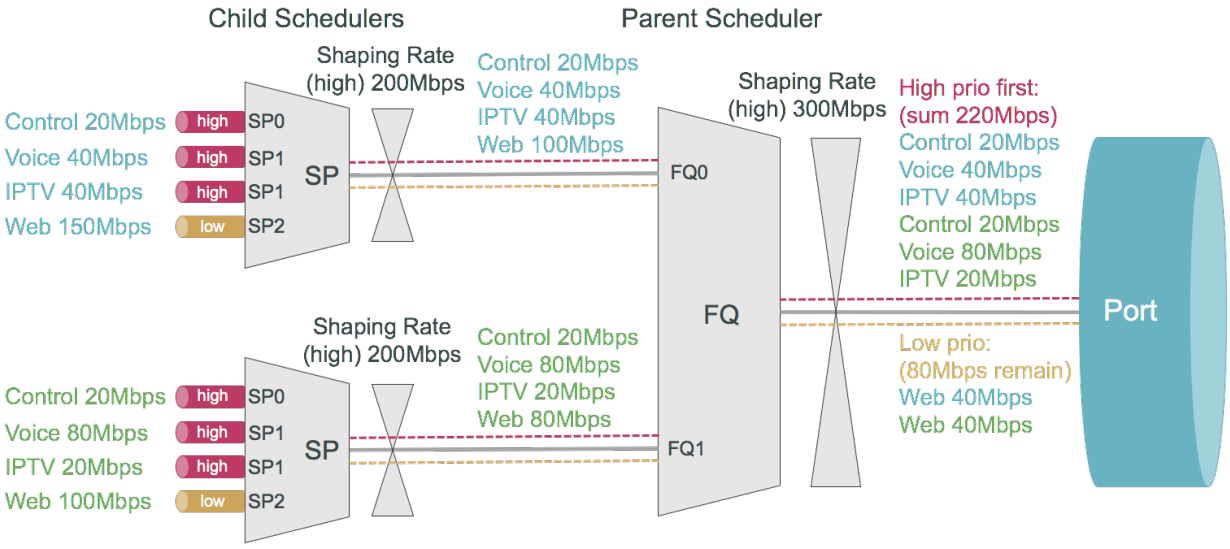


2.1.1. Simple Priority Propagation Scheduling Example

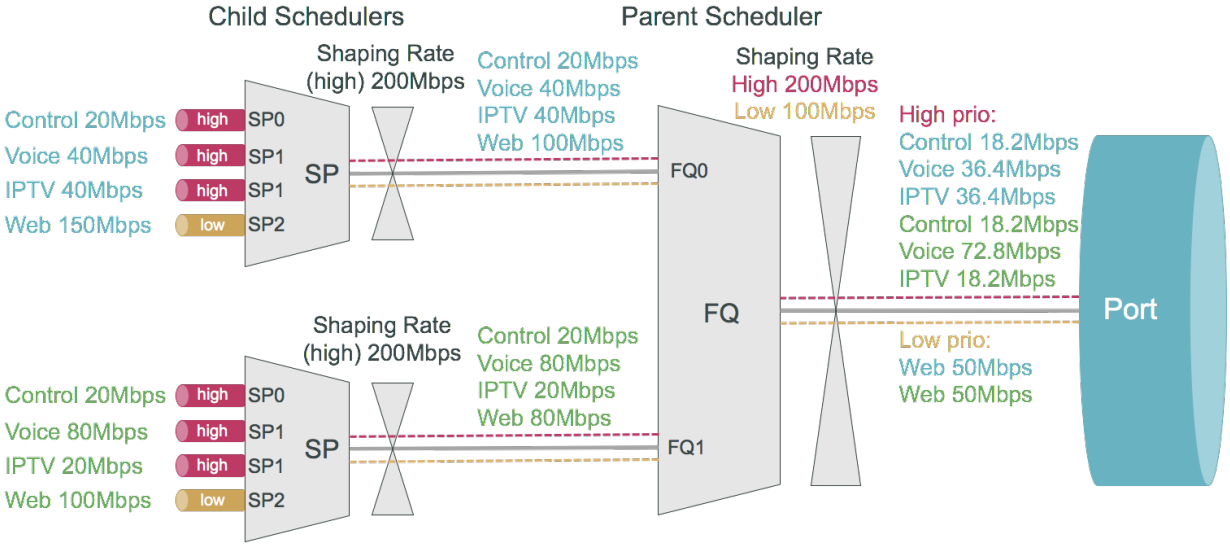
Without priority propagation, the parent scheduler drops traffic equally from all classes as it is unaware of priorities:



With priority propagation, the parent scheduler serves high priority flows first as shown in the figure below:



With priority propagation and dual-flow shaping, the parent scheduler serves high priority flows first up to the high flow shaping rate:



2.2. Behavior Aggregate (BA) Classifier

Classifiers assign the class to which a packet belongs. BA classification is performed on the ingress and maps incoming packet codepoint to a predefined class. BA Classification relies upon markings (that is, codepoint) placed in the headers of incoming packets:

- IEEE 802.1p: Priority - 3 bits
- IPv4: Type of Service byte (ToS) - 8 bits.
- IPv6: Traffic Class (TC) - 8 bits.
- MPLS: Experimental bits (EXP) - 3 bits.

- IEEE 802.1p and IPv4/IPv6 classifiers are applied on either Subscriber IFL or L3 IFL by attaching the classifier to a profile.
- MPLS Exp classifiers are applied either globally or per-instance (to support multiple VPN marking schemes) by attaching the classifier globally or to an instance.

Classifier configuration has the following guidelines and limitations:

- For IPv4: Only ToS based classification is possible. DSCP based classification is not possible.
- For IPv6: Only TC based classification is possible. DSCP based classification is not possible.
- For EXP classification, RBFS uses the uniform mode to copy MSB 3-btis from DSCP to EXP field at the time of MPLS encapsulation at the remote box.
- IPv4/IPv6 Classifiers do not match on labelled traffic. MPLS Classifier is required for the same.

- Default class for Queueing is *class-0*. If classification is not configured for an incoming packet's codepoint, the packet will be classified as *class-0*.
- RBFS supports 8 **classes**: *class-0* to *class-7*.

2.3. Multifield (MF) Classifier

Multifield (MF) classifiers assign the class to which a packet belongs based on multiple fields. Unlike the BA classifier where only CoS fields are used for classification, MF classifier additionally uses the following fields:

- **class**: traffic class of the packet (class-0 to class-7) set by prior BA classifier
- **source prefix**: source IPv4 or IPv6 prefix
- **destination prefix**: destination IPv4 or IPv6 prefix
- **protocol**: UDP or TCP
- **source port**: UDP or TCP source port
- **destination port**: UDP or TCP destination port
- **qos markings**: IPv4 TOS or IPv6 TC header value

The only action supported by an MF classifier is the assignment of a class.

The multifield classifiers can be bound globally (global.qos.global.config) or via QoS profile (global.qos.profile.config). The multifield classifier is processed after BA

classification which allows it to match on a selected class from BA classification or to change the assigned class by more granular match conditions.

Both classification stages (BA and MF) are optional, they can be combined together or used alone which are controlled by configurations.

The global multifield classifier applies to all traffic from any instance or interface.

The multifield classifier assigned via QoS profile applies only to ingress traffic received on the interface where profile is bound to it.

You can bind Multifield classifiers to MPLS core interfaces. Therefore downstream traffic (from core to subscriber) should be classified via global multifield classifier, while upstream traffic (from subscriber to core) can be classified via multifield classifier from QoS profile which is instantiated per subscriber with an implicit match on ingress logical interface (InLIF).

- RBFS supports 8 **classes**: *class-0* to *class-7*.
- The default class for Queue or Policer is **class-0**. If there is no multifield classification configured for an incoming packet, the packet will be classified as class-0.
- For MPLS traffic, MF classifier rule-name is mapped globally: rtb confd set forwarding-options class-of-service global multifield-classifier-name <mf_classifier_name>
- Per instance MF classifier for MPLS traffic is not supported in RBFS because of hardware limitations.

2.4. Remarking

The packet markers set the codepoint in a packet to a particular value, adding the marked packet to a particular behavior aggregate. When the marker changes the codepoint in a packet, it "remarks" the packet. The codepoint in a packet can be IPv4-ToS, IPv6-TC, MPLS-EXP, or IEEE 802.1p field.

In RBFS remarking can be performed at the ingress or egress:

- Ingress remarking is achieved by configuring the **remark-codepoint** field in the Classifier. Ingress remarking rewrites the IPv4-ToS or IPv6-TC at the ingress side with configured remark-codepoint. The configured remark-codepoint can be modified again at the egress side using remark-map. The ingress remarking is supported for IPv4, IPv6, and IEEE 802.1p BA classifiers.
- Egress remarking is achieved by configuring the remark-map. Remark Map is the mapping of **match-codepoint** and **color** to **remark-codepoint**. Egress remarking helps to remark the IPv4-ToS / IPv6-TC field in the IP header, or to write the EXP field in the MPLS label(s), or to write the IEEE 802.1p field in the VLAN header.

- **IPv4-ToS, IPv6-TC, or MPLS-EXP remarking**: If the remark-codepoint is not configured in the Classifier, match-codepoint in the remark-map is the ToS/TC value of the incoming IP packet. If the remark-codepoint is configured in the Classifier, match-codepoint is the same value as the remark-codepoint in Classifier.
- **IEEE 802.1p VLAN remarking**: In the current version of RBFS, match-codepoint in the VLAN remark-map is the Class derived at the ingress using Classifier.
- **Color** is used to set different remark-codepoint for same match-codepoint based on color marked by the Policer (i.e. green or yellow).

> - Color is a mandatory field in remark-map. To set the same remark-codepoint for a match-codepoint irrespective of color, we have to set color as “all”.
> - In tunnel termination cases the remark-codepoint in Classifier is of no use. Therefore the match_codepoint in remark-map shall be the ToS/TC value of the incoming IP packet.
> - For VLAN: Only class to IEEE 802.1p remarking is supported.
> - For IPv4: Only ToS based remarking is possible. DSCP based remarking is not possible.
> - For IPv6: Only TC based remarking is possible. DSCP based remarking is not possible.

Remarking is supported for:

- IEEE 802.1p : Priority - 3 bits.
- IPv4: Type of Service byte (ToS) - 8 bits.
- IPv6: Traffic Class (TC) - 8 bits.
- MPLS Over IPv4: MPLS Experimental bits (EXP) - 3 bits.
- MPLS Over IPv6: MPLS Experimental bits (EXP) - 3 bits.

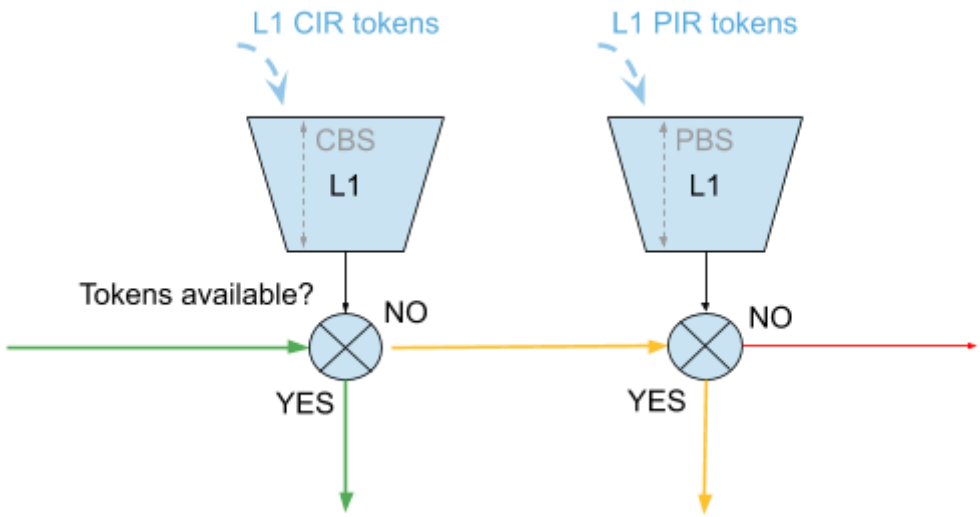
IPv4/v6 and IEEE 802.1p remark-map are applied on an interface - subscriber-ifl or l3ifl using Profile Name.

MPLS-Over-IPv4/v6 remark-map is applied either globally or per-instance (to support multiple VPN marking schemes) using Remark-Map Name.

2.5. Policer

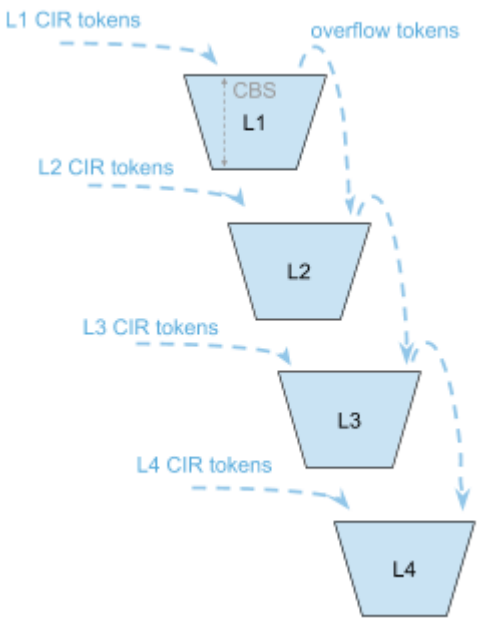
Policer defines the rate at which certain applications can access the hardware resource. So as to rate-limit the traffic from an application, policer hard-drops the unwanted packets in the ingress side.

In RBFS, policers support "**two-rate, three-color**" type in a 4-levels cascaded mode. This means that each policer level has two rates (CIR and PIR) and three colors (green, yellow and red) with two token buckets as shown below.



This means that traffic below CIR is marked green. Traffic above CIR but below PIR is yellow and above PIR is red. Traffic marked red will be dropped. Traffic marked yellow can be demoted by changing ToS, TC, or EXP using remark-map.

In 4 level cascade mode, unused tokens can be passed from higher priority levels to lower priorities where level 1 has highest and level 4 has the lowest priority as shown in the figure below.



Therefore a lower level configured with CIR 0 can still serve traffic if higher priority levels are not consuming all available tokens.

The available tokens per level are calculated by remaining CIR credits from upper

levels and additional credits based on configured CIR per level. Per default the resulting tokens are not limited. The optional max CIR rate attribute allows to limit the sum of tokens from CIR and upper levels. Let's assume level 1 and 2 are both configured with a CIR of 2m. Without max CIR (default behaviour) level 2 can reach up to 4m (level 1 CIR plus level 2 CIR). This can be limited by max CIR (for example, 3m). Obviously max CIR is not relevant for level 1.

Example

	CIR	RX	TX	CIR	RX	TX
L1	4m	1m	1m	4m	1m	1m
L2	6m	20m	9m	6m / max CIR 8	20m	8m
L3	0m	20m	0m	0m	20m	1m
L4	0m	20m	0m	0m	20m	0m
SUM	10m	61m	10m	10m	61m	10m

In the columns 2 through 4 of the preceding example table, L1 consumes only 1m of the available 4m and passes the remaining 3m to L2 which adds additional 6m based on their own configured CIR resulting in 9m.

In the columns 5 through 7 of the preceding example table, L1 consumes only 1m of the available 4m and passes the remaining 3m to L2 which adds additional 6m based on their own configured CIR resulting in 9m. But because of the CIR limit set to 8m, only 8m of 9m can be used at this level. The remaining 1m is now passed to L3 which does not add additional CIR based credits. In both examples L4 would be able to reach up to 10m if upper levels are not consuming credits.

RADIUS Controlled Dynamic Policer

The RBFS RADIUS services support dynamic policer rate updates. The dynamic policer rate when configured affects only the QoS instance of the corresponding subscriber but not other subscribers.

2.5.1. Class-Policer-Map

Since RBFS supports up to 8 classes but only 4 policer levels, there is a need to map multiple classes to the same policer level. A *class-policer-map* defines such mappings. Using class-policer-map configuration, one can map any class to any supported policer level (that includes mapping multiple or all classes to the same level). Similar to policer, a class-policer-map is attached to a profile.

If class to level mapping is not configured, no policing will be applied to the traffic for that class.

2.6. Queueing

Queuing helps to drop the unwanted traffic in advance at the ingress side in case of congestion. This is to ensure bandwidth for qualified services.

RBFS supports the following queueing techniques:

- Tail Drop (TD): This is a conventional congestion avoidance technique. When the network is congested, drop subsequent packets from the queue.
- Weighted Random Early Detection (WRED): This technique requires configuring “Minimum Threshold”, “Maximum Threshold” and “Drop Probability”, which define the start and end range where packets may get discarded. When the average queue size is below the min threshold, no packets will be discarded. The drop_probability parameter can be used to specify the drop probability at the max threshold. When the average queue size is between the min and max threshold, the drop probability increases linearly from zero percent (at the min threshold) to drop_probability percent (at the max threshold). When the average queue size is greater than the max threshold, all packets are discarded.
  - When the average queue size is less than the “Minimum Threshold”, no packets will be discarded.
  - When the average queue size is greater than the “Maximum Threshold”, all packets are discarded.
  - When the average queue size is between “Minimum Threshold” and “Maximum Threshold”, the drop probability increases linearly from zero percent (at the min threshold) to drop probability (at the max threshold).

- Default queue within a queue group is the one mapped to *class-0*. If classification is not configured for an incoming packet’s codepoint, the packet will be classified as *class-0*. Thus will be mapped to queue mapped to *class-0* in *Queue Classifier*. For more information, see [queue_classifier].
- Maximum supported Queue size depends upon DRAM/OCB memory. Since OCB is external memory, hardware does not limit the size that can be configured per Queues.

2.6.1. Class-Queue-Map

A class-queue-map defines the mapping of classes and queues. Class Queue Map is attached to a profile.

- You cannot map two classes to the same queue. The class to queue mapping is 1:1.
- If a queue group is created with four queues, only class-0 to class-3 can be mapped to the queues in class-queue-map, that is, class-4 to class-7 cannot be used.

2.6.2. Queue-Group

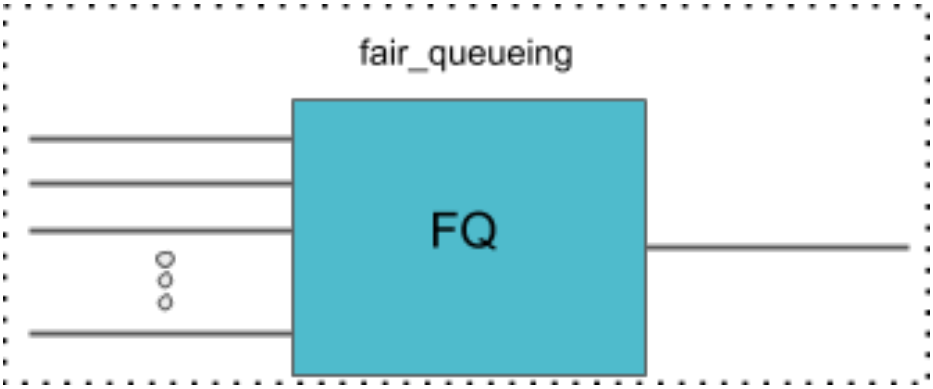
A Queue Group defines the Queue bundle. A Queue Group contains bundle of either 4 or 8 queues.

2.7. Scheduler

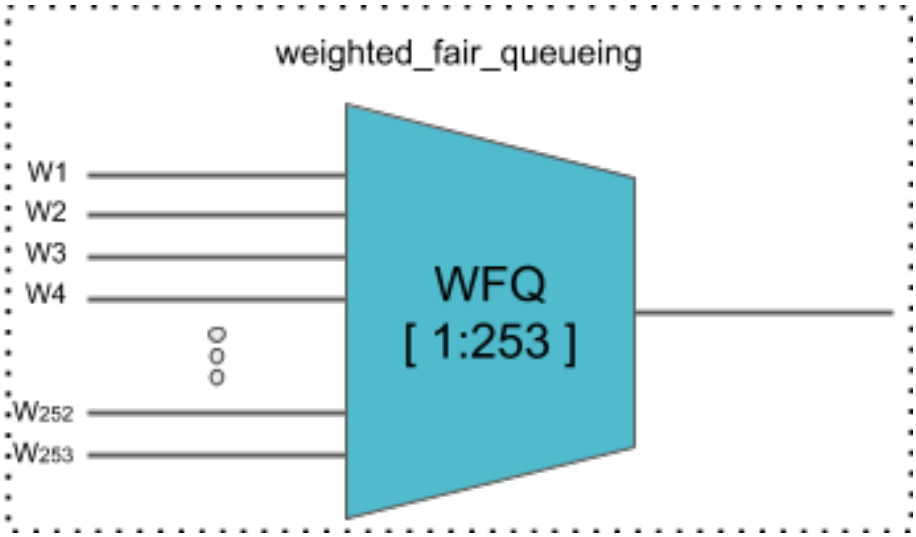
A scheduler configuration defines scheduler parameters such as type and shaping rate. The shaping rate defined for a scheduler applies to queue(s) associated with it.

The following scheduler types are supported:

- **Fair Queueing (FQ)**: Uses round-robin approach to select the next packet to service. This method ensures that all the flows are serviced equally. Configure scheduler type as *fair_queueing* to create FQ scheduler.



- **Weighted Fair Queueing (WFQ)**: Uses round-robin approach but with no guarantee of flow being serviced equally (like in FQ). The rotation of the next packet to service is based on the weight that is assigned to each flow. Configure scheduler type as *weighted_fair_queueing* to create WFQ scheduler.
  - Supported weight: 1 to 253



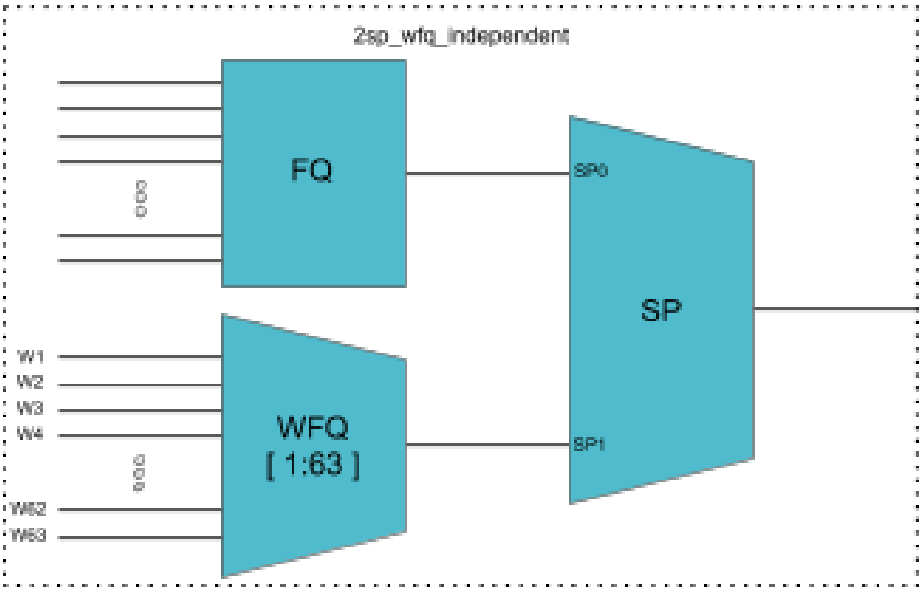
In any WFQ scheduler the lower the weight, the higher the bandwidth portion is awarded.

- **Strict Priority (SP)**: Uses priority based approach to service the flow. SP schedulers are supported in “hybrid” mode only. Hybrid mode combines FQ-WFQ schedulers using strict priority.

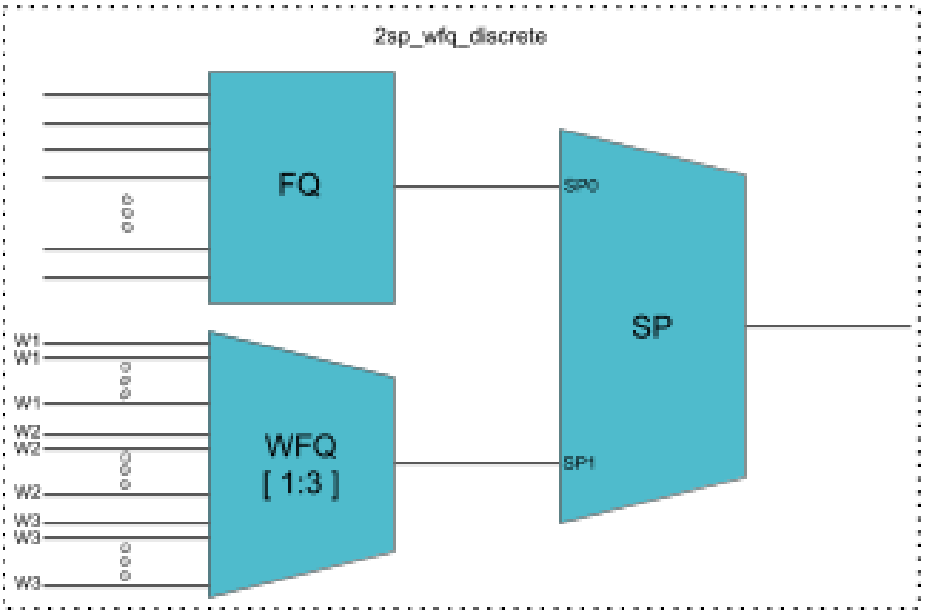
The priority order for SP is: **strict_priority_0 > strict_priority_1 > strict_priority_2 > strict_priority_3** (where **strict_priority_0** being highest priority and **strict_priority_3** being lowest)

The following SP scheduler types are supported:

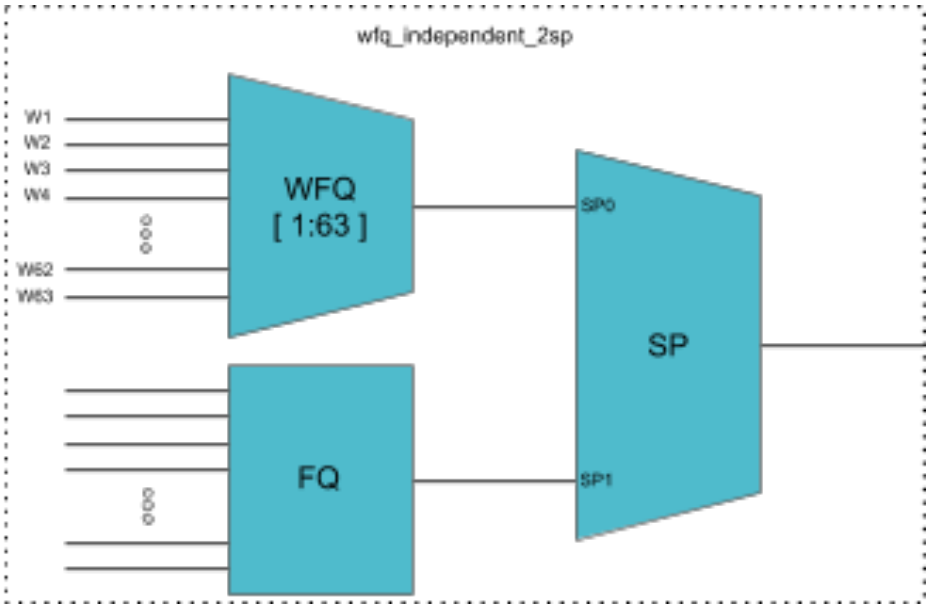
- **2 Strict Priority (2SP)**: Uses SP between 1-FQ and 1-WFQ. There are following types of 2SP hybrid schedulers:
  - type **“2sp_wfq_independent”**
    - Supported weight: 1 to 63



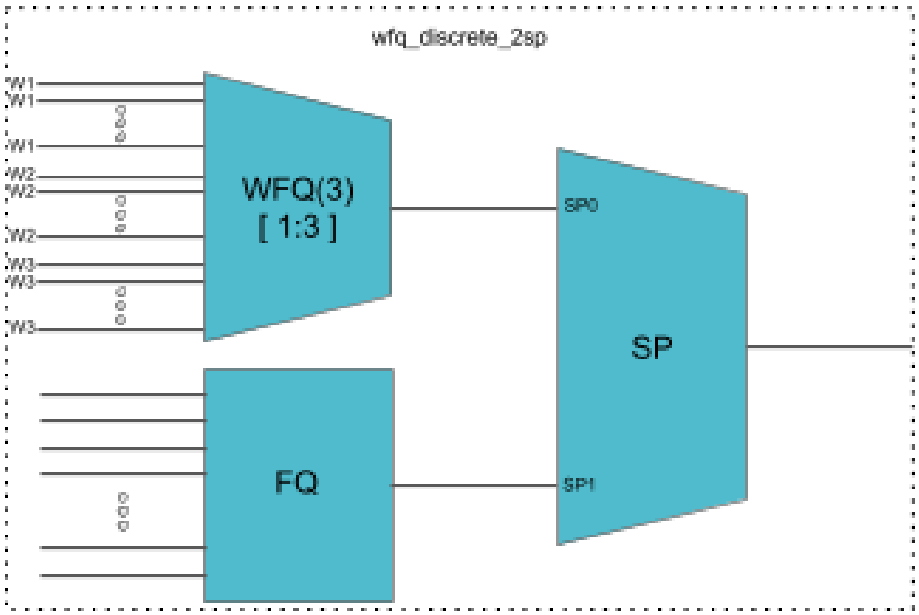
- type **“2sp_wfq_discrete”**
  - Supported weight: { 1, 2, 3 }



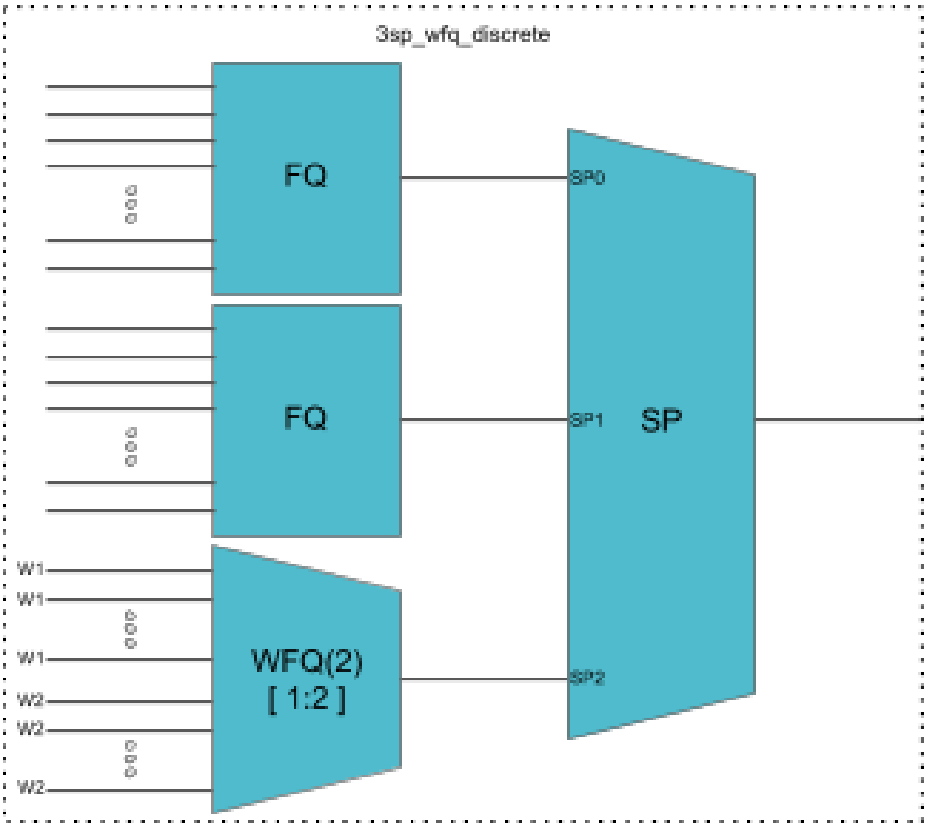
- type **“wfq_independent_2sp”**
  - Supported weight: 1 to 63



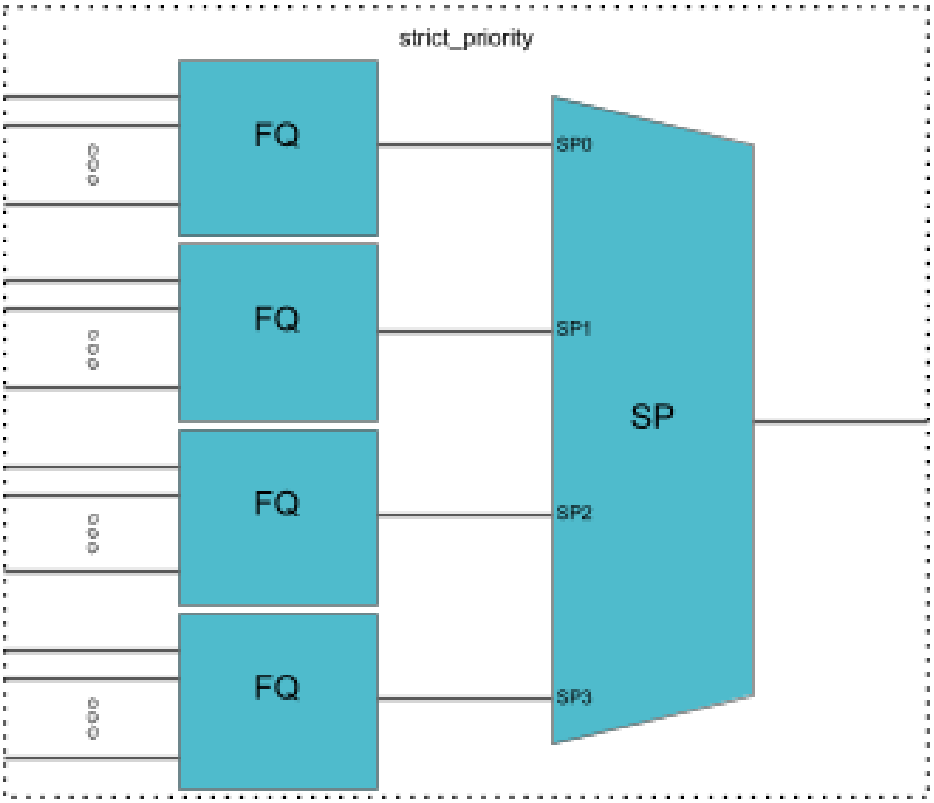
- type **“wfq_discrete_2sp”**
  - Supported weight: { 1, 2, 3 }



- **3 Strict Priority (3SP)**: maps 2-FQs and 1-WFQ
  - type: **“3sp_wfq_discrete“**
    - Supported weight: { 1, 2 }



- **4 Strict Priority (4SP)**: maps 4-FQs using SP
  - type **“strict_priority"**



2.8. Scheduler-Map

Scheduler Map defines the set of relationships between parents and children in egress scheduling hierarchy. Child in a Scheduler Map configuration can be either Queue or Scheduler. Whereas parents in a Scheduler Map configuration can be either Port or Scheduler (from the same Scheduler-Map or physical interface Scheduler-Map).

Connection Point and Weight

Child-queue or child-scheduler in a scheduler map configuration is connected to the parent-scheduler at "**connection point (CP)**". Connection point configuration also has "**weight**" associated with it if the parent has a WFQ scheduler corresponding to that connection point. Valid connection point value for a child to connect to parent **WFQ/FQ** scheduler is **no_priority** and to connect to parent **SP/Hybrid** scheduler is between **strict_priority_0** to **strict_priority_3** (based on number of Strict Priority points in parent scheduler).

Connection Types

There are five connection types in a scheduler map entry:

- queue_to_port
- queue_to_scheduler
- scheduler_to_scheduler
- scheduler_map_to_scheduler_map
- scheduler_to_port

- For the **queue_to_port** connection type, the scheduler has no role.
- **scheduler_map_to_scheduler_map** is used to statically map schedulers from 2 different scheduler-maps (that is, per IFL scheduler-map scheduler to Physical Interface scheduler-map scheduler).

2.9. Shaper

A shaper configuration defines the shaping rate in kilo-bits-per-seconds (kbps).

RADIUS Controlled Dynamic Shapers

RBFS RADIUS services support dynamic shaper updates. The dynamic shaper when configured affects only the QoS instance of the corresponding subscriber but not other subscribers.

2.10. Profiles

A profile configuration defines the QoS profile that is attached to either a Subscriber interface or an L3 interface.

Profile maps the following QoS constructs to a Subscriber or an L3 interface:

- Classifier
- Multifield (MF) Classifier
- Class Policer Map
- Policer
- Class Queue Map
- Scheduler Map
- Remark Map

2.11. Header Compensation

2.11.1. Queue Compensation

The rate at which the packets are dequeued from a queue dependents on the credit received by that queue. The source of the credit received by a queue is the egress port to which the queue is mapped. When a packet is dequeued, the credit balance is decreased by the packet size. But, the packet size that is used must be adjusted to model the packet size at the egress, rather than its actual size at the ingress queue. Thus the header compensation is used to adjust for the differences in header size between ingress-queue and egress-port. RBFS supports static header compensation configuration per queue (in bytes).

2.11.2. Port Compensation

Similar to queue header compensation where header compensation is performed at the per-queue level, RBFS supports the following header compensation at the per-port level:

- **Ingress Header Compensation**: Inline with the header compensation option that we have per-queue, RBFS supports static header compensation configuration at the ingress to be used by the policing. Header compensation changes the effective size of the packet to compensate for change in header size (such as the CRC removal) when considering the packet for policing. Unlike queue, RBFS ingress header compensation configuration is per ingress port (in bytes).
- **Egress Header Compensation**: Inline with the header compensation option that we have per-queue or per-port at the ingress, RBFS supports static header compensation configuration at the egress. The egress header compensation

configuration is per egress port (in bytes).

The supported range for header compensation is -64 to +64 bytes.

2.12. Multi-level H-QoS : Level-1 to Level-5

The following HQoS levels are required to build internet access services like FTTH, FTTC, or FTTB:

Level-1 (IFP)

Physical Interface Shaper.

Level-2 (PON TREE)

Each PON tree is a TDM based shared medium with typically ~2.5 GBit/s (GPON) shared by up to 32 consumers (ONT or DPU).

Level-3 (DPU)

In case of FTTB there is a single DPU with multiple consumers via G.Fast DSL connected which requires an additional hierarchy. This level is not needed for FTTH or FTTC.

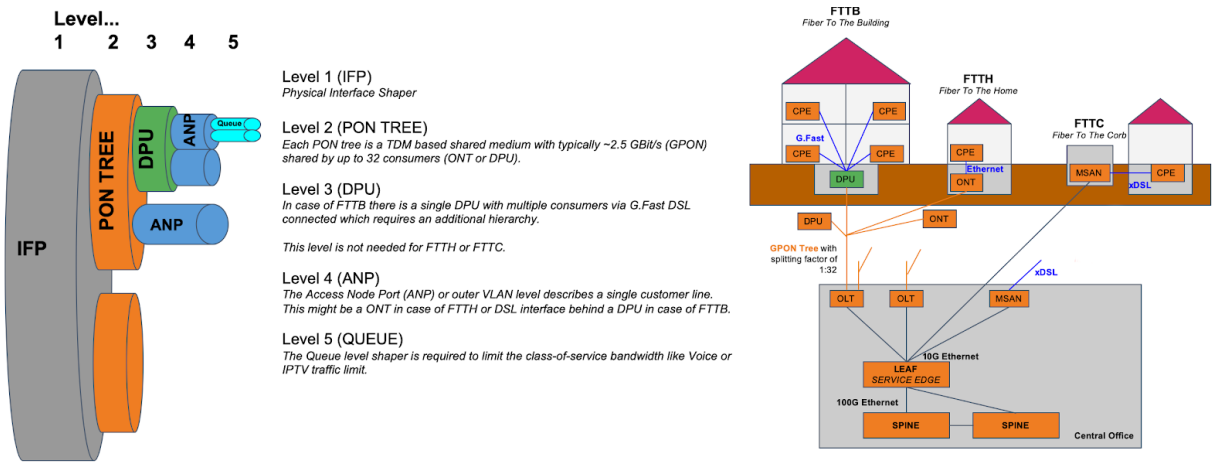
Level-4 (ANP or Session)

The Access Node Port (ANP) or outer VLAN level describes a single customer line. This might be an ONT in case of FTTH or DSL interface behind a DPU in case of FTTB. This level can be also represented on PPPoE sessions as long as just one session is permitted per VLAN.

Level-5 (QUEUE)

The Queue level shaper is required to limit the class-of-service bandwidth like Voice or IPTV traffic limit.

The figure below shows the diagram along with QoS representing Level-1 to Level-5 Hierarchical scheduling.

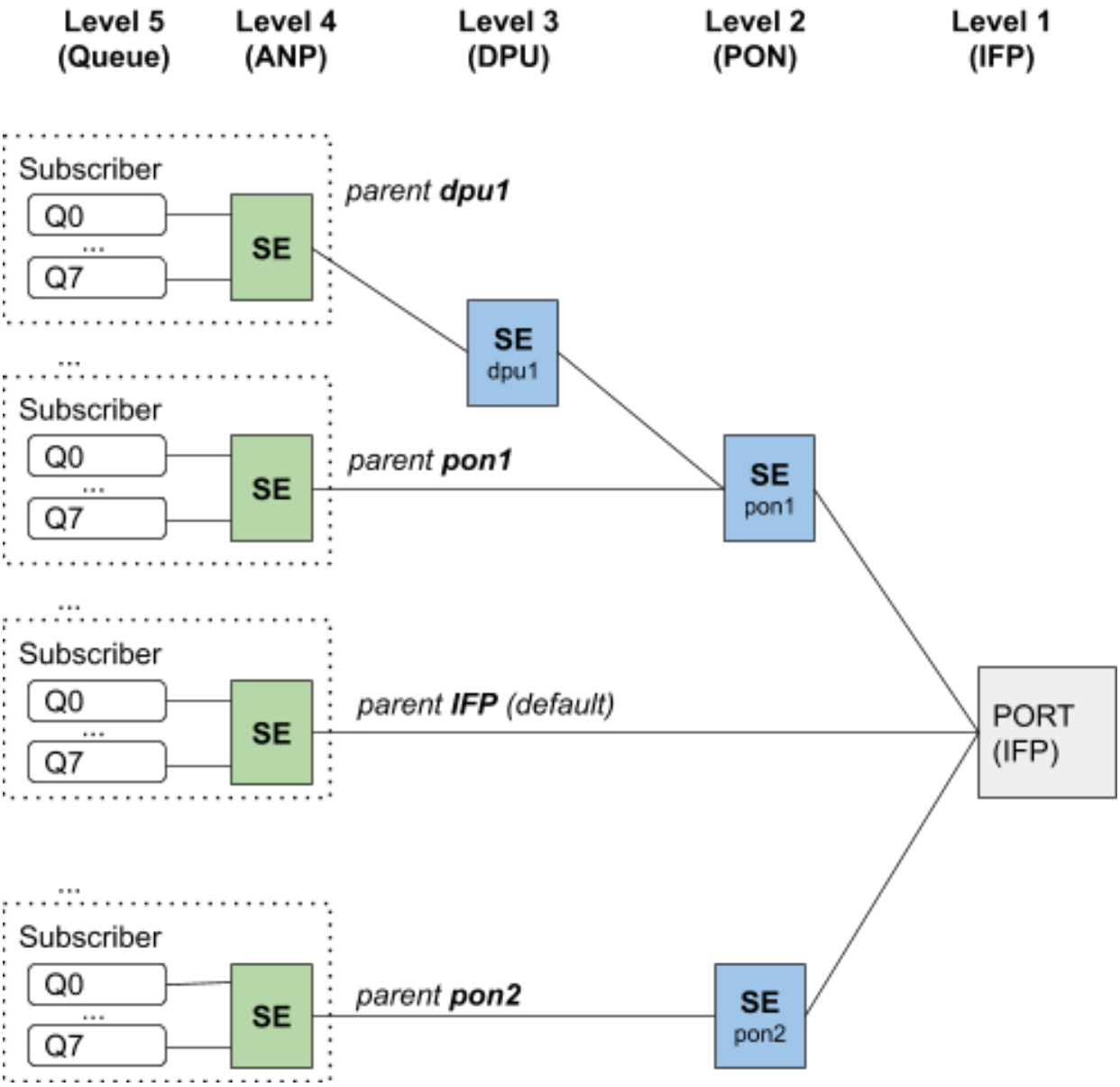


The levels 4 and 5 are configured per logical interface (i.e. subscriber-ifl or l3-ifl). Separate scheduler-map representing levels 1 to 3 connectivity shall be statically configured and mapped to corresponding physical interface (IFP).

Child scheduler in a subscriber scheduler-map is connected to parent scheduler in physical interface scheduler-map in following two ways:

- Statically using CLI.
- Dynamically via RADIUS in case of dynamic subscribers like PPPoE sessions (Subscriber-IFL).

The figure below shows the same details as the preceding figure before with the different levels but from the DPU-PON-IFP scheduler-map point of view.



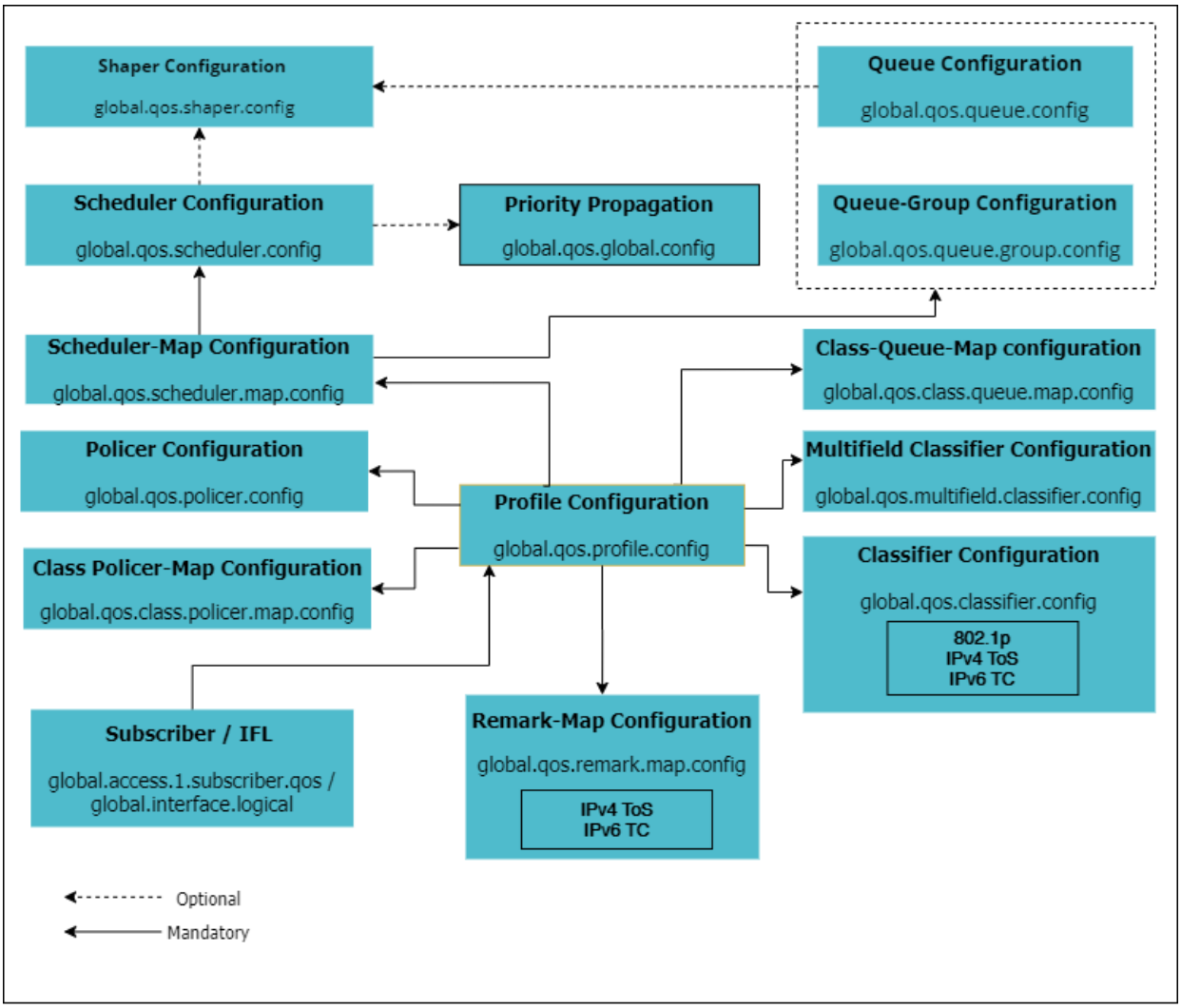
3. Configuring HQoS

To configure HQoS, perform the following steps which include creating a QoS profile and enabling QoS on a PPP Subscriber-Interface or L3-Interface.

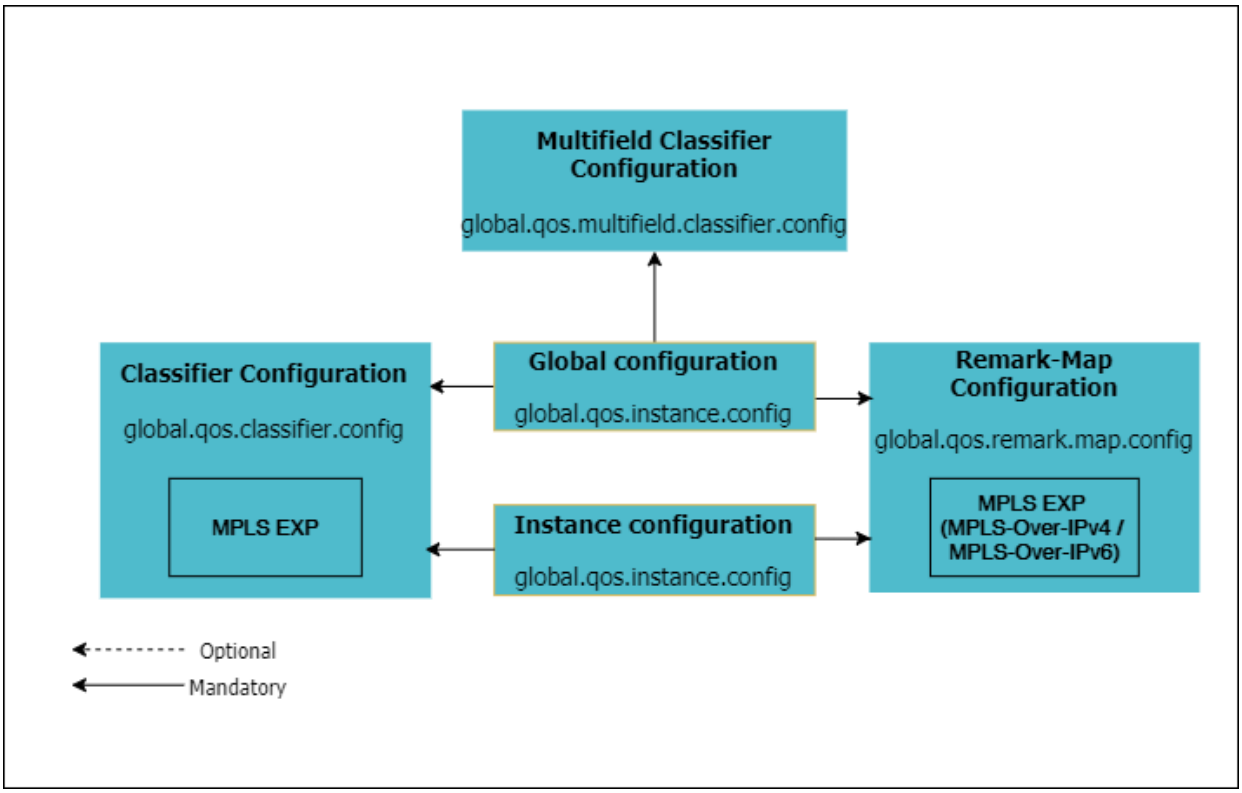
1. Create Behavioral Aggregate (BA) and/or Multifield (MF) classifier to classify the network traffic at the ingress.
2. Create a policer to police the classified traffic at the ingress.
3. Create necessary class-to-policer-map to map the classes to policer-levels (mandatory for policing).
4. Create necessary queues with proper size to queue the classified traffic at the egress.
5. Create queue-groups and configure the queue numbers (4/8) in the queue group.
6. Create necessary class-to-queue-map to map the classes to queues (mandatory for queuing).
7. Specify scheduler(s) with type as required.
8. (Optional) Attach a shaper to queue(s) and/or scheduler(s).
9. Specify a scheduler map to define set of relationships between parent (scheduler or port) and child (queue/queue-group or scheduler) at the egress.
10. Define a QoS profile with classifier, multifield-classifier, class-policer-map, policer, class-queue-map, scheduler-map, and remark-map based on user requirements.
11. Specify another scheduler map to represent level-3 to level-5 hierarchy in multi-level HQoS.
12. Map the MPLS EXP classifier either to an instance or configure it as global entity.
13. Map the MPLS-Over-IPv4/IPv6 remark-map either to an instance or configure it as global entity.
14. For downstream MPLS traffic, map the Multifield (MF) classifier as global entity.

> Priority propagation is enabled by default. To disable the Priority Propagation, we recommend doing this at the beginning and not during an active session.

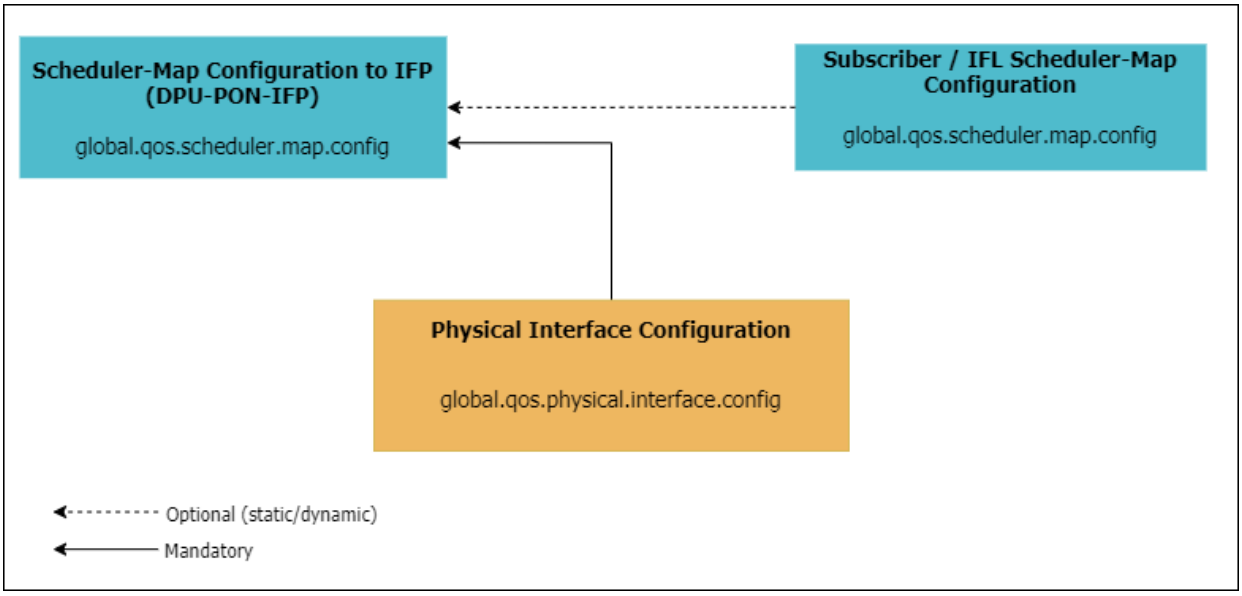
The figure below shows the dependencies between the various HQoS configuration elements.



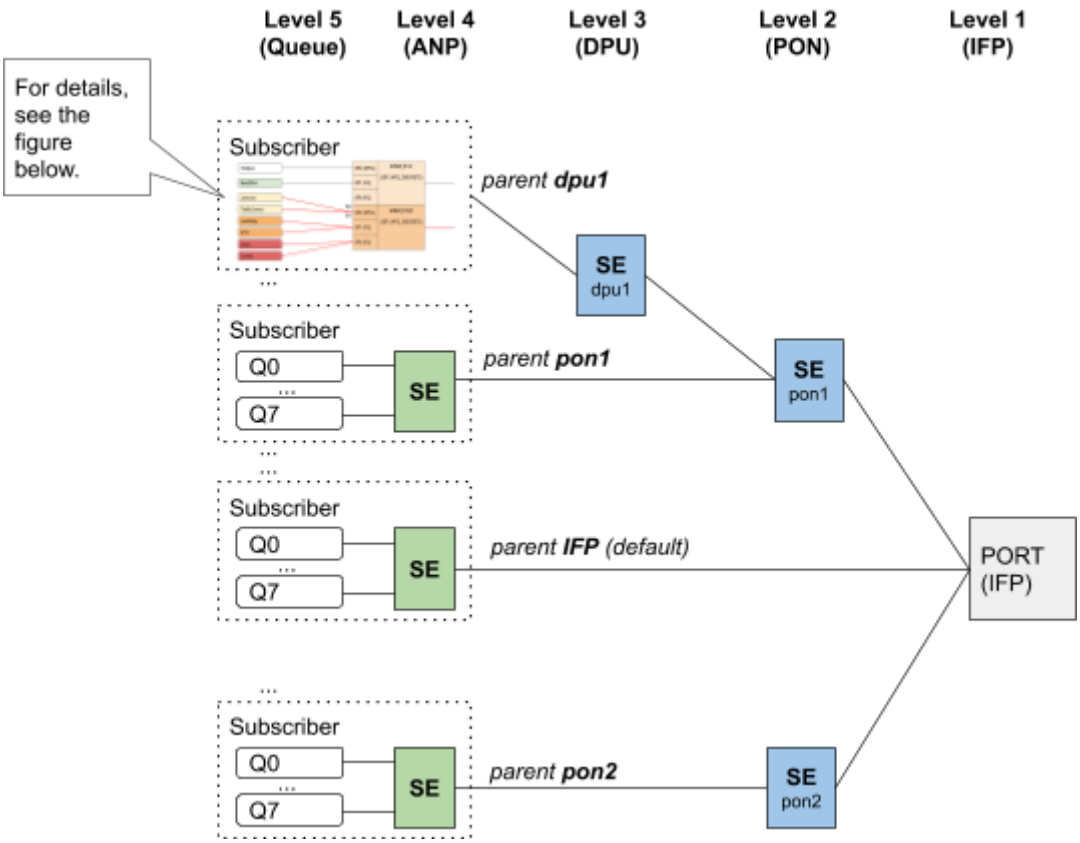
The figure below shows the dependencies for per instance or global classifier and remark-map configurations.

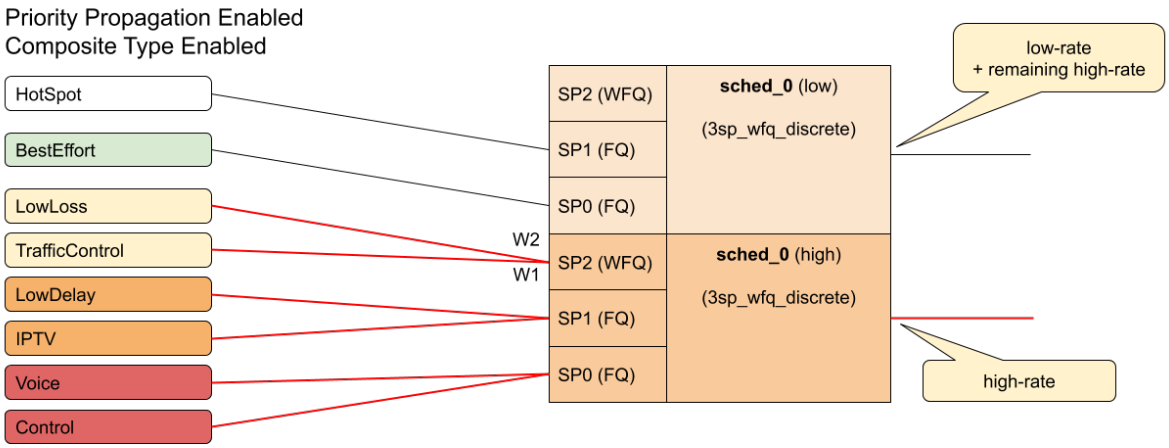


The figure below shows the additional dependencies for Multi-level HQoS.



The figures below show the scheduling hierarchy example.





The following sections provide the commands and examples for configuring HQoS.

- Behavior Aggregate (BA) Classifier Configuration
- Multifield Classifier Configuration
- Remark-Map Configuration
- Policer Configuration
- Class Policer-Map Configuration
- Queue Configuration
- Class Queue-Map Configuration

- Queue-Group Configuration
- Scheduler Configuration
- Scheduler-Map Configuration
- Shaper Configuration
- Profiles Configuration
- Interface Configuration

3.1. Behavior Aggregate (BA) Classifier Configuration

Syntax

[forwarding-options class-of-service]
set classifier classifier-name <classifier_name> **match-type** <match_type> **codepoint** <codepoint> **class** <class> [**remark-codepoint** <remark_codepoint>]

Command arguments

<classifier_name>	Specifies the classifier user-defined name
<match_type>	Specifies the type of traffic to classify, that is, ipv4-tos, ipv6-tc, ieee-802.1, exp
<codepoint>	Specifies the code-point value based on match-type
<class>	Specifies the traffic class as as class-0, class-1, class-2, class-3, class-4, class-5, class-6, and class-7
<remark_codepoint>	Specifies the ingress remarking codepoint

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set classifier classifier-name TC_voice1 match-type ipv6-
tc codepoint 96 class class-1
```

3.1.1. BA Classifier to Profile Mapping

IPv4, IPv6 and IEEE 802.1p BA classifiers are applied on a subscriber-ifl or l3ifl using the Profile Name.

Syntax

[forwarding-options class-of-service]
set profiles profile-name <profile_name> **classifer-name** <classifer_name>

MPLS Exp classifier is applied either globally or per-instance (to support multiple VPN marking schemes) using Classifier Name.

3.1.2. BA Classifier to Global Mapping

The MPLS classifiers can be applied globally using global configuration.

[forwarding-options class-of-service]
set global classifier-name <classifier_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global classifier-name TC_voice1
```

3.1.3. BA Classifier to Instance Mapping

The MPLS classifiers can be applied on an instance using the instance configuration.

[forwarding-options class-of-service]
set instance instance-name <instance_name> **classifier-name** <classifier_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set instance instance-name ip2vrf classifier-name
TC_voice1
```

3.2. Multifield Classifier Configuration

3.2.1. IPv4 Match Configuration

Syntax

[forwarding-options class-of-service]
set forwarding-options class-of-service multifield-classifier acl l3v4 rulename <rule_name> **ordinal** <ordinal_value> **match** [<source-ipv4-prefix> | <destination-ipv4-prefix> | <ip-protocol> | <source-l4-port> | <destination-l4-port> | <ipv4-traffic-class> | <forward-class>]

Command arguments

<rule_name>	Specifies the multifield classifier rule name
<ordinal_value>	Specifies the ordinal value that is used for traffic policy rule referencing
<source-ipv4-prefix>	Specifies the source IPv4 prefix address
<destination-ipv4-prefix>	Specifies the destination IPv4 prefix address
<ip-protocol>	Specifies the IP protocol such as UDP or TCP
<source-l4-port>	Specifies the Layer 4 source port number
<destination-l4-port>	Specifies the Layer 4 destination port number
<ipv4-traffic-class>	Specifies the IPv4 ToS value
<forward-class>	Specifies the forward class name

Example

```
rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtbr_mf1 ordinal 1 match source-ipv4-prefix 152.1.1.0/24

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtbr_mf1 ordinal 1 match destination-ipv4-prefix 200.1.1.0/24

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtbr_mf1 ordinal 1 match ip-protocol TCP

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtbr_mf1 ordinal 1 match ip-protocol UDP

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtbr_mf1 ordinal 1 match source-l4-port 56000

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtb_mf1 ordinal 1 match destination-l4-port 56000

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtb_mf1 ordinal 1 match ipv4-tos 224

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtb_mf1 ordinal 1 match forward-class class-7
```

3.2.2. IPv6 Match Configuration

Syntax

[forwarding-options class-of-service]
rtb confd set forwarding-options class-of-service multifield-classifier acl l3v6 rulename <rule_name> **ordinal** <ordinal_value> **match** [<source-ipv6-prefix> | <destination-ipv6-prefix> | <ip-protocol> | <source-l4-port> | <destination-l4-port> | <ipv6-traffic-class> | <forward-class>]

Command arguments

<rule_name>	Specifies the multifield classifier rule name
<ordinal_value>	Specifies the ordinal value that is used for traffic policy rule referencing
<source-ipv6-prefix>	Specifies the source IPv6 prefix address
<destination-ipv6-prefix>	Specifies the destination IPv6 prefix address
<ip-protocol>	Specifies the IP protocol such as UDP or TCP
<source-l4-port>	Specifies the Layer 4 source port number
<destination-l4-port>	Specifies the Layer 4 destination port number

<ipv6-traffic-class>	Specifies the IPv6 traffic class value
<forward-class>	Specifies the forward class name

Example

```
rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v6 rulename rtbr_mf1 ordinal 1 match destination-ipv6-prefix 200:1:1::0/64

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v6 rulename rtbr_mf1 ordinal 1 match ip-protocol TCP

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v6 rulename rtbr_mf1 ordinal 1 match ip-protocol UDP

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v6 rulename rtbr_mf1 ordinal 1 match source-l4-port 56000

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v6 rulename rtb_mf1 ordinal 1 match destination-l4-port 56000

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v6 rulename rtb_mf1 ordinal 1 match ipv6-traffic-class 224

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v6 rulename rtb_mf1 ordinal 1 match forward-class class-7
```

3.2.3. IPv4/IPv6 Action Configuration

Syntax

[forwarding-options class-of-service]
rtb confd set forwarding-options class-of-service multifield-classifier acl [l3v4|l3v6] rulename <rule_name> **ordinal** <ordinal_value> **action forward-class** <class>

Command arguments

<rule_name>	Specifies the rule name
<class>	class-0, class-1, class-2, class-3, class-4, class-5, class-6, class-7

Example

```
rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v4 rulename  rtb_mf1 ordinal 1 action forward-class class-4

rtb confd set forwarding-options class-of-service multifield-classifier acl
l3v6 rulename  rtb_mf1 ordinal 1 action forward-class class-4
```

3.2.4. MF Classifier to Profile Mapping

Syntax

[forwarding-options class-of-service]
set profiles profile-name <profile_name> **multifield-classifier-name** <multifield-classifer_name>

discrete] ===== Command arguments

<profile_name>	Specifies the profile name
<multifield-classifer_name>	Specifies the multifield classifier name

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set profiles profile-name rtbrick_residential_profile
multifield-classifier-name mf100
```

3.2.5. MF Classifier to Global Mapping

Syntax

[forwarding-options class-of-service]
set global multifield-classifier-name <multifield-classifier_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global multifield-classifier-name mf100
```

3.3. Remark-Map Configuration

Syntax

[forwarding-options class-of-service]
set forwarding-options class-of-service remark-map remark-map-name <remark_map_name> **remark-type** <remark_type> **match-codepoint** <match_codepoint> **color** <color> **remark-codepoint** <remark_codepoint>

Command arguments

<remark_map_name>	Specifies the remaking map name
<match_codepoint>	Specifies the match code point
<color>	Indicates the colour - all, green, yellow. Colour is used to set different remark-codepoint for same match-codepoint based on color marked by the Policer.
<remark_type>	Specifies the remarking type - ipv4-tos, ipv6-tc, mpls-over-ipv4, mpls-over-ipv6, ieee-802.1
<remark_codepoint>	Specifies the remarking codepoint

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set remark-map remark-map-name remark-exp remark-type
ipv6-tc match-codepoint 100 color green remark-codepoint 224
```

3.3.1. Remark-map Profile Mapping

Syntax

[forwarding-options class-of-service]
set profiles profile-name <profile_name> remark-map-name <remark_map_name>

discrete] ===== Command arguments

<profile_name>>	Specifies the profile name
<remark_map_name>	Specifies the remark map name

Example

```
TBD
```

3.3.2. Global Profile Mapping

[forwarding-options class-of-service]
set global remark-map-name <remark_map_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global remark-map-name remark-exp
```

3.3.3. Instance Profile Mapping

[forwarding-options class-of-service]
set instance instance-name <instance_name> **remark-map-name** <remark_map_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set instance instance-name ip2vrf remark-map-name remark-
exp
```

3.4. Policer Configuration

Syntax

[forwarding-options class-of-service]
root@rtbrick:confd> **edit policer policer-name** <policer_name>
[forwarding-options class-of-service policer policer-name voice_policer]

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit policer policer-name voice_policer
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd>
```

Levels configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set levels <levels>

Example

```
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set levels 4
```

Type configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set type <type>

Example

```
[ forwarding-options class-of-service policer policer-name Policer_Upstream ]
root@rtbrick:confd> set type two-rate-three-color
```

Flag configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set flags <flag>

Example

```
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set flags color-blind
```

Level rates configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set level1-rates cir <level1_cir> **pir** <level1_pir> **cbs** <level1_cbs> **pbs** <level1_pbs>

The same is applicable for level-2 to level-4.

Example

```
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set level1-rates cir 4000 pir 5000 cbs 400 pbs 600

root@rtbrick:confd> set level2-rates cir 4000 pir 5000 cbs 400 pbs 600
```

Level rates and max-rates configuration

set level1-rates cir <level1_cir> **pir** <level1_pir> **cbs** <level1_cbs> **pbs** <level1_pbs> **max-cir** <level1_max_cir> **max-pir** <level1_max_pir>

The same is applicable for level-2 to level-4.

Example

```
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set level2-rates cir 4000 pir 5000 cbs 400 pbs 600 max-
cir 4500 max-pir 5500
```

Command Arguments

<policer-name>	Specifies the policer name.
<levels>	Specifies levels in the Policer. Levels will be from 1 to 4.
<type>	Specifies the policer type.
<flag>	Set flags.
<level1_cir>	Set committed information rate (CIR) in kilobits per second (kbps) for level-1. The same is applicable for level-2 to level-4.
<level1_pir>	Set peak information rate (PIR) in kilobits per second (kbps) for level-1. The same is applicable for level-2 to level-4.
<level1_cbs>	Set Committed burst size (CBS) in kilobits for level-1. The same is applicable for level-2 to level-4.

<level1_pbs>	Set peak burst size (PBS) in kilobits for level-1. The same is applicable for level-2 to level-4.
<level1_max_cir>	Set the maximum for the level-1 committed information rate (CIR) in kilobits per second (kbps). The same is applicable for level-2 to level-4.
<level1_max_pir>	Set the maximum for the level-1 peak information rate (PIR) in kilobits per second (kbps). The same is applicable for level-2 to level-4.

3.5. Class Policer-Map Configuration

Syntax

[forwarding-options class-of-service]
root@rtbrick:confd> **edit class-policer-map class-policer-map-name** <class_policer_map_name> **class <class> policer-level** <policer_level>

Command arguments

class_policer_map_name>	Specifies the class policer map name,
<class>	Specifies the class such as class-0, class-1, class-2, class-3, class-4, class-5, class-6, class-7
<policer_level>	level-1, level-2, level-3, level-4

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set class-policer-map class-policer-map-name
classifier_residential class class-4 policer-level level-2
[ forwarding-options class-of-service ]
root@rtbrick:confd> set class-policer-map class-policer-map-name
classifier_residential class class-7 policer-level level-4
```

3.6. Queue Configuration

Syntax

[forwarding-options class-of-service]
edit queue queue-name <queue_name>
[forwarding-options class-of-service queue queue-name]

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit queue queue-name IPTV
```

Queue Size

set queue_size <queue_size>

Example

```
[ forwarding-options class-of-service queue queue-name IPTV ]
root@rtbrick:confd> set queue-size 6000
```

Queue WRED Profile

set wred minimum-threshold <minimum_threshold> **maximum-threshold** <maximum_threshold> **drop-probability** <drop_prob>

Example

```
[ forwarding-options class-of-service queue queue-name IPTV ]
root@rtbrick:confd> set wred minimum-threshold 15000 maximum-threshold 18000
drop-probability 70
```

Queue Shaper

set shaper-name <shaper_name>

Example

```
[ forwarding-options class-of-service queue queue-name IPTV ]
root@rtbrick:confd> set shaper-name session_shaper
```

Queue Header Compensation

set header-compensation bytes <header_compensation_bytes> <decrement | increment>

Example

```
rtb confd set forwarding-options class-of-service queue queue-name Control
header-compensation bytes 2 increment
```

Command arguments

<queue_name>	Specifies the user-defined queue name	
<queue_size>	Specifies the size of the queue in bytes	
<shaper_name>	(Optional) Specifies the shaper that is associated with the queue	
WRED	<minimum_threshold>	Specifies the minimum average queue size to apply WRED in bytes
	<maximum_threshold>	Specifies the maximum average queue size to apply WRED in bytes
	<drop_prob>	WRED drop probability applied at the maximum threshold
<header_compensation_bytes>	Specifies the the header compensation value	
<decrement>	decrement specifies that the header compensation value is negative	<increment>

3.7. Class Queue-Map Configuration

3.7.1. Class to Queue mapping

Syntax

[forwarding-options class-of-service]
edit class-queue-map class-queue-map-name <class_queue_map_name>
class <class> **queue_name** <queue_name>

Command arguments

<class_queue_map_name>	Specifies the class queue map name
<class>	Specifies the class such as class-0, class-1, class-2, class-3, class-4, class-5, class-6, class-7

Example

```
[ forwarding-options class-of-service class-queue-map class-queue-map-name
classifier_residential class class-4 ]
root@rtbrick:confd> set queue-name BestEffort
[ forwarding-options class-of-service class-queue-map class-queue-map-name
classifier_residential class class-4 ]
```

3.8. Queue-Group Configuration

Queue group size - 4 or 8

Syntax

[forwarding-options class-of-service]
edit queue-group queue-group-name <queue_group_name>
set queue-numbers <queue_numbers>

Command arguments

<queue_group_name>	User-defined name for the queue-group
<queue_numbers>	Specifies the number of queues in a Queue Group

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit queue-group queue-group-name queue_group_residential
[<Enter>] Execute the command
[ forwarding-options class-of-service ]
[ forwarding-options class-of-service queue-group queue-group-name
queue_group_residential ]
root@rtbrick:confd> set queue-numbers 8
```

3.9. Scheduler Configuration

Syntax

[forwarding-options class-of-service]
edit scheduler scheduler-name <scheduler_name>
[forwarding-options class-of-service scheduler scheduler-name sched_1]

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit scheduler scheduler-name sched_1
```

Scheduler Type

set type <type> [composite]

Example

```
[ forwarding-options class-of-service scheduler scheduler-name
rtbrick_sched_0 ]
root@rtbrick:confd> set type 3sp_wfq_discrete composite
```

Scheduler Shaper

set shaper-name <shaper_name>

Example

```
[ forwarding-options class-of-service scheduler scheduler-name sched_1 ]
root@rtbrick:confd> set shaper-name session_shaper
```

Command arguments

<scheduler-name>	User-defined Scheduler Name
<shaper_name>	(Optional) User-defined Shaper Name
<type>	Specifies the Scheduler Type 2sp_wfq_discrete 3sp_wfq_discrete strict_priority wfq_discrete_2sp 2sp_wfq_independent fair_queueing weighted_fair_queueing wfq_independent_2sp
composite	(Optional) keyword to specify the scheduler as composite type

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit scheduler scheduler-name sched_1
[ forwarding-options class-of-service scheduler scheduler-name sched_1 ]

[ forwarding-options class-of-service scheduler scheduler-name
rtbrick_sched_0 ]
root@rtbrick:confd> set type 3sp_wfq_discrete composite

[ forwarding-options class-of-service scheduler scheduler-name sched_1 ]
root@rtbrick:confd> set shaper-name shaper_1
```

3.10. Scheduler-Map Configuration

Syntax

[forwarding-options class-of-service]
root@rtbrick:confd> **edit scheduler-map scheduler-map-name** <scheduler_map_name> [**queue-group-name** <queue_group_name> **queue-name** <queue_name> | **scheduler-name** <scheduler_name>]

Command arguments

<scheduler_map_name>	User-defined Scheduler-Map Name
<scheduler_name>	User-defined Scheduler Name

<queue_group_name>	User-defined Queue-Group Name
<queue_name>	User-defined Queue-Name

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit scheduler-map scheduler-map-name
rtbrick_sched_map_residential queue-group-name queue_group_residential queue-
name IPTV
```

3.10.1. Scheduler to Port Configuration

Syntax

set scheduler-map-name <scheduler_map_name> **scheduler-name** <scheduler_name> **port-connection**

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential ]
root@rtbrick:confd> set scheduler-name rtbrick_sched_0 port-connection
```

3.10.2. Scheduler to Scheduler (same Scheduler-Map)

Syntax

set parent-scheduler <parent-scheduler> **connection-point** <connection_point> [weight <weight>]

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential ]
root@rtbrick:confd> set scheduler-name rtbrick_sched_0 parent-scheduler
rtbrick_sched_1 connection-point strict_priority_0 weight 1
```

3.10.3. Scheduler to Scheduler (different Scheduler-Map)

Syntax

set parent-scheduler <parent-scheduler> **connection-point** <connection_point> [weight <weight>] **scheduler-map-connection**

3.10.4. Queue to port

Syntax

set queue-group-name <queue_group_name> **queue-name** <queue_name> **port-connection**

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential ]
root@rtbrick:confd> set queue-group-name queue_group_residential queue-name
IPTV port-connection
```

3.10.5. Queue to scheduler

Syntax

set queue-group-name <queue_group_name> **queue-name** <queue_name> **parent-scheduler** <parent_scheduler_name> **parent-flow** <parent_flow> **connection-point** <connection_point> [weight <weight>]

The parent-flow configuration is optional.

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential ]
root@rtbrick:confd> set queue-group-name queue_group_residential queue-name
IPTV parent-scheduler rtbrick_sched_0 connection-point strict_priority_0
weight 1
```

3.11. Shaper Configuration

Syntax

[forwarding-options class-of-service]
edit shaper shaper-name <shaper_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit shaper shaper-name session_shaper
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
root@rtbrick:confd>
```

High Flow Shaping Rate

To configure only high-flow shaping rate, enter the following command:

rtb confd set forwarding-options class-of-service shaper shaper-name <shaper_name> **shaping-rate-high** <high_flow_rate>

Example

```
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
root@rtbrick:confd> set shaping-rate-high 9000
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
```

Low Flow Shaping Rate

To configure only low-flow shaping rate, enter the following command:

rtb confd set forwarding-options class-of-service shaper shaper-name <shaper_name> **shaping-rate-low** <low_flow_rate>

if priority propagation is not enabled, high-flow shaping value will be considered for shaper.

Example

```
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
root@rtbrick:confd> set shaping-rate-low 10000
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
root@rtbrick:confd>
```

Command Arguments

<shaper_name>	User-defined shaper name
<high_flow_rate>	High flow shaping rate in kilobits per second
<low_flow_rate>	Low flow shaping rate in kilobits per second

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit shaper shaper-name shaper_1
[ forwarding-options class-of-service shaper shaper-name shaper_1 ]
root@rtbrick:confd> set
 shaping-rate                            Set Rate in kbps


root@rtbrick:confd> set shaping-rate 10000
[ forwarding-options class-of-service shaper shaper-name shaper_1 ]
```

3.12. Priority Propagation

Syntax

rtb confd set forwarding-options class-of-service global priority-propagation <enable|disable>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global priority-propagation enable
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global priority-propagation dis
[<Enter>] Execute the command
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global priority-propagation disable
[ forwarding-options class-of-service ]
root@rtbrick:confd>
```

3.13. Profiles Configuration

Syntax

[forwarding-options class-of-service]
edit profiles profile-name <profile_name>

[forwarding-options class-of-service profiles profile-name <profile_name>]

Command arguments

<profile_name>	User-defined QoS Profile name

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit profiles profile-name rtbrick_residential_profile
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
```

BA Classifier

set classifier-name <classifier_name>

Command arguments

<classifier_name>	Classifier name

Example

```
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
root@rtbrick:confd> set classifier-name Classifier_IP
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
```

Multifield Classifier

set multifield-classifier-name <multifield_classifier_name>

Command arguments

<multifield_classifier_name>	MF Classifier name

Example

```
rtb confd set forwarding-options class-of-service global multifield-
classifier-name rtb_mf1
```

Class Policer-Map

set class-policer-map-name <class_policer_map_name>

Command arguments

<class_policer_map_name>	User-defined class to policer level map name map name

Example

```
[ forwarding-options class-of-service class-policer-map ]
root@rtbrick:confd> set class-policer-map-name class-policer1 class class-4
```

Policer

set policer-name <policer_name>

Command arguments

<policer_name>	User-defined Policer name

Example

```
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
root@rtbrick:confd> set policer-name Policer_Upstream
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
```

Class Queue-Map

set class-queue-map-name <class_queue_map_name>

Command arguments

<class_queue_map_name>	User-defined class to queue map name

Example

```
[ forwarding-options class-of-service class-queue-map ]
root@rtbrick:confd> set class-queue-map-name classifier_residential class
class-4
```

Scheduler-Map

set scheduler-map-name <scheduler_map_name>

Command arguments

<scheduler_map_name>	User-defined Scheduler map name

Example

```
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
root@rtbrick:confd> set scheduler-map-name rtbrick_sched_map_residential
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
root@rtbrick:confd>
```

Remark-Map

set remark-map-name <remark_map_name>

Command arguments

<remark_map_name>	Remarking map name

Example

```
[ forwarding-options class-of-service remark-map ]
root@rtbrick:confd> set remark-map-name remark-exp
```

3.14. Interface Configuration

3.14.1. Logical Interface QoS Profile

QOS Profile can be mapped to an L3 interface (that is, logical interface).

Syntax

```
root@rtbrick:confd> edit interface physical ifp-0/0/2 logical unit 1
[ interface physical ifp-0/0/2 logical unit 1 ]

root@rtbrick:confd> set class-of-service profile-name profile_up
[ interface physical ifp-0/0/2 logical unit 1 ]

root@rtbrick:confd>
```

Command arguments

<ifp-Name>	Physical Interface Name
<logical-unit-id>	Logical Interface Unit
<profile-Name>	Name of the QoS Profile

Example

```
root@rtbrick:confd> edit interface physical ifp-0/0/4 logical unit 0
[ interface physical ifp-0/0/4 logical unit 0 ]
root@rtbrick:confd> set class-of-service profile-name
rtbrick_residential_profile
[ interface physical ifp-0/0/4 logical unit 0 ]
root@rtbrick:confd>
```

3.14.2. Physical Interface Scheduler Map

Syntax

rtb confd set forwarding-options class-of-service interface physical <interface_name> **scheduler-map-name** <scheduler-map-name>

Command arguments

<interface_Name>	Physical Interface Name
<scheduler-map-name>	Scheduler map name

Example

```
TBD
```

3.14.3. Physical Interface Shaper

Syntax

rtb confd set forwarding-options class-of-service interface physical <interface_name> shaping-rate <rate>

Command arguments

<interface_Name>	Physical Interface Name
<rate>	Shaping Rate

Example

```
rtb confd set forwarding-options class-of-service interface physical ifp-
0/0/3 shaping-rate 10000
```

3.14.3.1. Physical Interface Header Compensation

3.14.3.1.1. Ingress Header Compensation

Syntax

rtb confd set forwarding-options class-of-service interface physical <interface_name> ingress-header-compensation bytes <header_compensation> <increment | decrement>

Command arguments

<interface_Name>	Physical Interface Name
<header_compensation>	Ingress header compensation bytes
<increment | decrement>	Header compensation to be decremented or incremented

Example

```
rtb confd set forwarding-options class-of-service interface physical ifp-
0/0/3 ingress-header-compensation bytes 2 increment
```

3.14.3.1.2. Egress Header Compensation

Syntax

rtb confd set forwarding-options class-of-service interface physical <interface_name> egress-header-compensation bytes <header_compensation> <increment | decrement>

Command arguments

<interface_Name>	Physical Interface Name
<header_compensation>	Ingress header compensation bytes
<increment | decrement>	Header compensation to be decremented or incremented

Example

```
rtb confd set forwarding-options class-of-service interface physical ifp-
0/0/3 egress-header-compensation bytes 2 increment
```

4. HQoS Show Running-Configuration

To display the running configuration, use the **show running-configuration** command.

Syntax

show running-configuration

```
"forwarding-options": [
      {
        "class-of-service": [
          {
            "classifier": [
              {
                "classifier-name:Classifier_EXP": {
                  "match-type:exp": {
                    "codepoint:0": {
                      "class": "class-0"
                    },
                    "codepoint:1": {
                      "class": "class-1"
                    },
                    "codepoint:2": {
                      "class": "class-2"
                    },
                    "codepoint:3": {
                      "class": "class-3"
                    },
                    "codepoint:4": {
                      "class": "class-4"
                    },
                    "codepoint:5": {
                      "class": "class-5"
                    },
                    "codepoint:6": {
                      "class": "class-6"
                    },
                    "codepoint:7": {
                      "class": "class-7"
                    }
                  }
                },
                "classifier-name:Classifier_IP": {
                  "match-type:ipv4-tos": {
                    "codepoint:8": {
                      "class": "class-0"
                    },
                    "codepoint:16": {
                      "class": "class-1"
                    },
                    "codepoint:32": {
```

```
                "class": "class-2"
              },
              "codepoint:64": {
                "class": "class-3"
              },
              "codepoint:96": {
                "class": "class-4"
              },
              "codepoint:164": {
                "class": "class-5"
              },
              "codepoint:196": {
                "class": "class-6"
              },
              "codepoint:224": {
                "class": "class-7"
              }
            }
          }
        }
      ],
      "class-policer-map": [
        {
          "class-policer-map-name:residential_policer_classifier": {
            "class:class-0": {
              "policer_level": "level-1"
            },
            "class:class-1": {
              "policer_level": "level-1"
            },
            "class:class-2": {
              "policer_level": "level-2"
            },
            "class:class-3": {
              "policer_level": "level-2"
            },
            "class:class-4": {
              "policer_level": "level-3"
            },
            "class:class-5": {
              "policer_level": "level-3"
            },
            "class:class-6": {
              "policer_level": "level-4"
            },
            "class:class-7": {
              "policer_level": "level-4"
            }
          }
        }
      ],
      "policer": [
        {
          "policer-name:Policer_Upstream": {
            "type": "two-rate-three-color",
            "flags": "color-blind",
            "levels": 4,
            "level1_cir": 1000,
```

```
                "level1_pir": 1100,
                "level1_cbs": 1000,
                "level1_pbs": 1000,
                "level2_cir": 2000,
                "level2_pir": 2100,
                "level2_cbs": 1000,
                "level2_pbs": 1000,
                "level3_cir": 2000,
                "level3_pir": 2100,
                "level3_cbs": 1000,
                "level3_pbs": 1000,
                "level4_cir": 1000,
                "level4_pir": 1100,
                "level4_cbs": 1000,
                "level4_pbs": 1000
              },
              "policer-name:_DEFAULT_POLICER_50_MB": {
                "type": "two-rate-three-color",
                "levels": 1,
                "level1_cir": 50000,
                "level1_pir": 50000,
                "level1_cbs": 33000,
                "level1_pbs": 33000
              }
            }
          ],
          "queue": [
            {
              "queue-name:BestEffort": {
                "queue_size": 25000
              },
              "queue-name:Control": {
                "queue_size": 25000
              },
              "queue-name:HotSpot": {
                "queue_size": 25000
              },
              "queue-name:IPTV": {
                "queue_size": 25000
              },
              "queue-name:LowDelay": {
                "queue_size": 25000
              },
              "queue-name:LowLoss": {
                "queue_size": 25000
              },
              "queue-name:TrafficControl": {
                "queue_size": 25000
              },
              "queue-name:Voice": {
                "queue_size": 25000
              }
            }
          ],
          "shaper": [
            {
              "shaper-name:control_shaper": {
                "shaper_rate_high": 3000,
```

```
                "shaper_rate_low": 6000
              },
              "shaper-name:session_shaper": {
                "shaper_rate_high": 2000,
                "shaper_rate_low": 4000
              },
              "shaper-name:voice_shaper": {
                "shaper_rate_high": 2500,
                "shaper_rate_low": 5000
              }
            }
          ],
          "class-queue-map": [
            {
              "class-queue-map-name:classifier_residential": {
                "class:class-0": {
                  "queue_name": "HotSpot"
                },
                "class:class-1": {
                  "queue_name": "BestEffort"
                },
                "class:class-2": {
                  "queue_name": "LowLoss"
                },
                "class:class-3": {
                  "queue_name": "TrafficControl"
                },
                "class:class-4": {
                  "queue_name": "LowDelay"
                },
                "class:class-5": {
                  "queue_name": "IPTV"
                },
                "class:class-6": {
                  "queue_name": "Voice"
                },
                "class:class-7": {
                  "queue_name": "Control"
                }
              }
            }
          ],
          "queue-group": [
            {
              "queue-group-name:queue_group_residential": {
                "queue_numbers": 8
              }
            }
          ],
          "scheduler": [
            {
              "scheduler-name:dpu_sched_1": {
                "type": "fair_queueing"
              },
              "scheduler-name:pon_sched_1": {
                "type": "fair_queueing"
              },
              "scheduler-name:rtbrick_sched_0": {
```

```
                "shaper_name": "session_shaper",
                "type": "3sp_wfq_discrete",
                "is_composite": true
              }
            }
          ],
          "profile": [
            {
              "profile-name:rtbrick_residential_profile": {
                "classifier_name": "Classifier_IP",
                "class_policer_map_name": "residential_policer_classifier",
                "policer_name": "Policer_Upstream",
                "scheduler_map_name": "rtbrick_sched_map_residential"
              }
            }
          ],
          "interface": [
            {
              "physical:ifp-0/0/3": {
                "scheduler_map_name": "dpu1",
                "shaper_rate": 5000
              }
            }
          ],
          "instance": [
            {
              "instance-name:default": {
                "classifier_name": "Classifier_EXP"
              },
              "instance-name:ip2vrf": {
              }
            }
          ],
          "scheduler-map": [
            {
              "scheduler-map-name:dpu1": {
                "scheduler-map-scheduler:dpu_sched_1": {
                  "parent_scheduler_name": "pon_sched_1",
                  "connection_point": "no_priority"
                },
                "scheduler-map-scheduler:pon_sched_1": {
                }
              },
              "scheduler-map-name:rtbrick_sched_map_residential": {
                "scheduler-map-queue:queue_group_residential IPTV": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "connection_point": "strict_priority_1"
                },
                "scheduler-map-queue:queue_group_residential Voice": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "connection_point": "strict_priority_0"
                },
                "scheduler-map-queue:queue_group_residential Control": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "connection_point": "strict_priority_0"
                },
                "scheduler-map-queue:queue_group_residential HotSpot": {
                  "parent_scheduler_name": "rtbrick_sched_0",
```

```
                    "parent_scheduler_flow": "low-flow",
                    "connection_point": "strict_priority_1"
                  },
                  "scheduler-map-queue:queue_group_residential LowLoss": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_2",
                    "weight": 2
                  },
                  "scheduler-map-queue:queue_group_residential LowDelay": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_1"
                  },
                  "scheduler-map-queue:queue_group_residential BestEffort": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "parent_scheduler_flow": "low-flow",
                    "connection_point": "strict_priority_0"
                  },
                  "scheduler-map-queue:queue_group_residential
TrafficControl": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_2",
                    "weight": 1
                  },
                  "scheduler-map-scheduler:rtbrick_sched_0": {
                  }
                }
              }
            ]
          }
        ]
      }
    ],
```

5. HQoS Show Commands

5.1. Display Policer Counters

This command allows you to display policer counters.

Syntax

rtb ifmd show qos policer counters

Example output

```
ubuntu@rtbrick:~$ rtb ifmd show qos policer counters
+-------------------------------+-------+---------------+---------------
+---------------+---------------+---------------+---------------+
              Name                Level     TotalPkts      TotalGrnPkts
AcceptedPkts     AccGrnPkts     DroppedPkts     DropGrnPkts
+-------------------------------+-------+---------------+---------------
+---------------+---------------+---------------+---------------+
         ip2vrf-pim-trap           1           0               0
0               0               0               0
        default-pim-trap           1           0               0
0               0               0               0
        ip2vrf-igmp-trap           1           0               0
0               0               0               0
        default-igmp-trap          1           0               0
0               0               0               0
     radius-srv1-v4-coa-trap       1           0               0
0               0               0               0
    lldp.ifp-0/0/1.trap.rule       1          116             116
116             116              0               0
    lldp.ifp-0/0/2.trap.rule       1          116             116
116             116              0               0
    radius-srv1-v4-acct-trap       1           0               0
0               0               0               0
    radius-srv1-v4-auth-trap       1           0               0
0               0               0               0
    lldp.ifp-0/0/49.trap.rule      1          116             116
116             116              0               0
    lldp.ifp-0/0/50.trap.rule      1          116             116
116             116              0               0
    lldp.ifp-0/0/51.trap.rule      1          116             116
116             116              0               0
    lldp.ifp-0/0/52.trap.rule      1          116             116
116             116              0               0
    lldp.ifp-0/0/53.trap.rule      1          116             116
116             116              0               0
   ppp-0/0/2/72339069014638594     1         116448          31362
39452           31362           76996             0
   ppp-0/0/2/72339069014638594     2           0               0
```

```
0                0                0                0
   ppp-0/0/2/72339069014638594     3            0                0
0                0                0                0
   ppp-0/0/2/72339069014638594     4            0                0
0                0                0                0
   ppp-0/0/2/72339069014638595     1          116418           31352
39440            31352            76978              0
   ppp-0/0/2/72339069014638595     2            0                0
0                0                0                0
   ppp-0/0/2/72339069014638595     3            0                0
0                0                0                0
   ppp-0/0/2/72339069014638595     4            0                0
0                0                0                0
 pppoed_ifp-0/0/2_13-14-13-1...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_13-14-13-1...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_15-16-15-1...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_15-16-15-1...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_17-18-17-1...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_17-18-17-1...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_19-20-19-2...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_19-20-19-2...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_11-4010-11...    1            0                0
0                0                0                0
 pppoed_ifp-0/0/2_11-4010-11...    1           696              696
696              696               0                0
 default_bgp_l4_trap_fe80::6...    1            0                0
0                0                0                0
 default_bgp_l4_trap_fe80::6...    1            0                0
0                0                0                0
ubuntu@rtbrick:~$
```

5.2. Display Queue Counters

This command allows you to display queue counters.

Syntax

```
rtb ifmd show qos queue counters
```

Example output

```
ubuntu@rtbrick:~$ rtb ifmd show qos queue counters
+-------------------------------+---------------+---------------+-------
+---------------+---------------+---------------+---------------+
             IF Name               Queue Grp         Queue       Class
Rx Bytes        Rx Pkts      Dropped Bytes    Dropped Pkts
+-------------------------------+---------------+---------------+-------
+---------------+---------------+---------------+---------------+
   ppp-0/0/2/72339069014638594  queue_group_...       IPTV         5
13307016         13254          12044988         11997
   ppp-0/0/2/72339069014638595  queue_group_...       IPTV         5
13276896         13224          12013864         11966
   ppp-0/0/2/72339069014638594  queue_group_...      Voice         6
0               0               0               0
   ppp-0/0/2/72339069014638595  queue_group_...      Voice         6
0               0               0               0
   ppp-0/0/2/72339069014638594  queue_group_...     Control        7
0               0               0               0
   ppp-0/0/2/72339069014638594  queue_group_...     HotSpot        1
0               0               0               0
   ppp-0/0/2/72339069014638594  queue_group_...     LowLoss        2
0               0               0               0
   ppp-0/0/2/72339069014638595  queue_group_...     Control        7
0               0               0               0
   ppp-0/0/2/72339069014638595  queue_group_...     HotSpot        1
0               0               0               0
   ppp-0/0/2/72339069014638595  queue_group_...     LowLoss        2
0               0               0               0
   ppp-0/0/2/72339069014638594  queue_group_...     LowDelay       4
0               0               0               0
   ppp-0/0/2/72339069014638595  queue_group_...     LowDelay       4
0               0               0               0
   ppp-0/0/2/72339069014638594  queue_group_...    BestEffort      0
0               0               0               0
   ppp-0/0/2/72339069014638595  queue_group_...    BestEffort      0
0               0               0               0
   ppp-0/0/2/72339069014638594  queue_group_...  TrafficControl    3
0               0               0               0
   ppp-0/0/2/72339069014638595  queue_group_...  TrafficControl    3
0               0               0               0
ubuntu@rtbrick:~$
```