

## Chapter 11

# Configuring Schedulers

You use *schedulers* to define the properties of output queues. These properties include the amount of interface bandwidth assigned to the queue, the size of the memory buffer allocated for storing packets, the priority of the queue, and the random early detection (RED) drop profiles associated with the queue.

You associate the schedulers with forwarding classes by means of *scheduler maps*. You can then associate each scheduler map with an interface, thereby configuring the hardware queues, packet schedulers, and RED processes that operate according to this mapping.

To configure class-of-service (CoS) schedulers, include the following statements at the [edit class-of-service] hierarchy level of the configuration:

```
class-of-service {
  interfaces {
    interface-name {
      scheduler-map map-name;
      scheduler-map-chassis map-name;
      schedulers number;
      shaping-rate rate;
      unit logical-unit-number {
        output-traffic-control-profile profile-name;
        scheduler-map map-name;
        shaping-rate rate;
      }
    }
  }
  fabric {
    scheduler-map {
      priority (high | low) scheduler scheduler-name;
    }
  }
  scheduler-maps {
    map-name {
      forwarding-class class-name scheduler scheduler-name;
    }
  }
}
```

```

schedulers {
  scheduler-name {
    buffer-size (percent percentage | remainder | temporal microseconds);
    drop-profile-map loss-priority (any | low | medium-low | medium-high | high)
      protocol (any | non-tcp | tcp) drop-profile profile-name;
    priority priority-level;
    transmit-rate (rate | percent percentage | remainder) <exact>;
  }
}
traffic-control-profiles profile-name {
  delay-buffer-rate (percent percentage | rate);
  guaranteed-rate (percent percentage | rate);
  scheduler-map map-name;
  shaping-rate (percent percentage | rate);
}
}

```

This chapter discusses the following topics:

- Default Schedulers on page 118
- Configuring a Scheduler on page 119
- Configuring the Scheduler Map on page 139
- Associating the Scheduler Map on page 140
- Configuring the Number of Schedulers for Ethernet IQ2 PICs on page 170

## Default Schedulers

---

Each forwarding class has an associated scheduler priority. Only two forwarding classes, best-effort and network-control (queue 0 and queue 3), are used in the JUNOS default scheduler configuration.

By default, the best-effort forwarding class (queue 0) receives 95 percent of the bandwidth and buffer space for the output link, and the network-control forwarding class (queue 3) receives 5 percent. The default drop profile causes the buffer to fill and then discard all packets until it has space.

The expedited-forwarding and assured-forwarding classes have no schedulers because, by default, no resources are assigned to queue 1 and queue 2. However, you can manually configure resources for the expedited-forwarding and assured-forwarding classes.

Also by default, each queue can exceed the assigned bandwidth if additional bandwidth is available from other queues. When a forwarding class does not fully use the allocated transmission bandwidth, the remaining bandwidth can be used by other forwarding classes if they receive a larger amount of offered load than the bandwidth allocated. For more information, see “Allocation of Leftover Bandwidth” on page 121.

The following default scheduler is provided when you install the JUNOS software. These settings are not visible in the output of the `show class-of-service` command; rather, they are implicit.

```
[edit class-of-service]
schedulers {
  network-control {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority low;
    drop-profile-map loss-priority any protocol any drop-profile terminal;
  }
  best-effort {
    transmit-rate percent 95;
    buffer-size percent 95;
    priority low;
    drop-profile-map loss-priority any protocol any drop-profile terminal;
  }
}
drop-profiles {
  terminal {
    fill-level 100 drop-probability 100;
  }
}
```

## Configuring a Scheduler

---

You configure a scheduler by assigning various properties to an output queue. For each queue, you can assign drop-profile maps, a transmission rate, a buffer size, and a queue priority. This section discusses the following topics:

- Configuring the Scheduler Drop Profile Map on page 119
- Configuring the Transmission Rate on page 120
- Configuring the Scheduler Buffer Size on page 122
- Configuring Priority Scheduling on page 132

### Configuring the Scheduler Drop Profile Map

Drop-profile maps associate drop profiles with a scheduler. The map examines the current loss priority setting of the packet (high, low, or any) and some Layer 4 information (TCP, non-TCP, or any) and assigns a drop profile according to these values. For example, you can specify that all TCP packets with low loss priority are assigned a drop profile that you name **low-drop**. You can associate multiple drop-profile maps with a single queue.

The scheduler drop profile defines the drop probabilities across the range of delay-buffer occupancy, thereby supporting the RED process. Depending on the drop probabilities, RED might drop packets aggressively long before the buffer becomes full, or it might drop only a few packets even if the buffer is almost full. For information on how to configure drop profiles, see “Configuring RED Drop Profiles” on page 111.

By default, the drop profile is mapped to packets with low PLP and any protocol type. To configure how packet types are mapped to a specified drop profile, include the `drop-profile-map` statement at the `[edit class-of-service schedulers scheduler-name]` hierarchy level:

```
[edit class-of-service schedulers scheduler-name]
drop-profile-map loss-priority (any | low | medium-low | medium-high | high)
protocol (any | non-tcp | tcp) drop-profile profile-name;
```

The map sets the drop profile for a specific PLP and protocol type. The inputs for the map are the PLP and the protocol type. The output is the drop profile. For more information about how CoS maps work, see Table 5 on page 9.

For each scheduler, you can configure four separate drop profile maps, one for each combination of loss priority (low or high) and IP transport protocol (TCP/IP or non-TCP/IP).

You can configure a maximum of 32 different drop profiles.

## Configuring the Transmission Rate

The transmission rate control determines the actual traffic bandwidth from each forwarding class you configure. The rate is specified in bits per second. Each queue is allocated some portion of the bandwidth of the outgoing interface.

This bandwidth amount can be a fixed value, such as 1 megabit per second (Mbps), a percentage of the total available bandwidth, or the rest of the available bandwidth. You can limit the transmission bandwidth to the exact value you configure, or allow it to exceed the configured rate if additional bandwidth is available from other queues. This property allows you to ensure that each queue receives the amount of bandwidth appropriate to its level of service.



**NOTE:** For 8-port, 12-port, and 48-port Fast Ethernet PICs, transmission scheduling is not supported.

On J-series Services Routers, you can use the `transmit-rate` statement to assign the WRR weights within a given priority level and not between priorities. For more information, see “Configuring Priority Scheduling” on page 132.

---

To configure transmission scheduling, include the `transmit-rate` statement at the `[edit class-of-service schedulers scheduler-name]` hierarchy level:

```
[edit class-of-service schedulers scheduler-name]
transmit-rate (rate | percent percentage | remainder) <exact>;
```

You can specify the transmit rate as follows:

- *rate*—Transmission rate, in bits per second. The rate can be from 3200 through 160,000,000,000 bits per second (bps).
- *percent percentage*—Percentage of transmission capacity.
- *remainder*—Use remaining rate available. In the configuration, you cannot combine the *remainder* and *exact* options.
- *exact*—Enforce the exact transmission rate or percentage you configure with the *transmit-rate rate* or *transmit-rate percent* statement. Under sustained congestion, a rate-controlled queue that goes into negative credit fills up and eventually drops packets. You specify the *exact* option as follows:

```
[edit class-of-service schedulers scheduler-name]
transmit-rate rate exact;
```

```
[edit class-of-service schedulers scheduler-name]
transmit-rate percent percentage exact;
```

In the configuration, you cannot combine the *remainder* and *exact* options.

### Example: Configuring the Transmission Rate

Configure the *best-effort* scheduler to use the remainder of the bandwidth on any interface to which it is assigned:

```
class-of-service {
  schedulers {
    best-effort {
      transmit-rate remainder;
    }
  }
}
```

### Allocation of Leftover Bandwidth

The allocation of leftover bandwidth is a complex topic. It is difficult to predict and to test, because the behavior of the software varies depending on the traffic mix.

If a queue receives offered loads in excess of the queue's bandwidth allocation, the queue has negative bandwidth credit, and receives a share of any available leftover bandwidth. Negative bandwidth credit means the queue has used up its allocated bandwidth. If a queue's bandwidth credit is positive, meaning it is not receiving offered loads in excess of its bandwidth configuration, then the queue does not receive a share of leftover bandwidth. If the credit is positive, then the queue does not need to use leftover bandwidth, because it can use its own allocation.

This use of leftover bandwidth is the default. If you do not want a queue to use any leftover bandwidth, you must configure it for strict allocation by including the *exact* option with the *transmit-rate* statement. With rate control in place, the specified bandwidth is strictly observed. (On J-series Services Routers, the *exact* option is useful within a given priority, but not between the priorities. For more information, see "Configuring Priority Scheduling" on page 132.)

On J-series Services Routers, leftover bandwidth is allocated to queues with negative credit in proportion to the configured transmit rate of the queues within a given priority level.

M-series and T-series platforms do not distribute leftover bandwidth in proportion to the configured transmit rate of the queues. Instead, the scheduler distributes the leftover bandwidth equally in round-robin fashion to queues that have negative bandwidth credit. All negative-credit queues can take the leftover bandwidth in equal share. This description suggests a simple round-robin distribution process among the queues with negative credits. In actual operation, a queue might change its bandwidth credit status from positive to negative and from negative to positive instantly while the leftover bandwidth is being distributed. Lower-rate queues tend to be allocated a larger share of leftover bandwidth, because their bandwidth credit is more likely to be negative at any given time, if they are overdriven persistently. Also, if there is a large packet size difference, (for example, queue 0 receives 64-byte packets, while queue 1 receives 1500-byte packets), then the actual leftover bandwidth distribution ratio can be skewed substantially, because each round-robin turn allows exactly one packet to be transmitted by a negative-credit queue, regardless of the packet size.

In summary, J-series Services Routers distribute leftover bandwidth in proportion to the configured rates of the negative-credit queues within a given priority level. M-series and T-series platforms distribute leftover bandwidth in equal share for the queues with the same priority and same negative-credit status.

### **Configuring the Scheduler Buffer Size**

To control congestion at the output stage, you can configure the delay-buffer bandwidth. The delay-buffer bandwidth provides packet buffer space to absorb burst traffic up to the specified duration of delay. Once the specified delay buffer becomes full, packets with 100 percent drop probability are dropped from the head of the buffer.

The default scheduler transmission rate for queues 0 through 7 are 95, 0, 0, 5, 0, 0, 0, and 0 percent of the total available bandwidth.

The default buffer size percentages for queues 0 through 7 are 95, 0, 0, 5, 0, 0, 0, and 0 percent of the total available buffer. The total available buffer per queue differs by PIC type, as shown in Table 18 on page 123.

To configure the buffer size, include the `buffer-size` statement at the `[edit class-of-service schedulers scheduler-name]` hierarchy level:

```
[edit class-of-service schedulers scheduler-name]
  buffer-size (percent percentage | remainder | temporal microseconds);
```

For each scheduler, you can configure the buffer size as one of the following:

- A percentage of the total buffer. The total buffer per queue is based on microseconds and differs by platform type, as shown in Table 18.
- The remaining buffer available. The remainder is the buffer percentage that is not assigned to other queues. For example, if you assign 40 percent of the delay buffer to queue 0, allow queue 3 to keep the default allotment of 5 percent, and assign the remainder to queue 7, then queue 7 uses approximately 55 percent of the delay buffer.
- A temporal value, in microseconds. For the temporal setting, the queuing algorithm starts dropping packets when it queues more than a computed number of bytes. This maximum is computed by multiplying the logical interface speed by the configured temporal value. The buffer size temporal value per queue differs by platform type, as shown in Table 18.

For information about configuring large buffer sizes on IQ PICs, see “Configuring Large Delay Buffers for Slower Interfaces” on page 124.

**Table 18: Buffer Size Temporal Value Ranges by Platform Type**

Platforms	Temporal Value Ranges
T-series and M320, Type 1 and Type 2 FPCs	1 through 80,000 microseconds
T-series and M320, Type 3 FPCs	1 through 50,000 microseconds
M7i, M10i, M5, and M10	1 through 100,000 microseconds
Other M-series	1 through 200,000 microseconds
IQ PICs on all platforms	1 through 100,000 microseconds
<b>With Large Buffer Sizes Enabled</b>	
IQ PICs on all platforms	1 through 500,000 microseconds
Gigabit Ethernet IQ VLANs	
With shaping rate up to 10 mbps	1 through 400,000 microseconds
With shaping rate up to 20 mbps	1 through 300,000 microseconds
With shaping rate up to 30 mbps	1 through 200,000 microseconds
With shaping rate up to 40 mbps	1 through 150,000 microseconds

For more information about configuring delay buffers, see the following subtopics:

- Configuring Large Delay Buffers for Slower Interfaces on page 124
- Enabling and Disabling the Memory Allocation Dynamic per Queue on page 130

### Configuring Large Delay Buffers for Slower Interfaces

By default, T1, E1, and NxDS0 interfaces and DLCIs configured on channelized IQ PICs are limited to 100,000 microseconds of delay buffer. (The default average packet size on the IQ PIC is 40 bytes.) For these interfaces, it might be necessary to configure a larger buffer size to prevent congestion and packet dropping. You can do so on the following PICs:

- Channelized IQ
- 4-port E3 IQ
- Gigabit Ethernet IQ and IQ2

Congestion and packet dropping occur when large bursts of traffic are received by slower interfaces. This happens when faster interfaces pass traffic to slower interfaces, which is often the case when edge devices receive traffic from the core of the network. For example, a 100,000-microsecond T1 delay buffer can absorb only 20 percent of a 5000-microsecond burst of traffic from an upstream OC3 interface. In this case, 80 percent of the burst traffic is dropped.

Table 19 shows some recommended buffer sizes needed to absorb typical burst sizes from various upstream interface types.

**Table 19: Recommended Delay Buffer Sizes**

Length of Burst	Upstream Interface	Downstream Interface	Recommended Buffer on Downstream Interface
5000 microseconds	OC3	E1 or T1	500,000 microseconds
5000 microseconds	E1, T1	E1 or T1	100,000 microseconds
1000 microseconds	T3	E1 or T1	100,000 microseconds

To ensure that traffic is queued and transmitted properly on E1, T1, and NxDS0 interfaces and DLCIs, you can configure a buffer size larger than the default maximum. To enable larger buffer sizes to be configured, include the `q-pic-large-buffer` statement at the `[edit chassis fpc slot-number pic pic-number]` hierarchy level:

```
[edit chassis fpc slot-number pic pic-number]
q-pic-large-buffer;
```

When you include the `q-pic-large-buffer` statement in the configuration, the larger buffer is transparently available for allocation to scheduler queues. The larger buffer maximum varies by interface type, as shown in Table 20.

**Table 20: Maximum Delay Buffer with ‘q-pic-large-buffer’ Enabled by Interface**

Platform, PIC, or Interface Type	Maximum Buffer Size
<b>With Large Buffer Sizes Not Enabled</b>	
T-series and M320, Type 1 and Type 2 FPCs	80,000 microseconds
T-series and M320, Type 3 FPCs	50,000 microseconds
Other M-series	200,000 microseconds
IQ PICs on all platforms	100,000 microseconds
<b>With Large Buffer Sizes Enabled</b>	
Channelized T3 and channelized OC3 DLCIs—Maximum sizes vary by shaping rate:	
With shaping rate from 64,000 through 255,999 bps	4,000,000 microseconds
With shaping rate from 256,000 through 511,999 bps	2,000,000 microseconds
With shaping rate from 512,000 through 1,023,999 bps	1,000,000 microseconds
With shaping rate from 1,024,000 through 2,048,000 bps	500,000 microseconds
With shaping rate from 2,048,001 through 10 mbps	400,000 microseconds
With shaping rate from 10,000,001 through 20 mbps	300,000 microseconds
With shaping rate from 20,000,001 through 30 mbps	200,000 microseconds
With shaping rate from 30,000,001 through 40 mbps	150,000 microseconds
With shaping rate up to 40,000,001 bps and above	100,000 microseconds
NxDSO IQ Interfaces—Maximum sizes vary by channel size:	
1xDSO through 3xDSO	4,000,000 microseconds
4xDSO through 7xDSO	2,000,000 microseconds
8xDSO through 15xDSO	1,000,000 microseconds
16xDSO through 32xDSO	500,000 microseconds
Other IQ interfaces	500,000 microseconds

If you configure a delay buffer larger than the new maximum, the candidate configuration can be committed successfully. However, the setting is rejected by the packet forwarding component, the default setting is used instead, and a system log warning message is generated.

For interfaces that support DLCI queuing, the large buffer is supported for DLCIs on which the configured shaping rate is less than or equal to the physical interface bandwidth. For instance, when you configure a Frame Relay DLCI on a Channelized T3 IQ PIC, and you configure the shaping rate to be 1.5 Mbps, the amount of delay buffer that can be allocated to the DLCI is 500,000 microseconds, which is equivalent to a T1 delay buffer. For more information about DLCI queuing, see “Associating the Scheduler Map and a Shaping Rate with a DLCI or VLAN” on page 148.

For *NxDS0* interfaces, the larger buffer sizes can be up to 4,000,000 microseconds, depending on the number of DS0 channels in the *NxDS0* interface. For slower *NxDS0* interfaces with fewer channels, the delay buffer can be relatively larger than for faster *NxDS0* interfaces with more channels. This is shown in Table 22 on page 127. To calculate specific buffer sizes for various *NxDS0* interfaces, see “Maximum Delay Buffer for *NxDS0* Interfaces” on page 126.

You can allocate the delay buffer as either a percentage or a temporal value. The resulting delay buffer is calculated differently depending how you configure the delay buffer, as shown in Table 21.

**Table 21: Delay-Buffer Calculations**

Delay Buffer Configuration	Formula	Example
Percentage	available interface bandwidth * configured percentage buffer-size * maximum buffer = queue buffer	<p>If you configure a queue on a T1 interface to use 30 percent of the available delay buffer, the queue receives 28,125 bytes of delay buffer:</p> <pre> sched-expedited {   transmit-rate percent 30;   buffer-size percent 30; }                     </pre> <p>1.5 Mbps * 0.3 * 500,000 microseconds = 225,000 bits = 28,125 bytes</p>
Temporal	available interface bandwidth * configured percentage transmit-rate * configured temporal buffer-size = queue buffer	<p>If you configure a queue on a T1 interface to use 500,000 microseconds of delay buffer, and you configure the transmission rate to be 20 percent, the queue receives 18,750 bytes of delay buffer:</p> <pre> sched-best {   transmit-rate percent 20;   buffer-size temporal 500000; }                     </pre> <p>1.5 Mbps * 0.2 * 500,000 microseconds = 150,000 bits = 18,750 bytes</p>
Percentage, with buffer size larger than transmit rate		<p>In this example, the delay buffer is allocated twice the transmit rate. Maximum delay buffer latency can be up to twice the 500,000-microsecond delay buffer if the queue’s transmit rate cannot exceed the allocated transmit rate.</p> <pre> sched-extra-buffer {   transmit-rate percent 10;   buffer-size percent 20; }                     </pre>
FRF.16 LSQ bundles	<p>For total bundle bandwidth &lt; T1 bandwidth, the delay-buffer rate is 1 second.</p> <p>For total bundle bandwidth &gt;= T1 bandwidth, the delay-buffer rate is 200 milliseconds (ms).</p>	

**Maximum Delay Buffer for *NxDS0* Interfaces**

Because *NxDS0* interfaces carry less bandwidth than a T1 or E1 interface, the buffer size on an *NxDS0* interface can be relatively larger, depending on the number of DS0 channels combined. The maximum delay buffer size is calculated with the following formula:

$$\text{Interface Speed} * \text{Maximum Delay Buffer Time} = \text{Delay Buffer Size}$$

For example, a 1xDS0 interface has a speed of 64 kilobits (Kb) per second. At this rate, the maximum delay buffer time is 4,000,000 microseconds. Therefore, the delay buffer size is 32 KB:

$$64 \text{ Kb per second} * 4,000,000 \text{ microseconds} = 32 \text{ kilobytes (KB)}$$

Table 22 shows the delay-buffer calculations for 1xDS0 through 32xDS0 interfaces.

**Table 22: NxDS0 Transmission Rates and Delay Buffers (1 of 2)**

Interface Speed	Delay Buffer Size
<b>1xDS0 Through 4xDS0: Maximum Delay Buffer Time Is 4,000,000 Microseconds</b>	
1xDS0: 64 Kb per second	32 KB
2xDS0: 128 Kb per second	64 KB
3xDS0: 192 Kb per second	96 KB
<b>4xDS0 Through 7xDS0: Maximum Delay Buffer Time Is 2,000,000 Microseconds</b>	
4xDS0: 256 Kb per second	64 KB
5xDS0: 320 Kb per second	80 KB
6xDS0: 384 Kb per second	96 KB
7xDS0: 448 Kb per second	112 KB
<b>8xDS0 Through 15xDS0: Maximum Delay Buffer Time Is 1,000,000 Microseconds</b>	
8xDS0: 512 Kb per second	64 KB
9xDS0: 576 Kb per second	72 KB
10xDS0: 640 Kb per second	80 KB
11xDS0: 704 Kb per second	88 KB
12xDS0: 768 Kb per second	96 KB
13xDS0: 832 Kb per second	104 KB
14xDS0: 896 Kb per second	112 KB
15xDS0: 960 Kb per second	120 KB
<b>16xDS0 Through 32xDS0: Maximum Delay Buffer Time Is 500,000 Microseconds</b>	
16xDS0: 1024 Kb per second	64 KB
17xDS0: 1088 Kb per second	68 KB
18xDS0: 1152 Kb per second	72 KB
19xDS0: 1216 Kb per second	76 KB
20xDS0: 1280 Kb per second	80 KB
21xDS0: 1344 Kb per second	84 KB
22xDS0: 1408 Kb per second	88 KB
23xDS0: 1472 Kb per second	92 KB
24xDS0: 1536 Kb per second	96 KB
25xDS0: 1600 Kb per second	100 KB
26xDS0: 1664 Kb per second	104 KB
27xDS0: 1728 Kb per second	108 KB
28xDS0: 1792 Kb per second	112 KB
29xDS0: 1856 Kb per second	116 KB

**Table 22: NxDSO Transmission Rates and Delay Buffers (2 of 2)**

Interface Speed	Delay Buffer Size
30xDSO: 1920 Kb per second	120 KB
31xDSO: 1984 Kb per second	124 KB
32xDSO: 2048 Kb per second	128 KB

**Example: Configuring Large Delay Buffers for Slower Interfaces**

Set large delay buffers on interfaces configured on a Channelized OC12 IQ PIC. The CoS configuration binds a scheduler map to the interface specified in the chassis configuration. For information about the delay-buffer calculations in this example, see Table 21 on page 126.

```

chassis {
  fpc 0 {
    pic 0 {
      q-pic-large-buffer; # Enabling large delay buffer
      max-queues-per-interface 8; # Enabling eight queues (M320 and T-series)
    }
  }
}

```

**Configuring the Delay Buffer Value for a Scheduler**

You can assign to a physical or logical interface a scheduler map that is composed of different schedulers (or queues). The physical interface’s large delay buffer can be distributed to the different schedulers (or queues) using the `transmit-rate` and `buffer-size` statements.

The example shows two schedulers, `sched-best` and `sched-exped`, with the delay buffer size configured as a percentage (20 percent) and temporal value (300,000 microseconds), respectively. The `sched-best` scheduler has a transmit rate of 10 percent. The `sched-exped` scheduler has a transmit rate of 20 percent.

The `sched-best` scheduler’s delay buffer is twice that of the specified transmit rate of 10 percent. Assuming that the `sched-best` scheduler is assigned to a T1 interface, this scheduler receives 20 percent of the total 500,000 microseconds of the T1 interface’s delay buffer. Therefore, the scheduler receives 18,750 bytes of delay buffer:

$$\text{available interface bandwidth} * \text{configured percentage buffer-size} * \text{maximum buffer} = \text{queue buffer}$$

$$1.5 \text{ Mbps} * 0.2 * 500,000 \text{ microseconds} = 150,000 \text{ bits} = 18,750 \text{ bytes}$$

Assuming that the `sched-best` scheduler is assigned to a T1 interface, this scheduler receives 300,000 microseconds of the T1 interface’s 500,000-microsecond delay buffer with the traffic rate at 20 percent. Therefore, the scheduler receives 11,250 bytes of delay buffer:

$$\text{available interface bandwidth} * \text{configured percentage transmit-rate} * \text{configured temporal buffer-size} = \text{queue buffer}$$

$$1.5 \text{ Mbps} * 0.2 * 300,000 \text{ microseconds} = 90,000 \text{ bits} = 11,250 \text{ bytes}$$

```

class-of-service {
  schedulers {
    sched-best {
      transmit-rate percent 10;
      buffer-size percent 20;
    }
    sched-exped {
      transmit-rate percent 20;
      buffer-size temporal 300000;
    }
  }
}

```

### Configuring the Physical Interface Shaping Rate

In general, the physical interface speed is the basis for calculating the delay buffer size. However, when you include the `shaping-rate` statement, the shaping rate becomes the basis for calculating the delay buffer size. This example configures the shaping rate on a T1 interface to 200 Kbps, which means that the T1 interface bandwidth is set to 200 Kbps instead of 1.5 Mbps. Because 200 Kbps is less than 4xDS0, this interface receives 4 seconds of delay buffer, or 800 Kbps. For more information, see Table 22 on page 127.

```

class-of-service {
  interfaces {
    t1-0/0/0:1:1 {
      shaping-rate 200k;
    }
  }
}

```

### Complete Configuration

This example shows a Channelized OC12 IQ PIC in FPC slot 0, PIC slot 0 and a channelized T1 interface with Frame Relay encapsulation. It also shows a scheduler map configuration on the physical interface.

```

chassis {
  fpc 0 {
    pic 0 {
      q-pic-large-buffer;
      max-queues-per-interface 8;
    }
  }
}
interfaces {
  coc12-0/0/0 {
    partition 1 oc-slice 1 interface-type coc1;
  }
  coc1-0/0/0:1 {
    partition 1 interface-type t1;
  }
}

```

```

t1-0/0/0:1:1 {
  encapsulation frame-relay;
  unit 0 {
    family inet {
      address 1.1.1.1/24;
    }
    dlcI 100;
  }
}
}
class-of-service {
  interfaces {
    t1-0/0/0:1:1 {
      scheduler-map smap-1;
    }
  }
  scheduler-maps {
    smap-1 {
      forwarding-class best-effort scheduler sched-best;
      forwarding-class expedited-forwarding scheduler sched-exped;
      forwarding-class assured-forwarding scheduler sched-assure;
      forwarding-class network-control scheduler sched-network;
    }
  }
  schedulers {
    sched-best {
      transmit-rate percent 40;
      buffer-size percent 40;
    }
    sched-exped {
      transmit-rate percent 30;
      buffer-size percent 30;
    }
    sched-assure {
      transmit-rate percent 20;
      buffer-size percent 20;
    }
    sched-network {
      transmit-rate percent 10;
      buffer-size percent 10;
    }
  }
}
}

```

### Enabling and Disabling the Memory Allocation Dynamic per Queue

In the JUNOS software, the memory allocation dynamic (MAD) is a mechanism that dynamically provisions extra delay buffer when a queue is using more bandwidth than it is allocated in the transmit rate setting. With this extra buffer, queues absorb traffic bursts more easily, thus avoiding packet drops. The MAD mechanism can provision extra delay buffer only when extra transmission bandwidth is being used by a queue. This means that the queue might have packet drops if there is no surplus transmission bandwidth available.

For T-series and M320 platforms only, the MAD mechanism is enabled unless the delay buffer is configured with a temporal setting for a given queue. The MAD mechanism is particularly useful for forwarding classes carrying latency-immune traffic for which the primary requirement is maximum bandwidth utilization. In contrast, for latency-sensitive traffic, you might wish to disable the MAD mechanism because large delay buffers are not optimum.

To enable the MAD mechanism, include the `buffer-size percent` statement at the `[edit class-of-service schedulers scheduler-name]` hierarchy level:

```
[edit class-of-service schedulers scheduler-name]
  buffer-size percent percentage;
```

If desired, you can configure a buffer size that is greater than the configured transmission rate. The buffer can accommodate packet bursts that exceed the configured transmission rate, if sufficient excess bandwidth is available:

```
class-of-service {
  schedulers {
    sched-best {
      transmit-rate percent 20;
      buffer-size percent 30;
    }
  }
}
```

As stated previously, you can use a temporal delay buffer configuration to disable the MAD mechanism on a queue, thus limiting the size of the delay buffer. However, the effective buffer latency for a temporal queue is bounded not only by the buffer size value but also by the associated drop profile. If a drop profile specifies a drop probability of 100 percent at a fill-level less than 100 percent, the effective maximum buffer latency is smaller than the buffer size setting. This is because the drop profile specifies that the queue drop packets before the queue's delay buffer is 100 percent full.

Such a configuration might look like the following example:

```
class-of-service {
  drop-profiles {
    plp-high {
      fill-level 70 drop-probability 100;
    }
    plp-low {
      fill-level 80 drop-probability 100;
    }
  }
  schedulers {
    sched {
      buffer-size temporal 500000;
      drop-profile-map loss-priority low protocol any drop-profile plp-low;
      drop-profile-map loss-priority high protocol any drop-profile plp-high;
      transmit-rate percent 20;
    }
  }
}
```

## Configuring Priority Scheduling

The JUNOS software supports multiple levels of transmission priority, which in order of increasing priority are **low**, **medium-low**, **medium-high**, and **high**, and **strict-high**. This allows the software to service higher-priority queues before lower-priority queues.

Priority scheduling determines the order in which an output interface transmits traffic from the queues, thus ensuring that queues containing important traffic are provided better access to the outgoing interface. This is accomplished through a procedure in which the software examines the priority of the queue. In addition, the software determines if the individual queue is within its defined bandwidth profile. The bandwidth profile is discussed in “Configuring the Transmission Rate” on page 120. This binary decision, which is reevaluated on a regular time cycle, compares the amount of data transmitted by the queue against the amount of bandwidth allocated to it by the scheduler. When the transmitted amount is less than the allocated amount, the queue is considered to be in profile. A queue is out of profile when its transmitted amount is larger than its allocated amount.

The queues for a given output physical interface (or output logical interface if per-unit scheduling is enabled on that interface) are divided into sets based on their priority. Any such set contains queues of the same priority.

The software traverses the sets in descending order of priority. If at least one of the queues in the set has a packet to transmit, the software selects that set. A queue from the set is selected based on the weighted round robin (WRR) algorithm, which operates within the set.

The JUNOS software performs priority queuing using the following steps:

1. The software locates all high-priority queues that are currently in profile. These queues are serviced first in a weighted round-robin fashion.
2. The software locates all medium-high priority queues that are currently in profile. These queues are serviced second in a weighted round-robin fashion.
3. The software locates all medium-low priority queues that are currently in profile. These queues are serviced third in a weighted round-robin fashion.
4. The software locates all low-priority queues that are currently in profile. These queues are serviced fourth in a weighted round-robin fashion.
5. The software locates all high-priority queues that are currently out of profile and are not rate limited. The weighted round-robin algorithm is applied to these queues for servicing.
6. The software locates all medium-high priority queues that are currently out of profile and are not rate limited. The weighted round-robin algorithm is applied to these queues for servicing.

7. The software locates all medium-low priority queues that are currently out of profile and are not rate limited. The weighted round-robin algorithm is applied to these queues for servicing.
8. The software locates all low-priority queues that are currently out of profile and are also not rate limited. These queues are serviced last in a weighted round-robin manner.

To configure priority scheduling, include the `priority` statement at the `[edit class-of-service schedulers scheduler-name]` hierarchy level:

```
[edit class-of-service schedulers scheduler-name]
priority priority-level;
```

The priority level can be `low`, `medium-low`, `medium-high`, `high`, or `strict-high`. The priorities map to numeric priorities in the underlying hardware. In some cases, different priorities behave similarly, because two software priorities behave differently only if they map to two distinct hardware priorities. For more information, see Table 23 on page 139.

Higher-priority queues transmit packets ahead of lower priority queues as long as the higher-priority forwarding classes retain enough bandwidth credit. When you configure a higher-priority queue with a significant fraction of the transmission bandwidth, the queue might lock out (or *starve*) lower priority traffic.

Strict priority queuing works differently on different platforms. For more information, see “Configuring Strict-High Priority on J-series Platforms” on page 134 and “Configuring Strict-High Priority on M-series and T-series Platforms” on page 137.

### Example: Configuring Priority Scheduling

Configure priority scheduling, as shown in the following example:

1. Configure a scheduler, `be-sched`, with medium-low priority.

```
[edit class-of-service]
schedulers {
  be-sched {
    priority medium-low;
  }
}
```

2. Configure a scheduler map, `be-map`, that associates `be-sched` with the best-effort forwarding class.

```
[edit class-of-service]
scheduler-maps {
  be-map {
    forwarding-class best-effort scheduler be-sched;
  }
}
```

3. Assign `be-map` to a Gigabit Ethernet interface, `ge-0/0/0`.

```
[edit class-of-service]
interfaces {
  ge-0/0/0 {
    scheduler-map be-map;
  }
}
```

### Configuring Strict-High Priority on J-series Platforms

On J-series Services Routers, you can configure one queue per interface to have **strict-high** priority, which causes delay-sensitive traffic, such as voice traffic, to be dequeued and forwarded with minimum delay. Packets that are queued in a strict priority queue are dequeued before packets in other queues, including high priority queues.

The JUNOS software implementation of strict priority queuing on J-series platforms allows you to configure traffic policing that prevents lower-priority queues from being starved. The strict priority queue does not cause starvation of other queues because the configured policer allows the queue to exceed the configured bandwidth only when other queues are not congested. If the interface is congested, the software polices strict priority queues to the configured bandwidth.

To prevent queue starvation of other queues, you must configure an egress policer that defines a limit for the amount of traffic that the queue can service. The software services all traffic in the strict priority queue that is under the defined limit. When strict priority traffic exceeds the limit, the policer marks the traffic in excess of the limit as out-of-profile. If the output port is congested, the software drops out-of-profile traffic.



**NOTE:** The out-of-profile limit does not work when you configure shaping on the interface.

---

If you wish, you can configure a second policer with an upper limit. When strict priority traffic exceeds the upper limit, the software drops the traffic in excess of the upper limit, regardless of whether the output port is congested. This upper-bound policer is not a requirement for preventing starvation of the lower priority queues. The policer for the lower bound, which marks the packets as out-of-profile, is sufficient to prevent starvation of other queues.

Following is a step-by-step description of how strict priority queuing and policing works on J-series Services Routers:

1. To identify delay-sensitive traffic, configure a behavior aggregate (BA) or multifield (MF) classifier.
2. To minimize delay, assign all delay-sensitive packets to the strict priority queue.

3. To prevent starvation on other queues, configure a policer that checks the data stream entering the strict priority queue. The policer defines a lower bound, marks the packets that exceed the lower bound as out-of-profile, and drops the out-of-profile packets if the physical interface is congested. If there is no congestion, the software forwards all packets, including the out-of-profile packets.
4. Optionally, you can configure another policer that defines an upper bound and drops the packets that exceed the upper bound, regardless of congestion on the physical interface.

To configure strict priority queuing and prevent starvation of other queues, include the `priority strict-high` statement at the `[edit class-of-service schedulers scheduler-name]` hierarchy level and the `if-exceeding` and `then out-of-profile` statements at the `[edit firewall policer policer-name]` hierarchy level:

```
[edit class-of-service schedulers scheduler-name]
priority strict-high;

[edit firewall policer policer-name]
if-exceeding {
    bandwidth-limit bps;
    bandwidth-percent number;
    burst-size-limit bytes;
}
then out-of-profile;
```

To verify your configuration, you can issue the following operational mode commands:

- `show class-of-service scheduler-map map-name`
- `show interfaces interface-name extensive`
- `show interfaces queue interface-name`

### **Example: Configuring Strict-High Priority on J-series Platforms**

Use a BA classifier to classify traffic based on the IP precedence of the packet. The classifier defines IP precedence value 101 as voice traffic and 000 as data traffic.

Configure two policers on the output interface that identify excess voice traffic belonging to the `voice-class` forwarding class. If the traffic exceeds 1 Mbps, a policer marks the traffic in excess of 1 Mbps as out-of-profile. If the traffic exceeds 2 Mbps, the second policer discards the traffic in excess of 2 Mbps.

**Configure a  
BA Classifier**

```
class-of-service {
  classifiers {
    inet-precedence corp-traffic {
      forwarding-class voice-class {
        loss-priority low code-points 101;
      }
      forwarding-class data-class {
        loss-priority high code-points 000;
      }
    }
  }
}
```

**Configure the  
Forwarding Classes**

```
forwarding-classes {
  queue 0 voice-class;
  queue 1 data-class;
}
}
```

**Configure the  
Scheduler Map**

```
scheduler-maps {
  corp-map {
    forwarding-class voice-class scheduler voice-sched;
    forwarding-class data-class scheduler data-sched;
  }
}
```

**Configure the  
Schedulers**

```
schedulers {
  voice-sched {
    priority strict-high;
  }
  data-sched {
    priority low;
  }
}
```

**Apply the BA Classifier  
to an Input Interface**

```
interfaces {
  fe-0/0/0 {
    unit 0 {
      classifiers {
        inet-precedence corp-traffic;
      }
    }
  }
}
```

**Apply the  
Scheduler Map to  
an Output Interface**

```
e1-1/0/1 {
  scheduler-map corp-map;
}
}
```

**Configure Two Policers**

```
firewall {
  policer voice-excess {
    if-exceeding {
      bandwidth-limit 1m;
      burst-size-limit 200k;
    }
    then out-of-profile;
  }
}
```

```

    policer voice-drop {
      if-exceeding {
        bandwidth-limit 2m;
        burst-size-limit 200k;
      }
      then discard;
    }
  filter voice-term {
    term 01 {
      from {
        forwarding-class voice-class;
      }
      then {
        policer voice-drop;
        next term;
      }
    }
    term 02 {
      from {
        forwarding-class voice-class;
      }
      then policer voice-excess;
    }
    term 03 {
      then accept;
    }
  }
}

```

#### Apply the Filter to the Output Interface

```

interfaces {
  e1-1/0/1 {
    unit 0 {
      family inet {
        filter {
          output voice-term;
        }
        address 11.1.1.1/24;
      }
    }
  }
}

```

### Configuring Strict-High Priority on M-series and T-series Platforms

On M-series and T-series platforms, you can configure one queue per interface to have **strict-high** priority, which works the same as **high** priority, but provides unlimited transmission bandwidth. As long as the queue with **strict-high** priority has traffic to send, it receives precedence over all other queues, except queues with **high** priority. Queues with **strict-high** and **high** priority take turns transmitting packets until the **strict-high** queue is empty, the **high** priority queues are empty, or the **high** priority queues run out of bandwidth credit. Only when these conditions are met can lower priority queues send traffic.

When you configure a queue to have **strict-high** priority, you do not need to include the **transmit-rate** statement in the queue configuration because the transmission rate of a **strict-high** priority queue is not limited by the WRR configuration. If you do configure a transmission rate on a **strict-high** priority queue, it does not affect the WRR operation. The transmission rate only serves as a placeholder in the output of commands such as the **show interface queue** command.

**Strict-high** priority queues might starve **low** priority queues. The **high** priority allows you to protect traffic classes from being starved by traffic in a **strict-high** queue. For example, a network-control queue might require a small bandwidth allocation (say, 5 percent). You can assign **high** priority to this queue to prevent it from being underserved.

A queue with **strict-high** priority supersedes bandwidth guarantees for queues with lower priority; therefore, we recommend that you use the **strict-high** priority to ensure proper ordering of special traffic, such as voice traffic. You can preserve bandwidth guarantees for queues with lower priority by allocating to the queue with **strict-high** priority only the amount of bandwidth that it generally requires. For example, consider the following allocation of transmission bandwidth:

- Q0 BE—20 percent, **low** priority
- Q1 EF—30 percent, **strict-high** priority
- Q2 AF—40 percent, **low** priority
- Q3 NC—10 percent, **low** priority

This bandwidth allocation assumes that, in general, the EF forwarding class requires only 30 percent of an interface's transmission bandwidth. However, if short bursts of traffic are received on the EF forwarding class, 100 percent of the bandwidth is given to the EF forwarding class because of the **strict-high** setting.

### Transmission Scheduling and Platform Differences

When you configure queue priorities, note the following hardware platform differences:

- On all platforms, you can configure one queue per interface to have **strict-high** priority. However, the **strict-high** priority works differently on J-series platforms than it does on M-series and T-series platforms. For more information, see “Configuring Strict-High Priority on J-series Platforms” on page 134 and “Configuring Strict-High Priority on M-series and T-series Platforms” on page 137.
- The **strict-high** priority works differently on AS PIC link services IQ (Isq) interfaces. For link services IQ interfaces, a **strict-high**-priority queue might starve all the other queues. For more information, see the *JUNOS Services Interfaces Configuration Guide*.

- On J-series Services Routers, high priority queues might starve low priority queues. For example:

Queue priority and transmission rate:

Queue 0: priority low, transmit-rate 50 percent  
Queue 2: priority high, transmit-rate 30 percent

Traffic profile:

Queue 0: 100 percent of the interface speed  
Queue 2: 100 percent of the interface speed

Results:

Queue 0: 0 percent of traffic is delivered.  
Queue 2: 100 percent of traffic is delivered.

- On J-series Services Routers, you can use the `transmit-rate` statement to assign the WRR weights within a given priority level and not between priorities.
- On J-series Services Routers, the `transmit-rate exact` option is useful within a given priority and not between the priorities.
- The priority levels you configure map to hardware priority levels. These priority mappings depend on the FPC type in which the PIC is mounted.

Table 23 shows the priority mappings by FPC type. Note, for example, that on M320 FPCs and T-series enhanced FPCs, the software priorities `medium-low` and `medium-high` behave similarly because they map to the same hardware priority level.

**Table 23: Scheduling Priority Mappings by FPC Type**

Priority Levels	Mappings for FPCs	Mappings for M320 FPCs and T-series Enhanced FPCs
low	0	0
medium-low	0	1
medium-high	1	1
high	1	2
strict-high (full interface bandwidth)	1	2

## Configuring the Scheduler Map

Once you define a scheduler, you can include it in a *scheduler map*, which maps a specified forwarding class to a scheduler configuration. To do this, include the `scheduler-maps` statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
scheduler-maps {
  map-name {
    forwarding-class class-name scheduler scheduler-name;
  }
}
```

## Associating the Scheduler Map

---

Physical interfaces (for example, `t3-0/0/0`, `t3-0/0/0:0`, and `ge-0/0/0`) support scheduling with any encapsulation type pertinent to that physical interface. For a single port, you cannot apply scheduling to the physical interface if you have applied scheduling to one or more of the associated logical interfaces.

Logical interfaces (for example, `t3-0/0/0 unit 0` and `ge-0/0/0 unit 0`) support scheduling on data link connection identifiers (DLCIs) or VLANs only.

In the JUNOS software implementation, the term *logical interfaces* generally refers to interfaces you configure by including the `unit` statement at the `[edit interfaces interface-name]` hierarchy level. Logical interfaces have the `.logical` descriptor at the end of the interface name, as in `ge-0/0/0.1` or `t1-0/0/0:0.1`, where the logical unit number is 1.

Although channelized interfaces are generally thought of as logical or virtual, the JUNOS software sees T3, T1, and NxDS0 interfaces within a channelized IQ PIC as physical interfaces. For example, both `t3-0/0/0` and `t3-0/0/0:1` are treated as physical interfaces by the JUNOS software. In contrast, `t3-0/0/0.2` and `t3-0/0/0:1.2` are considered logical interfaces because they have the `.2` at the end of the interface names.

Within the `[edit class-of-service]` hierarchy level, you cannot use the `.logical` descriptor when you assign properties to logical interfaces. Instead, you must include the `unit` statement in the configuration. For example:

```
[edit class-of-service]
user@host# set interfaces t3-0/0/0 unit 0 scheduler-map map1
```

This section discusses the following topics:

- Associating the Scheduler Map with a Physical Interface on page 141
- Associating the Scheduler Map and a Shaping Rate with a Physical Interface on page 141
- Associating the Scheduler Map and a Shaping Rate with a DLCI or VLAN on page 148
- Oversubscribing Interface Bandwidth on page 153
- Providing a Guaranteed Minimum Rate on page 160
- Associating the Scheduler Map with the Packet Forwarding Component Queues on page 164
- Associating a Scheduler with a Fabric Priority on page 169

### Associating the Scheduler Map with a Physical Interface

After you have defined the scheduler map, as described in “Configuring the Scheduler Map” on page 139, you can associate it with an output interface. To do this, include the `scheduler-map` statement at the [edit class-of-service interfaces *interface-name*] hierarchy level:

```
[edit class-of-service interfaces interface-name]  
scheduler-map map-name;
```

Interface wildcards are supported.

Generally, you can associate schedulers with physical interfaces only. For some IQ interfaces, you can also associate schedulers with the logical interface. For more information, see “Associating the Scheduler Map and a Shaping Rate with a DLCI or VLAN” on page 148.



**NOTE:** For original Channelized OC12 PICs, limited CoS functionality is supported. For more information, contact Juniper Networks customer support.

---

### Associating the Scheduler Map and a Shaping Rate with a Physical Interface

For IQ PICs, you can configure physical interfaces to shape traffic based on the rate-limited bandwidth of the total interface bandwidth. This allows you to shape the output of the physical interface, so that the interface transmits less traffic than it is physically capable of carrying.

If you do not configure a shaping rate on the physical interface, the default physical interface bandwidth is based on the channel bandwidth and the time slot allocation.



**NOTE:** The `shaping-rate` statement cannot be applied to a physical interface on J-series routing platforms.

---

To configure shaping on the interface, include the `shaping-rate` statement at the [edit class-of-service interfaces *interface-name*] hierarchy level:

```
[edit class-of-service interfaces interface-name]  
shaping-rate rate;
```

You can specify a peak bandwidth rate in bps, either as a complete decimal number or as a decimal number followed by the abbreviation k (1000), m (1,000,000), or g (1,000,000,000). For physical interfaces, the range is from 1000 through 160,000,000,000 bps. (For logical interfaces, the range is 1000 through 32,000,000,000 bps.) The sum of the bandwidths you allocate to all physical interfaces on a PIC must not exceed the bandwidth of the PIC.

If you configure a shaping rate that exceeds the physical interface bandwidth, the new configuration is ignored, and the previous configuration remains in effect. For example, if you configure a shaping rate that is 80 percent of the physical interface bandwidth, then change the configuration to 120 percent of the physical interface bandwidth, the 80 percent setting remains in effect. This holds true unless the PIC is restarted, in which case the default bandwidth goes into effect. As stated previously, the default bandwidth is based on the channel bandwidth and the time slot allocation.

Optionally, you can instead configure scheduling and rate shaping on logical interfaces, as described in “Associating the Scheduler Map and a Shaping Rate with a DLCI or VLAN” on page 148. In general, logical and physical interface traffic shaping is mutually exclusive. You can include the `shaping-rate` statement at the `[edit class-of-service interfaces interface-name]` hierarchy level or the `[edit class-of-service interfaces interface-name unit logical-unit-number]` hierarchy level, but not both. For Gigabit Ethernet IQ PICs only, you can configure hierarchical traffic shaping, meaning the shaping is performed on both the physical interface and the logical interface. For more information, see “Configuring Hierarchical Input Shapers” on page 206.

To view the results of your configuration, issue the following show commands:

- `show class-of-service interface interface-name`
- `show interface interface-name extensive`
- `show interfaces queue`

### Shaping Rate Calculations

For shaping rate and WRR, the information included in the calculations varies by PIC type, as shown in Table 24.



**NOTE:** Gigabit Ethernet IQ2 PICs are unique in that they support ingress scheduling and shaping. The calculations shown for Gigabit Ethernet IQ2 PICs apply to both ingress and egress scheduling and shaping. For other PICs, the calculations apply to egress scheduling and shaping only.

For more information, see “Shaping Input and Output Traffic on Ethernet IQ2 Interfaces” on page 199.

---

**Table 24: Shaping Rate and WRR Calculations by PIC Type**

PIC Type	Platform	Shaping Rate and WRR Calculations Include
Gigabit Ethernet IQ2 PIC	All	For ingress and egress: L3 header + L2 header + frame check sequence (FCS)
Gigabit Ethernet IQ PIC	All	L3 header + L2 header + FCS
Non-IQ PIC	T-series and M320 with Enhanced FPC	L3 header + L2 header + 4-byte FCS + interpacket gap (IPG) + start-of-frame delimiter (SFD) + preamble
	T-series with Nonenhanced FPC	L3 header
	Other M-series	L3 header + L2 header
IQ PIC with a SONET/SDH interface	All	L3 header + L2 header + FCS
Non-IQ PIC with a SONET/SDH interface	T-series and M320 with Enhanced FPC	L3 header + L2 header + 4-byte FCS + IPG + SFD + Preamble
	T-series with Nonenhanced FPC	L3 header
	Other M-series	L3 header + L2 header

### Examples: Associating a Scheduler Map and a Shaping Rate with a Physical Interface

The section includes the following examples:

- Channelized T1 IQ PIC Clear-Channel T1 on page 143
- Channelized T1 IQ PIC CT1 > DS0 on page 144
- Channelized E1 IQ PIC Clear-Channel E1 on page 144
- Channelized E1 IQ PIC CE1 > DS0 on page 145
- Channelized DS3 IQ PIC Clear-Channel T3 on page 146
- Channelized DS3 IQ PIC Fractional T1 on page 146
- Channelized DS3 IQ PIC CT3 > T1 > DS0 on page 147

#### Channelized T1 IQ PIC Clear-Channel T1

```

interfaces {
  ct1-2/1/0 {
    no-partition interface-type t1;
  }
  t1-2/1/0 {
    unit 0 {
      family inet {
        address 10.40.1.1/30;
      }
    }
  }
}

```

```

class-of-service {
  interfaces {
    t1-2/1/0 {
      shaping-rate 3000;
    }
  }
}

Channelized T1 IQ PIC
CT1 > DS0

interfaces {
  ct1-0/0/9 {
    partition 1 timeslots 1-2 interface-type ds;
  }
  ds-0/0/9:1 {
    no-keepalives;
    unit 0 {
      family inet {
        address 10.10.1.1/30;
      }
    }
  }
}

class-of-service {
  interfaces {
    ds-0/0/9:1 {
      scheduler-map sched_port_1;
      shaping-rate 2000;
    }
  }
}

Channelized E1 IQ PIC
Clear-Channel E1

interfaces {
  ce1-2/1/0 {
    no-partition interface-type e1;
  }
  e1-2/1/0 {
    unit 0 {
      family inet {
        address 10.40.1.1/30;
      }
    }
  }
}

class-of-service {
  interfaces {
    e1-2/1/0 {
      shaping-rate 4000;
    }
  }
}

```

**Channelized E1 IQ PIC**  
**CE1 > DS0**

```

interfaces {
  ce1-1/3/1 {
    partition 1 timeslots 1-4 interface-type ds;
    partition 2 timeslots 5-6 interface-type ds;
  }
  ds-1/3/1:1 {
    no-keepalives;
    unit 0 {
      family inet {
        address 10.10.1.1/30;
      }
    }
  }
  ds-1/3/1:2 {
    no-keepalives;
    unit 0 {
      family inet {
        address 10.10.1.5/30;
      }
    }
  }
}

class-of-service {
  interfaces {
    ds-1/3/1:1 {
      scheduler-map sched_port_1;
      shaping-rate 1000;
    }
    ds-1/3/1:2 {
      scheduler-map sched_port_1;
      shaping-rate 1500;
    }
  }
}

```

**Channelized DS3 IQ PIC  
Clear-Channel T3**

```

interfaces {
  ct3-2/1/0 {
    no-partition;
  }
  t3-2/1/0 {
    unit 0 {
      family inet {
        address 10.40.1.1/30;
      }
    }
  }
}
class-of-service {
  interfaces {
    t3-2/1/0 {
      shaping-rate 2500;
      unit 0 {
        scheduler-map sched_port_1;
      }
    }
  }
}

```

**Channelized DS3 IQ PIC  
Fractional T1**

```

interfaces {
  ct3-1/1/3 {
    partition 1-3 interface-type t1;
  }
  t1-1/1/3:1 {
    t1-options {
      timeslots 1-2;
    }
    unit 0 {
      family inet {
        address 10.10.1.1/30;
      }
    }
  }
  t1-1/1/3:2 {
    t1-options {
      timeslots 3-6;
    }
    unit 0 {
      family inet {
        address 10.10.1.5/30;
      }
    }
  }
}

```

```

t1-1/1/3:3 {
  t1-options {
    timeslots 7-12;
  }
  unit 0 {
    family inet {
      address 10.10.1.9/30;
    }
  }
}
}
class-of-service {
  interfaces {
    t1-1/1/3:1 {
      scheduler-map sched_port_1;
      shaping-rate 1200;
    }
    t1-1/1/3:2 {
      scheduler-map sched_port_1;
      shaping-rate 1300;
    }
    t1-1/1/3:3 {
      scheduler-map sched_port_1;
      shaping-rate 1400;
    }
  }
}
}

```

**Channelized DS3 IQ PIC**  
**CT3 > T1 > DS0**

```

interfaces {
  ct3-2/1/3 {
    partition 1 interface-type ct1;
  }
  ct1-2/1/3:1 {
    partition 1 timeslots 1-4 interface-type ds;
  }
  ds-2/1/3:1:1 {
    unit 0 {
      family inet {
        address 10.20.144.1/30;
      }
    }
  }
}
class-of-service {
  interfaces {
    ds-2/1/3:1:1 {
      scheduler-map sched_port_1;
      shaping-rate 1100;
    }
  }
}
}
}

```

## Associating the Scheduler Map and a Shaping Rate with a DLCI or VLAN

By default, output scheduling is not enabled on logical interfaces. Logical interfaces without shaping configured share a default scheduler. This scheduler has a committed information rate (CIR) that equals 0. (The CIR is the guaranteed rate.) The default scheduler has a peak information rate (PIR) that equals the physical interface shaping rate.

*Logical interface scheduling* (also called *per-unit scheduling*) allows you to enable multiple output queues on a logical interface and associate an output scheduler and shaping rate with the queues. You can configure logical interface scheduling on the following PICs:

- Adaptive Services PIC, on link services IQ interfaces (lsq)
- Channelized E1 IQ PIC
- Channelized OC3 IQ PIC
- Channelized OC12 IQ PIC (Per-unit scheduling is not supported on T1 interfaces configured on this PIC.)
- Channelized STM1 IQ PIC
- Channelized T3 IQ PIC
- E3 IQ PIC
- Gigabit Ethernet IQ PIC
- Gigabit Ethernet IQ2 PIC
- Link services PIM (ls-) on J-series platforms

For J-series Services Routers only, you can configure per-unit scheduling for virtual channels. For more information, see “Configuring Virtual Channels” on page 215.

For Channelized and Gigabit Ethernet IQ PICs only, you can configure a shaping rate for a VLAN or DLCI and oversubscribe the physical interface by including the **shaping-rate** statement at the [edit class-of-service traffic-control-profiles] hierarchy level. With this configuration approach, you can independently control the delay-buffer rate, as described in “Oversubscribing Interface Bandwidth” on page 153.

Physical interfaces (for example, **t3-0/0/0**, **t3-0/0/0:0**, and **ge-0/0/0**) support scheduling with any encapsulation type pertinent to that physical interface. For a single port, you cannot apply scheduling to the physical interface if you apply scheduling to one or more of the associated logical interfaces.

For Gigabit Ethernet IQ2 PIC PICs only, you can configure hierarchical traffic shaping, meaning the shaping is performed on both the physical interface and the logical interface. You can also configure input traffic scheduling and shared scheduling. For more information, see “Shaping Input and Output Traffic on Ethernet IQ2 Interfaces” on page 199.

Logical interfaces (for example, `t3-0/0/0.0`, `ge-0/0/0.0`, and `t1-0/0/0:0.1`) support scheduling on DLCIs or VLANs only. Furthermore, logical interface scheduling is not supported on PICs that do not have IQ.



**NOTE:** In the JUNOS software implementation, the term *logical interfaces* generally refers to interfaces you configure by including the `unit` statement at the `[edit interfaces interface-name]` hierarchy level. As such, logical interfaces have the *.logical* descriptor at the end of the interface name, as in `ge-0/0/0.1` or `t1-0/0/0:0.1`, where the logical unit number is 1.

Although channelized interfaces are generally thought of as logical or virtual, the JUNOS software sees T3, T1, and NxDS0 interfaces within a channelized IQ PIC as physical interfaces. For example, both `t3-0/0/0` and `t3-0/0/0:1` are treated as physical interfaces by the JUNOS software. In contrast, `t3-0/0/0.2` and `t3-0/0/0:1.2` are considered logical interfaces because they have the `.2` at the end of the interface names.

Within the `[edit class-of-service]` hierarchy level, you cannot use the *.logical* descriptor when you assign properties to logical interfaces. Instead, you must include the `unit` statement in the configuration. For example:

```
[edit class-of-service]
user@host# set interfaces t3-0/0/0 unit 0 scheduler-map map1
```

Table 25 shows the interfaces that support transmission scheduling.

**Table 25: Transmission Scheduling Support by Interfaces Type (1 of 2)**

Interface Type	PIC Type	Supported	Examples
<b>IQ PICs</b>			
Physical interfaces	ATM2 IQ	Yes	Example of supported configuration: [edit class-of-service interfaces at-0/0/0] scheduler-map map-1;
Channelized interfaces configured on IQ PICs	Channelized DS3 IQ	Yes	Example of supported configuration: [edit class-of-service interfaces t1-0/0/0:1] scheduler-map map-1;
Logical interfaces (DLCIs and VLANs only) configured on IQ PICs	Gigabit Ethernet IQ with VLAN tagging enabled	Yes	Example of supported configuration: [edit class-of-service interfaces ge-0/0/0 unit 1] scheduler-map map-1;
	E3 IQ with Frame Relay encapsulation	Yes	Example of supported configuration: [edit class-of-service interfaces e3-0/0/0 unit 1] scheduler-map map-1;
	Channelized OC3 IQ with Frame Relay encapsulation	Yes	Example of supported configuration: [edit class-of-service interfaces t1-1/0/0:1:1 unit 0] scheduler-map map-1;
	Channelized STM1 IQ with Frame Relay encapsulation	Yes	Example of supported configuration: [edit class-of-service interfaces e1-0/0/0:1 unit 1] scheduler-map map-1;
	Channelized T3 IQ with Frame Relay encapsulation	Yes	Example of supported configuration: [edit class-of-service interfaces t1-0/0/0 unit 1] scheduler-map map-1;

**Table 25: Transmission Scheduling Support by Interfaces Type (2 of 2)**

Interface Type	PIC Type	Supported	Examples
Logical interfaces configured on IQ PICs (interfaces that are not DLCIs or VLANs)	E3 IQ PIC with Cisco HDLC encapsulation	No	Example of unsupported configuration: [edit class-of-service interfaces e3-0/0/0 unit 1] scheduler-map map-1;
	ATM2 IQ PIC with LLC/SNAP encapsulation	No	Example of unsupported configuration: [edit class-of-service interfaces at-0/0/0 unit 1] scheduler-map map-1;
	Channelized OC12 IQ PIC with PPP encapsulation	No	Example of unsupported configuration: [edit class-of-service interfaces t1-0/0/0:1 unit 1] scheduler-map map-1;
<b>Non-IQ PICs</b>			
Physical interfaces	T3	Yes	Example of supported configuration: [edit class-of-service interfaces t3-0/0/0] scheduler-map map-1;
Channelized OC12 PIC	Channelized OC12	Yes	Example of supported configuration: [edit class-of-service interfaces t3-0/0/0:1] scheduler-map map-1;
Channelized interfaces (except the Channelized OC12 PIC)	Channelized STM1	No	Example of unsupported configuration: [edit class-of-service interfaces e1-0/0/0:1] scheduler-map map-1;
Logical interfaces	Fast Ethernet	No	Example of unsupported configuration: [edit class-of-service interfaces fe-0/0/0 unit 1] scheduler-map map-1;
	Gigabit Ethernet	No	Example of unsupported configuration: [edit class-of-service interfaces ge-0/0/0 unit 0] scheduler-map map-1;
	ATM1	No	Example of unsupported configuration: [edit class-of-service interfaces at-0/0/0 unit 2] scheduler-map map-1;
	Channelized OC12	No	Example of unsupported configuration: [edit class-of-service interfaces t3-0/0/0:0 unit 2] scheduler-map map-1;

To configure transmission scheduling on logical interfaces, perform the following steps:

1. Enable scheduling on the interface by including the `per-unit-scheduler` statement at the `[edit interfaces interface-name]` hierarchy level:

```
[edit interfaces interface-name]  
per-unit-scheduler;
```

When you include this statement, the maximum number of VLANs supported is 767 on a single-port Gigabit Ethernet IQ PIC. On a dual-port Gigabit Ethernet IQ PIC, the maximum number is 383.

2. Associate a scheduler with the interface by including the `scheduler-map` statement at the `[edit class-of-service interfaces interface-name unit logical-unit-number]` hierarchy level:

```
[edit class-of-service interfaces interface-name unit logical-unit-number]  
scheduler-map map-name;
```

3. Configure shaping on the interface by including the `shaping-rate` statement at the `[edit class-of-service interfaces interface-name unit logical-unit-number]` hierarchy level:

```
[edit class-of-service interfaces interface-name unit logical-unit-number]  
shaping-rate rate;
```

By default, the logical interface bandwidth is the average of unused bandwidth for the number of logical interfaces that require default bandwidth treatment. You can specify a peak bandwidth rate in bps, either as a complete decimal number or as a decimal number followed by the abbreviation `k` (1000), `m` (1,000,000), or `g` (1,000,000,000). The range is from 1000 through 32,000,000,000 bps.

For FRF.16 bundles on link services interfaces, only shaping rates based on percentage are supported.

**Example: Associating the Scheduler Map Name with a DLCI or VLAN**

Associate the scheduler `sched-map-logical-0` with logical interface `unit 0` on physical interface `t3-1/0/0`, and allocate 10 megabits per second (Mbps) of transmission bandwidth to the logical interface.

Associate the scheduler `sched-map-logical-1` with logical interface `unit 1` on physical interface `t3-1/0/0`, and allocate 20 Mbps of transmission bandwidth to the logical interface.

The allocated bandwidth is shared among the individual forwarding classes in the scheduler map. Although these schedulers are configured on a single physical interface, they are independent from each other. Traffic on one logical interface unit does not affect the transmission priority, bandwidth allocation, or drop behavior on the other logical interface unit.

For another example, see the *JUNOS Feature Guide*.

```
[edit interfaces]
t3-1/0/0:1 {
  encapsulation frame-relay;
  per-unit-scheduler;
}

[edit class-of-service]
interfaces {
  t3-1/0/0:1 {
    unit 0 {
      scheduler-map sched-map-logical-0;
      shaping-rate 10m;
    }
    unit 1 {
      scheduler-map sched-map-logical-1;
      shaping-rate 20m;
    }
  }
}

scheduler-maps {
  sched-map-logical-0 {
    forwarding-class best-effort scheduler sched-best-effort-0;
    forwarding-class assured-forwarding scheduler sched-bronze-0;
    forwarding-class expedited-forwarding scheduler sched-silver-0;
    forwarding-class network-control scheduler sched-gold-0;
  }
  sched-map-logical-1 {
    forwarding-class best-effort scheduler sched-best-effort-1;
    forwarding-class assured-forwarding scheduler sched-bronze-1;
    forwarding-class expedited-forwarding scheduler sched-silver-1;
    forwarding-class network-control scheduler sched-gold-1;
  }
}
```

```

schedulers {
  sched-best-effort-0 {
    transmit-rate 4m;
  }
  sched-bronze-0 {
    transmit-rate 3m;
  }
  sched-silver-0 {
    transmit-rate 2m;
  }
  sched-gold-0 {
    transmit-rate 1m;
  }
  sched-best-effort-1 {
    transmit-rate 8m;
  }
  sched-bronze-1 {
    transmit-rate 6m;
  }
  sched-silver-1 {
    transmit-rate 4m;
  }
  sched-gold-1 {
    transmit-rate 2m;
  }
}

```

### **Oversubscribing Interface Bandwidth**

The term *oversubscribing interface bandwidth* means configuring shaping rates (peak information rates [PIRs]) so that their sum exceeds the interface bandwidth.

On Channelized IQ PICs, Gigabit Ethernet IQ, and FRF.16 link services IQ (LSQ) interfaces on AS PICs, you can oversubscribe interface bandwidth. This means that the logical interfaces (and DLCIs within an FRF.16 bundle) can be oversubscribed when there is leftover bandwidth. The oversubscription is capped to the configured PIR. Any unused bandwidth is distributed equally among oversubscribed logical interfaces or DLCIs.

For networks that are not likely to experience congestion, oversubscribing interface bandwidth improves network utilization, thereby allowing more customers to be provisioned on a single interface. If the actual data traffic does not exceed the interface bandwidth, oversubscription allows you to sell more bandwidth than the interface can support.

We recommend avoiding oversubscription in networks that are likely to experience congestion. Be cautious not to oversubscribe a service by too much, because this can cause degradation in the performance of the routing platform during congestion. When you configure oversubscription, starvation of some output queues can occur if the actual data traffic exceeds the physical interface bandwidth. You can prevent degradation by using statistical multiplexing to ensure that the actual data traffic does not exceed the interface bandwidth.



**NOTE:** You cannot oversubscribe interface bandwidth when you configure traffic shaping using the method described in “Associating the Scheduler Map and a Shaping Rate with a DLCI or VLAN” on page 148.

To configure oversubscription of the interface, perform the following steps:

1. Include the `shaping-rate` statement at the `[edit class-of-service traffic-control-profiles profile-name]` hierarchy level:

```
[edit class-of-service traffic-control-profiles profile-name]
shaping-rate (percent percentage | rate);
```

On LSQ interfaces, you can configure the shaping rate as a percentage from 1 through 100.

On IQ and IQ2 interfaces, you can configure the shaping rate as an absolute rate from 1000 through 160,000,000,000 bits per second.

Alternatively, you can configure a shaping rate for a logical interface and oversubscribe the physical interface by including the `shaping-rate` statement at the `[edit class-of-service interfaces interface-name unit logical-unit-number]` hierarchy level. However, with this configuration approach, you cannot independently control the delay-buffer rate, as described in Step 2.



**NOTE:** For channelized and Gigabit Ethernet IQ interfaces, the `shaping-rate` and `guaranteed-rate` statements are mutually exclusive. You cannot configure some logical interfaces to use a shaping rate and others to use a guaranteed rate. This means there are no service guarantees when you configure a PIR. For these interfaces, you can configure either a PIR or a committed information rate (CIR), but not both.

This restriction does not apply to Gigabit Ethernet IQ2 PICs or LSQ interfaces on AS PICs. For LSQ and Gigabit Ethernet IQ2 interfaces, you can configure both a PIR and a CIR on an interface. For more information about CIRs, see “Providing a Guaranteed Minimum Rate” on page 160.

For more information about Gigabit Ethernet IQ2 PICs, see “Shaping Input and Output Traffic on Ethernet IQ2 Interfaces” on page 199.

2. Optionally, you can base the delay-buffer calculation on a delay-buffer rate. To do this, include the `delay-buffer-rate` statement at the `[edit class-of-service traffic-control-profiles profile-name]` hierarchy level:

```
[edit class-of-service traffic-control-profiles profile-name]
delay-buffer-rate (percent percentage | rate);
```

The delay-buffer rate overrides the shaping rate as the basis for the delay-buffer calculation. In other words, the shaping rate or scaled shaping rate is used for delay-buffer calculations only when the delay-buffer rate is not configured.

For LSQ interfaces, if you do not configure a delay-buffer rate, the guaranteed rate (CIR) is used to assign buffers. If you do not configure a guaranteed rate, the shaping rate (PIR) is used in the undersubscribed case, and the scaled shaping rate is used in the oversubscribed case.

On LSQ interfaces, you can configure the delay-buffer rate as a percentage from 1 through 100.

On IQ and IQ2 interfaces, you can configure the delay-buffer rate as an absolute rate from 1000 through 160,000,000,000 bits per second.

The actual delay buffer is based on the calculations described in Table 21 on page 126 and Table 22 on page 127. For an example showing how the delay-buffer rates are applied, see “Examples: Oversubscribing Interface Bandwidth” on page 158.

Configuring large buffers on relatively slow-speed links can cause packet aging. To help prevent this problem, the software requires that the sum of the delay-buffer rates be less than or equal to the port speed.

This restriction does not eliminate the possibility of packet aging, so you should be cautious when using the `delay-buffer-rate` statement. Though some amount of extra buffering might be desirable for burst absorption, delay-buffer rates should not far exceed the service rate of the logical interface.

If you configure delay-buffer rates so that the sum exceeds the port speed, the configured delay-buffer rate is not implemented for the last logical interface that you configure. Instead, that logical interface receives a delay-buffer rate of zero, and a warning message is displayed in the CLI. If bandwidth becomes available (because another logical interface is deleted or deactivated, or the port speed is increased), the configured delay-buffer-rate is reevaluated and implemented if possible.

If you do not configure a delay-buffer rate or a guaranteed rate, the logical interface receives a delay-buffer rate in proportion to the shaping rate and the remaining delay-buffer rate available. In other words, the delay-buffer rate for each logical interface with no configured delay-buffer rate is equal to:

$$(\text{remaining delay-buffer rate} * \text{shaping rate}) / (\text{sum of shaping rates})$$

where the remaining delay-buffer rate is equal to:

$$(\text{interface speed}) - (\text{sum of configured delay-buffer rates})$$

- To assign a scheduler map to the logical interface, include the `scheduler-map` statement at the [edit class-of-service traffic-control-profiles *profile-name*] hierarchy level:

```
[edit class-of-service traffic-control-profiles profile-name]
scheduler-map map-name;
```

For information about configuring schedulers and scheduler maps, see “Configuring a Scheduler” on page 119 and “Configuring the Scheduler Map” on page 139.

4. Optionally, you can enable large buffer sizes to be configured. To do this, include the `q-pic-large-buffer` statement at the `[edit chassis fpc slot-number pic pic-number]` hierarchy level:

```
[edit chassis fpc slot-number pic pic-number]
q-pic-large-buffer;
```

If you do not include this statement, the delay-buffer size is more restricted. We recommend restricted buffers for delay-sensitive traffic, such as voice traffic. For more information, see “Configuring Large Delay Buffers for Slower Interfaces” on page 124.

5. To enable scheduling on logical interfaces, include the `per-unit-scheduler` statement at the `[edit interfaces interface-name]` hierarchy level:

```
[edit interfaces interface-name]
per-unit-scheduler;
```

When you include this statement, the maximum number of VLANs supported is 767 on a single-port Gigabit Ethernet IQ PIC. On a dual-port Gigabit Ethernet IQ PIC, the maximum number is 383.

6. To apply the traffic-scheduling profile to the logical interface, include the `output-traffic-control-profile` statement at the `[edit class-of-service interfaces interfaces interface-name unit logical-unit-number]` hierarchy level:

```
[edit class-of-service interfaces interface-name unit logical-unit-number]
output-traffic-control-profile profile-name;
```

You cannot include the `output-traffic-control-profile` statement in the configuration if any of the following statements are included in the logical interface configuration: `scheduler-map`, `shaping-rate`, `adaptive-shaper`, `virtual-channel-group`.

Table 26 shows how the bandwidth and delay buffer are allocated in various configurations.

**Table 26: Bandwidth and Delay Buffer Allocations by Configuration Scenario (1 of 2)**

Configuration Scenario	Delay Buffer Allocation
You do not oversubscribe the interface. You do not configure a guaranteed rate. You do not configure a shaping rate. You do not configure a delay-buffer rate.	Logical interface receives the remaining bandwidth and receives a delay buffer in proportion to the remaining bandwidth.
You do not oversubscribe the interface. You configure a shaping rate at the <code>[edit class-of-service interfaces interface-name unit logical-unit-number]</code> hierarchy level.	For backward compatibility, the shaped logical interface receives a delay buffer based on the shaping rate. The multiplicative factor depends on whether you include the <code>q-pic-large-buffer</code> statement. For more information, see “Configuring Large Delay Buffers for Slower Interfaces” on page 124.  Unshaped logical interfaces receive the remaining bandwidth and a delay buffer in proportion to the remaining bandwidth.

**Table 26: Bandwidth and Delay Buffer Allocations by Configuration Scenario (2 of 2)**

Configuration Scenario	Delay Buffer Allocation
You oversubscribe the interface. You do not configure a guaranteed rate. You do not configure a shaping rate. You do not configure a delay-buffer rate.	Logical interface receives minimal bandwidth with no guarantees and receives a minimal delay buffer equal to 4 MTU-sized packets.
You oversubscribe the interface. You configure a shaping rate. You do not configure a guaranteed rate. You do not configure a delay-buffer rate.	<p>Logical interface receives a delay buffer based on the scaled shaping rate:</p> $\text{scaled shaping rate} = \frac{(\text{shaping-rate} * [\text{physical interface bandwidth}])}{\text{SUM (shaping-rates of all logical interfaces on the physical interface)}}$ <p>The logical interface receives variable bandwidth, depending on how much oversubscription and statistical multiplexing is present. If the amount of oversubscription is low enough that statistical multiplexing does not make all logical interfaces active at the same time and the physical interface bandwidth is not exceeded, the logical interface receives bandwidth equal to the shaping rate. Otherwise, the logical interface receives a smaller amount of bandwidth. In either case, the logical interface bandwidth does not exceed the shaping rate.</p>
You oversubscribe the interface. You configure a shaping rate. You configure a delay-buffer rate.	<p>Logical interface receives a delay buffer based on the delay-buffer rate. For example, on IQ and IQ2 interfaces:</p> <ul style="list-style-type: none"> <li>delay-buffer-rate &lt;= 10 Mbps: 400-millisecond (ms) delay buffer</li> <li>delay-buffer-rate &lt;= 20 Mbps: 300-ms delay buffer</li> <li>delay-buffer-rate &lt;= 30 Mbps: 200-ms delay buffer</li> <li>delay-buffer-rate &lt;= 40 Mbps: 150-ms delay buffer</li> <li>delay-buffer-rate &gt; 40 Mbps: 100-ms delay buffer</li> </ul> <p>On LSQ DLCIs, if total bundle bandwidth &lt; T1 bandwidth: delay-buffer-rate = 1 second</p> <p>On LSQ DLCIs, if total bundle bandwidth &gt;= T1 bandwidth: delay-buffer-rate = 200 ms</p> <p>The multiplicative factor depends on whether you include the <code>q-pic-large-buffer</code> statement. For more information, see “Configuring Large Delay Buffers for Slower Interfaces” on page 124.</p> <p>The logical interface receives variable bandwidth, depending on how much oversubscription and statistical multiplexing is present. If the amount of oversubscription is low enough that statistical multiplexing does not make all logical interfaces active at the same time and the physical interface bandwidth is not exceeded, the logical interface receives bandwidth equal to the shaping rate. Otherwise, the logical interface receives a smaller amount of bandwidth. In either case, the logical interface bandwidth does not exceed the shaping rate.</p>
You oversubscribe the interface. You do not configure a shaping rate. You configure a guaranteed rate. You configure a delay-buffer rate.	Logical interface receives a delay buffer based on the delay-buffer rate.
You oversubscribe the interface. You do not configure a shaping rate. You do not configure a guaranteed rate. You configure a delay-buffer rate.	This scenario is not allowed. If you configure a delay-buffer rate, the traffic-control profile must also include either a shaping rate or a guaranteed rate.
You oversubscribe the interface. You configure a shaping rate. You configure a guaranteed rate. You do not configure a delay-buffer rate.	<p>Logical interface receives a delay buffer based on the guaranteed rate.</p> <p>This configuration is valid on LSQ interfaces and Gigabit Ethernet IQ2 interfaces only. On channelized interfaces, you cannot configure both a shaping rate (PIR) and a guaranteed rate (CIR).</p>

## Verifying Your Configuration

To verify your configuration, you can issue this following operational mode commands:

- `show class-of-service interfaces`
- `show class-of-service traffic-control-profile profile-name`

## Examples: Oversubscribing Interface Bandwidth

This section provides the following examples:

### Oversubscribing a Channelized Interface

Two logical interface units, 0 and 1, are shaped to rates 2 Mbps and 3 Mbps, respectively. The delay-buffer rates are 750 Kbps and 500 Kbps, respectively. The actual delay buffers allocated to each logical interface are 1 second of 750 Kbps and 2 seconds of 500 Kbps, respectively. The 1- and 2-second values are based on the following calculations:

```
delay-buffer-rate < [16 x 64 Kbps]: 1 second of delay-buffer-rate
delay-buffer-rate < [8 x 64 Kbps]: 2 seconds of delay-buffer-rate
```

For more information about these calculations, see “Maximum Delay Buffer for NxDS0 Interfaces” on page 126.

```
chassis {
  fpc 3 {
    pic 0 {
      q-pic-large-buffer;
    }
  }
}
interfaces {
  t1-3/0/0 {
    per-unit-scheduler;
  }
}
class-of-service {
  traffic-control-profiles {
    tc-profile1 {
      shaping-rate 2m;
      delay-buffer-rate 750k; # 750 Kbps is less than 16 x 64 Kbps
      scheduler-map sched-map1;
    }
    tc-profile2 {
      shaping-rate 3m;
      delay-buffer-rate 500k; # 500 Kbps is less than 8 x 64 Kbps
      scheduler-map sched-map2;
    }
  }
}
```

```

interface t1-3/0/0 {
  unit 0 {
    output-traffic-control-profile tc-profile1;
  }
  unit 1 {
    output-traffic-control-profile tc-profile2;
  }
}

```

**Oversubscribing an LSQ Interface** Apply a traffic-control profile to a logical interface representing a DLCI on an FRF.16 bundle:

```

interfaces {
  lsq-1/3/0:0 {
    per-unit-scheduler;
    unit 0 {
      dlc1 100;
    }
    unit 1 {
      dlc1 200;
    }
  }
}

class-of-service {
  traffic-control-profiles {
    tc_0 {
      shaping-rate percent 100;
      guaranteed-rate percent 60;
      delay-buffer-rate percent 80;
    }
    tc_1 {
      shaping-rate percent 80;
      guaranteed-rate percent 40;
    }
  }
  interfaces {
    lsq-1/3/0 {
      unit 0 {
        output-traffic-control-profile tc_0;
      }
      unit 1 {
        output-traffic-control-profile tc_1;
      }
    }
  }
}

```

## Providing a Guaranteed Minimum Rate

On Gigabit Ethernet IQ PICs, Channelized IQ PICs, and FRF.16 LSQ interfaces on AS PICs, you can configure guaranteed bandwidth, also known as a committed information rate (CIR). This allows you to specify a guaranteed rate for each logical interface. The guaranteed rate is a minimum. If excess physical interface bandwidth is available for use, the logical interface receives more than the guaranteed rate provisioned for the interface.

You cannot provision the sum of the guaranteed rates to be more than the physical interface bandwidth, or the bundle bandwidth for LSQ interfaces. If the sum of the guaranteed rates exceeds the interface or bundle bandwidth, the commit operation does not fail, but the software automatically decreases the rates so that the sum of the guaranteed rates is equal to the available bundle bandwidth.

To configure a guaranteed minimum rate, perform the following steps:

1. Include the `guaranteed-rate` statement at the `[edit class-of-service traffic-control-profiles profile-name]` hierarchy level:

```
[edit class-of-service traffic-control-profiles profile-name]  
  guaranteed-rate (percent percentage | rate);
```

On LSQ interfaces, you can configure the guaranteed rate as a percentage from 1 through 100.

On IQ and IQ2 interfaces, you can configure the guaranteed rate as an absolute rate from 1000 through 160,000,000,000 bits per second.



**NOTE:** For channelized and Gigabit Ethernet IQ interfaces, the `shaping-rate` and `guaranteed-rate` statements are mutually exclusive. You cannot configure some logical interfaces to use a shaping rate and others to use a guaranteed rate. This means there are no service guarantees when you configure a PIR. For these interfaces, you can configure either a PIR or a CIR, but not both.

This restriction does not apply to Gigabit Ethernet IQ2 PICs or LSQ interfaces on AS PICs. For LSQ and Gigabit Ethernet IQ2 interfaces, you can configure both a PIR and a CIR on an interface. For more information about CIRs, see “Providing a Guaranteed Minimum Rate” on page 160.

For more information about Gigabit Ethernet IQ2 PICs, see “Shaping Input and Output Traffic on Ethernet IQ2 Interfaces” on page 199.

2. Optionally, you can base the delay-buffer calculation on a delay-buffer rate. To do this, include the `delay-buffer-rate` statement `[edit class-of-service traffic-control-profiles profile-name]` hierarchy level:

```
[edit class-of-service traffic-control-profiles profile-name]  
  delay-buffer-rate (percent percentage | rate);
```

On LSQ interfaces, you can configure the delay-buffer rate as a percentage from 1 through 100.

On IQ and IQ2 interfaces, you can configure the delay-buffer rate as an absolute rate from 1000 through 160,000,000,000 bits per second.

The actual delay buffer is based on the calculations described in Table 21 on page 126 and Table 22 on page 127. For an example showing how the delay-buffer rates are applied, see “Example: Providing a Guaranteed Minimum Rate” on page 163.

If you do not include the `delay-buffer-rate` statement, the delay-buffer calculation is based on the guaranteed rate, the shaping rate if no guaranteed rate is configured, or the scaled shaping rate if the interface is oversubscribed.

If you do not specify a shaping rate or a guaranteed rate, the logical interface receives a minimal delay-buffer rate and minimal bandwidth equal to four MTU-sized packets.

You can configure a rate for the delay buffer that is higher than the guaranteed rate. This can be useful when the traffic flow might not require much bandwidth in general, but in some cases traffic can be bursty and therefore will need a large buffer.

Configuring large buffers on relatively slow-speed links can cause packet aging. To help prevent this problem, the software requires that the sum of the delay-buffer rates be less than or equal to the port speed. This restriction does not eliminate the possibility of packet aging, so you should be cautious when using the `delay-buffer-rate` statement. Though some amount of extra buffering might be desirable for burst absorption, delay-buffer rates should not far exceed the service rate of the logical interface.

If you configure delay-buffer rates so that the sum exceeds the port speed, the configured delay-buffer rate is not implemented for the last logical interface that you configure. Instead, that logical interface receives a delay-buffer rate of 0, and a warning message is displayed in the CLI. If bandwidth becomes available (because another logical interface is deleted or deactivated, or the port speed is increased), the configured delay-buffer-rate is reevaluated and implemented if possible.

If the guaranteed rate of a logical interface cannot be implemented, that logical interface receives a delay-buffer rate of 0, even if the configured delay-buffer rate is within the interface speed. If at a later time the guaranteed rate of the logical interface can be met, the configured delay-buffer rate is reevaluated and if the delay-buffer rate is within the remaining bandwidth, it is implemented.

If any logical interface has a configured guaranteed rate, all other logical interfaces on that port that do not have a guaranteed rate configured receive a delay-buffer rate of 0. This is because the absence of a guaranteed rate configuration corresponds to a guaranteed rate of 0 and, consequently, a delay-buffer rate of 0.

- To assign a scheduler map to the logical interface, include the `scheduler-map` statement at the `[edit class-of-service traffic-control-profiles profile-name]` hierarchy level:

```
[edit class-of-service traffic-control-profiles profile-name]
scheduler-map map-name;
```

For information about configuring schedulers and scheduler maps, see “Configuring a Scheduler” on page 119 and “Configuring the Scheduler Map” on page 139.

- To enable large buffer sizes to be configured, include the `q-pic-large-buffer` statement at the `[edit chassis fpc slot-number pic pic-number]` hierarchy level:

```
[edit chassis fpc slot-number pic pic-number]
q-pic-large-buffer;
```

If you do not include this statement, the delay-buffer size is more restricted. For more information, see “Configuring Large Delay Buffers for Slower Interfaces” on page 124.

- To enable scheduling on logical interfaces, include the `per-unit-scheduler` statement at the `[edit interfaces interface-name]` hierarchy level:

```
[edit interfaces interface-name]
per-unit-scheduler;
```

When you include this statement, the maximum number of VLANs supported is 767 on a single-port Gigabit Ethernet IQ PIC. On a dual-port Gigabit Ethernet IQ PIC, the maximum number is 383.

- To apply the traffic-scheduling profile to the logical interface, include the `output-traffic-control-profile` statement at the `[edit class-of-service interfaces interfaces interface-name unit logical-unit-number]` hierarchy level:

```
[edit class-of-service interfaces interface-name unit logical-unit-number]
output-traffic-control-profile profile-name;
```

Table 27 shows how the bandwidth and delay buffer are allocated in various configurations.

**Table 27: Bandwidth and Delay Buffer Allocations by Configuration Scenario**

Configuration Scenario	Delay Buffer Allocation
You do not configure a guaranteed rate. You do not configure a delay-buffer rate.	Logical interface receives minimal bandwidth with no guarantees and receives a minimal delay buffer equal to 4 MTU-sized packets.
You configure a guaranteed rate. You do not configure a delay-buffer rate.	Logical interface receives bandwidth equal to the guaranteed rate and a delay buffer based on the guaranteed rate. The multiplicative factor depends on whether you include the <code>q-pic-large-buffer</code> statement. For more information, see “Configuring Large Delay Buffers for Slower Interfaces” on page 124.
You configure a guaranteed rate. You configure a delay-buffer rate.	Logical interface receives bandwidth equal to the guaranteed rate and a delay buffer based on the delay-buffer rate. The multiplicative factor depends on whether you include the <code>q-pic-large-buffer</code> statement. For more information, see “Configuring Large Delay Buffers for Slower Interfaces” on page 124.

## Verifying Your Configuration

To verify your configuration, you can issue the following operational mode commands:

- `show class-of-service interfaces`
- `show class-of-service traffic-control-profile profile-name`

## Example: Providing a Guaranteed Minimum Rate

Two logical interface units, 0 and 1, are provisioned with a guaranteed minimum of 750 Kbps and 500 Kbps, respectively. For logical unit 1, the delay buffer is based on the guaranteed rate setting. For logical unit 0, a delay-buffer rate of 500 Kbps is specified. The actual delay buffers allocated to each logical interface are 2 seconds of 500 Kbps. The 2-second value is based on the following calculation:

$$\text{delay-buffer-rate} < [8 \times 64 \text{ Kbps}]; \text{ 2 seconds of delay-buffer-rate}$$

For more information about this calculation, see “Maximum Delay Buffer for NxDSO Interfaces” on page 126.

```

chassis {
  fpc 3 {
    pic 0 {
      q-pic-large-buffer;
    }
  }
}
interfaces {
  t1-3/0/1 {
    per-unit-scheduler;
  }
}
class-of-service {
  traffic-control-profiles {
    tc-profile3 {
      guaranteed-rate 750k;
      scheduler-map sched-map3;
      delay-buffer-rate 500k; # 500 Kbps is less than 8 x 64 Kbps
    }
    tc-profile4 {
      guaranteed-rate 500k; # 500 Kbps is less than 8 x 64 Kbps
      scheduler-map sched-map4;
    }
  }
  interface t1-3/0/1 {
    unit 0 {
      output-traffic-control-profile tc-profile3;
    }
    unit 1 {
      output-traffic-control-profile tc-profile4;
    }
  }
}

```

## Associating the Scheduler Map with the Packet Forwarding Component Queues

On IQ interfaces, the traffic that is fed from the packet forwarding components into the PIC uses low PLP by default and is distributed evenly across the four chassis queues (not PIC queues), regardless of the scheduling configuration for each logical interface. This default behavior can cause traffic congestion.

To control the aggregated traffic transmitted from the chassis queues into the PIC, you can configure the chassis queues to derive their scheduling configuration from the associated logical interface's. Include the `scheduler-map-chassis derived` statement at the `[edit class-of-service interfaces type-fpc/pic]` hierarchy level:

```
[edit class-of-service interfaces type-fpc/pic/*]
scheduler-map-chassis derived;
```



**CAUTION:** If you include the `scheduler-map-chassis derived` statement in the configuration, packet loss might occur when you subsequently add or remove logical interfaces at the `[edit interfaces interface-name]` hierarchy level.

When fragmentation occurs on the egress interface, the first set of packet counters displayed in the output of the `show interfaces queue` command show the post-fragmentation values. The second set of packet counters (under the **Packet Forwarding Engine Chassis Queues** field) show the pre-fragmentation values. For more information about the `show interfaces queue` command, see the *JUNOS Interfaces Command Reference*.

---

You can include both the `scheduler-map` and the `scheduler-map-chassis derived` statements in the same interface configuration. The `scheduler-map` statement controls the scheduler inside the PIC, while the `scheduler-map-chassis derived` statement controls the aggregated traffic transmitted into the entire PIC. For the Gigabit Ethernet IQ PIC, include both statements.

For more information about the `scheduler-map` statement, see “Associating the Scheduler Map with a Physical Interface” on page 141. For information about logical interface scheduling configuration, see “Associating the Scheduler Map and a Shaping Rate with a DLCI or VLAN” on page 148.

Generally, when you include the `scheduler-map-chassis` statement in the configuration, you must use an interface wildcard for the interface name, as in `type-fpc/pic/*`. The wildcard must use this format—for example, `so-1/2/*`, which means all interfaces on FPC slot 1, PIC slot 2. There is one exception—you can apply the chassis scheduler map to a specific interface on the Gigabit Ethernet IQ PIC only.

According to JUNOS software wildcard rules, specific interface configurations override wildcard configurations. For chassis scheduler map configuration, this rule does not apply; instead, specific interface CoS configurations are added to the chassis scheduler map configuration. For more information about how wildcards work with chassis scheduler maps, see “Examples: Scheduling Packet Forwarding Component Queues” on page 165. For general information about wildcards, see the *JUNOS System Basics Configuration Guide*.

For more information, see the following sections:

- Assigning a Custom Scheduler to the Packet Forwarding Component Queues on page 165
- Examples: Scheduling Packet Forwarding Component Queues on page 165

### Assigning a Custom Scheduler to the Packet Forwarding Component Queues

Optionally, you can apply a custom scheduler to the chassis queues instead of configuring the chassis queues to automatically derive their scheduling configuration from the logical interfaces on the PIC.

To assign a custom scheduler to the packet forwarding component queues, include the `scheduler-map-chassis` statement at the [edit class-of-service interfaces *type-fpc/pic*] hierarchy level:

```
[edit class-of-service interfaces type-fpc/pic/*]
scheduler-map-chassis map-name;
```

For information about defining the scheduler map referenced by *map-name*, see “Configuring the Scheduler Map” on page 139.

### Examples: Scheduling Packet Forwarding Component Queues

#### Associating a Chassis Scheduler Map with a 2-Port IQ PIC

The two interfaces are `so-0/1/0` and `so-0/1/1`.

According to customary wildcard rules, the `so-0/1/0` configuration overrides the `so-0/1/*` configuration, implying that the chassis scheduler map `MAP1` is not applied to `so-0/1/0`. However, the wildcard rule is not obeyed in this case; the chassis scheduler map applies to both interfaces `so-0/1/0` and `so-0/1/1`.

```
class-of-service {
  interfaces {
    so-0/1/0 {
      unit 0 {
        classifiers {
          inet-precedence default;
        }
      }
    }
    so-0/1/* {
      scheduler-map-chassis derived;
    }
  }
}
```

**Not Recommended: Gigabit Ethernet IQ Example** On a Gigabit Ethernet IQ PIC, you can apply the chassis scheduler map at both the specific interface level and the wildcard level. We do not recommend this because the wildcard chassis scheduler map takes precedence, which might not be the desired effect. For example, if you want to apply the chassis scheduler map MAP1 to port 0 and MAP2 to port 1, we do not recommend the following:

```
[edit class-of-service]
interfaces {
  ge-0/1/0 {
    scheduler-map-chassis MAP1;
  }
  ge-0/1/* {
    scheduler-map-chassis MAP2;
  }
}
```

**Recommended: Gigabit Ethernet IQ Example** Instead, we recommend this:

```
[edit class-of-service]
interfaces {
  ge-0/1/0 {
    scheduler-map-chassis MAP1;
  }
  ge-0/1/1 {
    scheduler-map-chassis MAP2;
  }
}
```

**Configuring ATM CoS with a Normal Scheduler and a Chassis Scheduler** For ATM2 IQ interfaces, the CoS configuration differs significantly from that of other interface types. For more information about ATM CoS, see “Configuring CoS on ATM Interfaces” on page 257.

```
[edit class-of-service]
interfaces {
  at-1/2/* {
    scheduler-map-chassis derived;
  }
}

[edit interfaces]
at-1/2/0 {
  atm-options {
    vpi 0;
    linear-red-profiles red-profile-1 {
      queue-depth 35000 high-plp-threshold 75 low-plp-threshold 25;
    }
  }
  scheduler-maps map-1 {
    vc-cos-mode strict;
    forwarding-class best-effort {
      priority low;
      transmit-weight percent 25;
      linear-red-profile red-profile-1;
    }
  }
}
```

```

unit 0 {
  vci 0.128;
  shaping {
    vbr peak 20m sustained 10m burst 20;
  }
  atm-scheduler-map map-1;
  family inet {
    address 192.168.0.100/32 {
      destination 192.168.0.101;
    }
  }
}
}

```

**Configuring Two T3  
Interfaces on a  
Channelized DS3 IQ PIC**

```

[edit interfaces]
ct3-3/0/0 {
  no-partition interface-type t3; # use entire port 0 as T3
}
ct3-3/0/1 {
  no-partition interface-type t3; # use entire port 1 as T3
}
t3-3/0/0 {
  unit 0 {
    family inet {
      address 10.0.100.1/30;
    }
  }
}
t3-3/0/1 {
  unit 0 {
    family inet {
      address 10.0.101.1/30;
    }
  }
}
}

```

**Associating Normal  
Schedulers with the  
Two T3 Interfaces**

Configure a scheduler for the aggregated traffic transmitted into both T3 interfaces.

```

[edit class-of-service]
interfaces {
  t3-3/0/0 {
    scheduler-map sched-qct3-0;
  }
  t3-3/0/1 {
    scheduler-map sched-qct3-1;
  }
}
}

```

```

scheduler-maps {
  sched-qct3-0 {
    forwarding-class best-effort scheduler be-qct3-0;
    forwarding-class expedited-forwarding scheduler ef-qct3-0;
    forwarding-class assured-forwarding scheduler as-qct3-0;
    forwarding-class network-control scheduler nc-qct3-0;
  }
  sched-qct3-1 {
    forwarding-class best-effort scheduler be-qct3-1;
    forwarding-class expedited-forwarding scheduler ef-qct3-1;
    forwarding-class assured-forwarding scheduler as-qct3-1;
    forwarding-class network-control scheduler nc-qct3-1;
  }
  sched-chassis-to-q {
    forwarding-class best-effort scheduler be-chassis;
    forwarding-class expedited-forwarding scheduler ef-chassis;
    forwarding-class assured-forwarding scheduler as-chassis;
    forwarding-class network-control scheduler nc-chassis;
  }
}
schedulers {
  be-qct3-0 {
    transmit-rate percent 40;
  }
  ef-qct3-0 {
    transmit-rate percent 30;
  }
  as-qct3-0 {
    transmit-rate percent 20;
  }
  nc-qct3-0 {
    transmit-rate percent 10;
  }
  ...
}

```

**Associating a Chassis Scheduler with the Two T3 Interfaces**

Bind a scheduler to the aggregated traffic transmitted into the entire PIC. The chassis scheduler controls the traffic from the packet forwarding components feeding the interface t3-3/0/\*.

```

[edit class-of-service]
interfaces {
  t3-3/0/* {
    scheduler-map-chassis derived;
  }
}

```

## Default Fabric Priority Queuing

On M320 and T-series platforms, the default behavior is for fabric priority queuing on egress interfaces to match the scheduling priority you assign. High-priority egress traffic is automatically assigned to high-priority fabric queues. Likewise, low-priority egress traffic is automatically assigned to low-priority fabric queues.

For information about overriding automatic fabric priority queuing, see “Overriding Fabric Priority Queuing” on page 90 and “Associating a Scheduler with a Fabric Priority” on page 169.

## Associating a Scheduler with a Fabric Priority

On M320 and T-series platforms only, you can associate a scheduler with a class of traffic that has a specific priority while transiting the fabric. Traffic transiting the fabric can have two priority values: **low** or **high**. To associate a scheduler with a fabric priority, include the **priority** and **scheduler** statements at the [edit class-of-service fabric scheduler-map] hierarchy level:

```
[edit class-of-service fabric scheduler-map]
priority (high | low) scheduler scheduler-name;
```



**NOTE:** For a scheduler that you associate with a fabric priority, you can include only the **drop-profile-map** statement at the [edit class-of-service schedulers *scheduler-name*] hierarchy level. You cannot include the **buffer-size**, **transmit-rate**, and **priority** statements at that hierarchy level.

For information about associating a forwarding class with a fabric priority, see “Overriding Fabric Priority Queuing” on page 90.

### Example: Associating a Scheduler with a Fabric Priority

Associate a scheduler with a class of traffic that has a specific priority while transiting the fabric:

```
[edit class-of-service]
schedulers {
  fab-be-scheduler {
    drop-profile-map loss-priority low protocol any drop-profile fab-profile-1;
    drop-profile-map loss-priority high protocol any drop-profile fab-profile-2;
  }
  fab-ef-scheduler {
    drop-profile-map loss-priority low protocol any drop-profile fab-profile-3;
    drop-profile-map loss-priority high protocol any drop-profile fab-profile-4;
  }
}
drop-profiles {
  fab-profile-1 {
    fill-level 100 drop-probability 100;
    fill-level 85 drop-probability 50;
  }
  fab-profile-2 {
    fill-level 100 drop-probability 100;
    fill-level 95 drop-probability 50;
  }
}
```

```

fab-profile-3 {
  fill-level 75 drop-probability 100;
  fill-level 95 drop-probability 50;
}
fab-profile-4 {
  fill-level 100 drop-probability 100;
  fill-level 80 drop-probability 50;
}
}
fabric {
  scheduler-map {
    priority low scheduler fab-be-scheduler;
    priority high scheduler fab-ef-scheduler;
  }
}

```

## Configuring the Number of Schedulers for Ethernet IQ2 PICs

---

You can oversubscribe the Ethernet IQ2 family of PICs. Because of the bursty nature of Ethernet use, traffic received by the PIC can be several orders of magnitude greater than the maximum bandwidth leaving the PIC and entering the router. Several configuration statements apply only to Ethernet IQ2 PICs and allow the PIC to intelligently handle the oversubscribed traffic.

This section discusses the following topics:

- Ethernet IQ2 PIC Schedulers on page 170
- Example: Configuring a Scheduler Number for an Ethernet IQ2 PIC Port on page 171

### ***Ethernet IQ2 PIC Schedulers***

By default, each Ethernet IQ2 PIC is allocated a fixed number of the 1024 available schedulers for each port during PIC initialization. For example, the 8-port Gigabit Ethernet IQ2 PIC is allocated 128 schedulers for each port. This number cannot be changed after the PIC is operational and can limit the utilization of shapers among the ports.

Three schedulers are reserved on each port. One is for control traffic, one is for port-level shaping, and the last is for unshaped logical interface traffic. These are allocated internally and automatically.

When you configure schedulers for a port on an Ethernet IQ2 PIC:

- The three reserved schedulers are added to the configured value.
- The configured value is adjusted upward to the nearest multiple of 4 (schedulers are allocated in multiples of 4).

- After all configured schedulers are allocated, any remaining unallocated schedulers are partitioned equally across the other ports.
- Any remaining schedulers that cannot be allocated meaningfully across the ports are allocated to the last port.

If the configured scheduler number is changed, the Ethernet IQ2 PIC will be restarted when the configuration is committed.

To configure the number of schedulers assigned to a port on an Ethernet IQ2 PIC, include the `schedulers` statement for the Ethernet IQ2 PIC interface at the `[edit interfaces interface-name]` hierarchy level:

```
[edit interfaces ge-1/2/1]
schedulers number;
```

You can configure between 1 and 1024 schedulers on a port.

### **Example: Configuring a Scheduler Number for an Ethernet IQ2 PIC Port**

This example allocates 100 schedulers to port 1 on an 8-port Gigabit Ethernet IQ2 PIC. The example shows the final scheduler allocation numbers for each port on the PIC. By default, each port would have been allocated  $1024 / 4 = 128$  schedulers.

```
[edit interfaces]
ge-1/2/1 {
    schedulers 100;
}
```

This configuration results in the port and scheduler configuration shown in Table 28.

**Table 28: Scheduler Allocation for an Ethernet IQ2 PIC**

Ethernet IQ2 PIC Port	Number of Allocated Schedulers
0	128
1	104 (100 configured, plus 3 reserved, rounded up to multiple of 4: $100 + 3 + 1 = 104$ )
2	128
3	128
4	128
5	128
6	128
7	152 (128 plus the 24 remaining that cannot be meaningfully allocated to other ports)

\

