Configuring Hierarchical Modular QoS

Hierarchical QoS allows you to specify QoS behavior at multiple policy levels, which provides a high degree of granularity in traffic management.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Enhanced Hierarchical Ingress Policing	no	yes
Hierarchical Policing	yes	yes
Hierarchical QoS	yes	yes
Three-Parameter Scheduler	yes	yes

Feature History for Hierarchical QoS on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.1	The Hierarchical Policing feature was introduced on Cisco ASR 9000 Series Routers on ASR 9000 Ethernet Line Cards. The Hierarchical QoS feature was introduced on Cisco ASR 9000 Series Routers on ASR 9000 Ethernet Line Cards. The Three-Parameter Scheduler feature was introduced on Cisco ASR 9000 Series Routers on ASR 9000 Ethernet Line Cards.
Release 3.9.0	The Hierarchical QoS feature was supported on the SIP 700 for the ASR 9000. (two-level policies only)

Release 4.0.0	The Enhanced Hierarchical Ingress Policing feature was introduced on Cisco ASR 9000 Series Routers on the SIP 700 for the ASR 9000. The Hierarchical Policing feature was supported on Cisco ASR 9000 Series Routers on the SIP 700 for the ASR 9000. For the Hierarchical QoS feature, support was added for three-level policies on the SIP 700 for the ASR 9000. The Three-Parameter Scheduler feature was supported on the SIP 700 for the ASR 9000.

- How to Configure Hierarchical QoS, on page 2
- Verifying the Configuration of Hierarchical Policies, on page 23
- Additional References, on page 23

How to Configure Hierarchical QoS

When configuring hierarchical QoS, consider the following guidelines:

- When defining polices, start at the bottom level of the hierarchy. For example, for a two-level hierarchical policy, define the bottom-level policy and then the top-level policy. For a three-level hierarchical policy, define the bottom-level policy, the middle-level policy, and then the top-level policy.
- Do not specify the input or output keyword in the service-policy command when configuring a bottom-level policy within a top-level policy.
- Configure bottom-level policies only in middle-level and top-level policies.
- When you attach an undefined policy as a child policy, a policy-map (with only class-default) is created.

Service Fragment on LACP

- Supports only physical and bundle interfaces. No support on BVI, Satellite, and BNG.
- All sub interface policys in a port with service-fragment policy must refer to one of the service fragments in port policy.
- You must perform removal of sub-interface policy before port policy.

Port policy configurations - Defining a service fragment

This configuration task explains how to define a service fragment in a port policy. The **service-fragment** command, in the policy map configuration mode helps define the service fragment.

Aspects need to be considered while defining a service-fragment are:

- All service fragment names must be unique in a port policy. However, same names can be reused across policies.
- A class in a port policy which defines a service fragment can only specify shape, BWRR (Budgeted Weighted Round Robin), and child policy actions. Only flat policies are supported at port level.

- In a 2-level policy, only a child policy can define service fragments. A parent policy can not define service fragments and should have one class with only shape actions.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-map-name*
3. **class** *class-name*
4. **service-fragment** *name*
5. **exit**
6. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**service-fragment** *name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)#` **`service-fragment s1`**	Defines a service-fragment. The defined service fragment (s1) will be referred to for the sub-interface policy configuration.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)#` **`exit`**	Returns the router to policy map configuration mode.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Configuring sub-interface policy

This configuration task explains configuring sub-interface policy using the **fragment** command. The **fragment** command refers to the previously configured service-fragment and has to be applied on the corresponding port.

Sub-interface policy limitations:

- Sub-interface policies need to refer to a service-fragment in the parent policy in a 2-level sub-interface policy.
- The sub-interface policy actions in a parent policy should not have shape, policy, bandwidth actions in percentages (only in absolute numbers).

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-map-name*
3. **class** *class-name*
4. **fragment** *name*
5. **exit**
6. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name*	Enters policy map class configuration mode.

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	• Specifies the name of the class whose policy you want to create or change.
Step 4	**fragment** *name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# fragment s1`	Refers to a previously defined service-fragment (here, s1 is the defined service-fragment).
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Applying a service fragment policy on a physical interface

To apply a qos policy on an interface, use the **service-fragment-parent** command. This can be used only after a service-fragment policy is defined on a port.

SUMMARY STEPS

1. **configure**
2. **interface** *interface-path-id*
3. **service-policy** { **input** | **output** | **type** } **service-fragment-parent**
4. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**interface** *interface-path-id* **Example:** `RP/0/RSP0/CPU0:router (config) # interface gig 0/1/0/22`	Specifies the interface for which the service-policy is being defined.
Step 3	**service-policy** { **input** | **output** | **type** } **service-fragment-parent** **Example:** `RP/0/RSP0/CPU0:router (config-if) # service-policy input s1 service-fragment-parent`	Applies the service policy on the defined service-fragment.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Configuring the Three-Parameter Scheduler

When configuring the Three-Parameter Scheduler, consider the following guidelines:

- To use the three-parameter scheduler, a queueing class must be enabled. To enable a queueing class, you must configure at least one of the three parameters. When at least one parameter is configured, a queue is assigned to the class.
- If you configure only one parameter, the scheduler uses default values for the other two parameters.
- You can configure all 3 parameters in the same class.
- Minimum bandwidth must be less than maximum bandwidth.
- You can configure the three-parameter scheduler on the second generation of ASR 9000 Series Carrier Ethernet line cards and the third generation of ASR 9000 Series High Density Ethernet line cards.

ASR 9000 Ethernet Line Cards

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*

3. **class** *class-name*
4. **shape average** {**percent** *percentage* | *rate* [*units*]}
5. **exit**
6. **policy-map** *policy-name*
7. **class class-default**
8. **bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*] **or shape average** {**percent** *percentage* | *rate* [*units*]}
9. **service-policy** *policy-map-name*
10. **end**
11. or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map bottom-child`	Creates or modifies the bottom-level policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class Bronze`	Assigns the traffic class that you specify to the policy map. Enters policy map class configuration mode.
Step 4	**shape average** {**percent** *percentage*	*rate* [*units*]} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average 1 mbps`	Shapes traffic to the indicated bit rate.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Exits policy map class configuration mode.
Step 6	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# policy-map Top-Parent`	Creates or modifies the top-level policy.

	Command or Action	Purpose
Step 7	**class class-default** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Configures or modifies the parent class-default class. **Note** • You can configure only the class-default class in a parent policy. Do not configure any other traffic class.

Step 8	**bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*] **or shape average** {**percent** *percentage* | *rate* [*units*]}
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	`or`
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`
	`or`
	`RP/0/RSP0/CPU0:router(config-pmap-c)# shape average percent 50`
		Specifies the minimum bandwidth allocated to a class as a percentage of link bandwidth.
		Specifies how to allocate excess bandwidth to a class.
		Specifies maximum bandwidth as a percentage of link bandwidth (when other classes are not using all of their bandwidth share).
		Note
		• You must configure at least one of the three parameters.

Step 9	**service-policy** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy Bottom-Child`	Applies a bottom-level policy to the top-level class-default class.
Step 10	**end**	

Step 11	or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

SIP 700 for the ASR 9000

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*] **or shape average** {**percent** *percentage* | *rate* [*units*]}
5. **exit**
6. **policy-map** *policy-name*
7. **class class-default**
8. **shape average** {**percent** *percentage* | *rate* [*units*]}
9. **service-policy** *policy-map-name*
10. **end**
11. or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map bottom-child`	Creates or modifies the bottom-level policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class Bronze`	Assigns the traffic class that you specify to the policy map. Enters policy map class configuration mode.

Step 4	**bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*] **or shape average** {**percent** *percentage* | *rate* [*units*]}
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	`or`
		Specifies the minimum bandwidth allocated to a class as a percentage of link bandwidth.
		Specifies how to allocate excess bandwidth to a class.
		Specifies maximum bandwidth as a percentage of link bandwidth (when other classes are not using all of their bandwidth share).
		Note

	Command or Action	Purpose

	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# shape average percent 50`
		• You must configure at least one of the three parameters.

Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Exits policy map class configuration mode.
Step 6	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# policy-map Top-Parent`	Creates or modifies the top-level policy.
Step 7	**class class-default** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Configures or modifies the parent class-default class. **Note** • You can configure only the class-default class in a parent policy. Do not configure any other traffic class.
Step 8	**shape average** {**percent** *percentage* | *rate* [*units*]} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average 1 mbps`	(Optional) Shapes traffic to the indicated bit rate.
Step 9	**service-policy** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy Bottom-Child`	Applies a bottom-level policy to the top-level class-default class.
Step 10	**end**	

Step 11	or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

	Command or Action	Purpose
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Attaching Hierarchical Policies to Physical and Virtual Links

To attach hierarchical policies to interfaces, subinterfaces, virtual circuits, and virtual LANs, use the **service-policy** {**input** | **output**} *policy-map-name* command.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **service-policy {input | output}** *policy-map-name*
4. **end**
5. or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface pos 0/2/0/0`	Specifies the interface to attach the hierarchical policy.
Step 3	**service-policy {input | output}** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy input All_Traffic`	Attaches the policy map you specify. • **input**—Apply the QoS policy to inbound packets. • **output**—Apply the QoS policy to outbound packets. • *policy-map-name*—Name of a previously configured top-level policy map
Step 4	**end**	

	Command or Action	Purpose

Step 5	or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Enhanced Hierarchical Ingress Policing

The difference between configuring enhanced hierarchical ingress policing and configuring hierarchical ingress policing is the addition of the child-conform-aware command.

When used in the parent policer, the child-conform-aware command prevents the parent policer from dropping any ingress traffic that conforms to the maximum rate specified in the child policer.

Restrictions

Enhanced Hierarchical Ingress Policing has the following limitations:

- Sum of all child policer rates cannot be greater than the parent policer rate.
- Single-rate two-color policer (color blind) only.
- Configurations that specify burst size in the **police rate** command are supported; configurations that specify peak burst become single-rate three-color policers and are therefore rejected.
- Configure the **child-conform-aware** command only in the parent policer.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **service-policy** *policy-map-name*
5. **police rate** {*value* [*units*] | **percent** *percentage*} [**burst** *burst-size* [*burst-units*]] [**peak-rate** *value* [*units*]] [**peak-burst** *peak-burst* [*burst-units*]]

6. **child-conform-aware**
7. **conform-action** [**drop** | **set** *options* | **transmit]**
8. **exceed-action** [**drop** | **set** *options* | **transmit**]
9. **end** or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map parent`	Enters policy map configuration mode. Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Enters policy map class configuration mode. Specifies the name of the class whose policy you want to create or change.
Step 4	**service-policy** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy child`	Applies the bottom-level policy map to the parent class-default class. **Note** • Do not specify an input or output keyword.
Step 5	**police rate** {*value* [*units*] | **percent** *percentage*} [**burst** *burst-size* [*burst-units*]] [**peak-rate** *value* [*units*]] [**peak-burst** *peak-burst* [*burst-units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate percent 50`	Configures traffic policing and enters policy map police configuration mode.
Step 6	**child-conform-aware** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# child-conform-aware`	Prevents the parent policer from dropping any ingress traffic that conforms to the maximum rate specified in a child policer.
Step 7	**conform-action** [**drop** | **set** *options* | **transmit]** **Example:**	Configures the action to take on packets that conform to the rate limit. The allowed action is:

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action transmit`	**transmit**—Transmits the packets.
Step 8	**exceed-action** [**drop** | **set** *options* | **transmit**] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	Configures the action to take on packets that exceed the rate limit. The allowed action is: **drop**—Drops the packet.

Step 9	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Two-Level Hierarchical Queueing Policy: Example

The following example shows a two-level policy applied at the Multilink Frame Relay main interface. The same policy can be applied at Multilink PPP main interface.

```
class-map match-any video
 match precedence 1
 end-class-map
!
class-map match-any premium
 match precedence 2 3
 end-class-map
!
class-map match-any voice-ip
 match precedence 0
 end-class-map
!
class-map match-any best-effort
 match precedence 4
 end-class-map
```

```
policy-map parent_shape
 class class-default
  service-policy child_policy
  shape average percent 90
 !
 end-policy-map
!

policy-map child_policy
 class voice-ip
  priority level 1
  police rate percent 20
  !
 !
 class video
  bandwidth percent 40
 !
 class premium
  bandwidth percent 10
  random-detect precedence 2 10 ms 100 ms
  random-detect precedence 3 20 ms 200 ms
  queue-limit 200 ms
 !
 class best-effort
  bandwidth percent 20
  queue-limit 200 ms
 !
 class class-default
 !
 end-policy-map
!

interface Multilink0/2/1/0/1
 service-policy output parent_shape
 encapsulation frame-relay
 frame-relay intf-type dce
```

Three-Level Hierarchical Queueing Policy: Examples

Three-Level Hierarchical Queueing Policy: Examples

In this example, policy grand-parent is applied to the main Ethernet interface. The grand-parent policy limits all outbound traffic of the interface up to 500 Mbps. The parent policy has class vlan1 and vlan2, and traffic in vlan1 or vlan2 is limited to 40 percent of 500 Mbps. The policy child_policy classifies traffic based on different services and allocates bandwidth for each class accordingly.

```
class-map match-any video
 match precedence 1
 end-class-map
!
class-map match-any premium
 match precedence 2 3
 end-class-map
!
class-map match-any voice-ip
 match precedence 0
 end-class-map
!
class-map match-any best-effort
 match precedence 4
```

```
 end-class-map

class-map match-any vlan1
 match vlan 1
 end-class-map

class-map match-any vlan2
 match vlan 2
 end-class-map

policy-map grand-parent
class class-default
 shape average 500 Mbps
 service-policy parent
 !
 end-policy-map

policy-map parent
 class vlan1
  service-policy child_policy
  shape average percent 40
 !
 class vlan2
  service-policy child_policy
  shape average percent 40
 !
 end-policy-map
!

policy-map child_policy
 class voice-ip
  priority level 1
  police rate percent 20
  !
 !
 class video
  bandwidth percent 40
 !
 class premium
  bandwidth percent 10
  random-detect precedence 2 10 ms 100 ms
  random-detect precedence 3 20 ms 200 ms
  queue-limit 200 ms
 !
 class best-effort
  bandwidth percent 20
  queue-limit 200 ms
 !
 class class-default
 !
 end-policy-map

 interface GigabitEthernet0/0/0/9
 service-policy output grand-parent
```

SIP 700 for the ASR 9000

In this example, the policy parent_policy is applied to the Multilink Frame Relay main interface. The policy parent_policy has two classes, which match on Frame Relay DLCIs. The Multilink Frame Relay main interface has two Frame Relay PVCs configured (DLCI 16, DLCI 17).

```
interface Multilink0/2/1/0/1
```

```
 mtu 1504
 service-policy output parent_policy
 encapsulation frame-relay
 frame-relay intf-type dce
!

policy-map parent_policy
 class parentQ_1
  service-policy child_queuing_policy
  shape average 64 kbps
 !
 class parentQ_2
  service-policy child_queuing_policy
  shape average 1 mbps
 !
 class class-default
 !
 end-policy-map
!

class-map match-any parentQ_1 <----- class map parent class dlci=16
 match frame-relay dlci 16
 end-class-map
!

class-map match-any parentQ_2 <----- class map parent class dlci=17
 match frame-relay dlci 17
 end-class-map
!

interface Multilink0/2/1/0/1.16 point-to-point <------ dlci 16 pvc config
 ipv4 address 192.1.1.1 255.255.255.0
 pvc 16
  encap cisco
 !
!
interface Multilink0/2/1/0/1.17 point-to-point <------ dlci 17 pvc config
 ipv4 address 192.1.2.1 255.255.255.0
 pvc 17
  encap cisco
 !
!
policy-map child_queuing_policy <--------- child policy map
 class voice-ip
  priority level 1
  police rate percent 20
  !
 !
 class video
  bandwidth percent 40
 !
 class premium
  service-policy gchild_policy
  bandwidth percent 10
  random-detect discard-class 2 10 ms 100 ms
  random-detect discard-class 3 20 ms 200 ms
  queue-limit 200 ms
 !
 class best-effort
  bandwidth percent 20
  queue-limit 200 ms
 !
 class class-default
 !
```

```
 end-policy-map
!

policy-map gchild_policy <-------- grandchild policy map
 class premium_g1
  police rate percent 10
  !
  set discard-class 2
 !
 class premium_g2
  police rate percent 50
  !
  set discard-class 3
 !
 class class-default
 !
 end-policy-map
!

show run class-map  <----------- shows all class-map configs
Mon Aug  2 11:35:19.479 UTC
class-map match-any video
 match precedence 1
 end-class-map
!
class-map match-any premium
 match precedence 2 3
 end-class-map
!
class-map match-any voice-ip
 match precedence 0
 end-class-map
!
class-map match-any parentQ_1
 match frame-relay dlci 16
 end-class-map
!
class-map match-any parentQ_2
 match frame-relay dlci 17
 end-class-map
!
class-map match-any premium_g1
 match precedence 2
 end-class-map
!
class-map match-any premium_g2
 match precedence 3
 end-class-map
!
class-map match-any best-effort
 match precedence 4
 end-class-map
```

Three-Parameter Scheduler: Examples

Three-Parameter Scheduler: Examples

This example shows how to configure a three-parameter scheduler in a two-level hierarchical policy.

```
policy-map Bottom-ChildA
class A1
```

```
      shape average 400 kbps
class A2
     shape average 400 kbps

policy-map Bottom-ChildB
class B1
      shape average 250 kbps
class B2
     shape average 450 kbps

policy-map Top-Parent
 class parentA
  shape average 500 kbps
  bandwidth percent 30
  bandwidth remaining percent 80
 service-policy Bottom-ChildA
class parentB
  shape average 500 kbps
  bandwidth percent 60
  bandwidth remaining percent 10
 service-policy Bottom-ChildB
```

SIP 700 for the ASR 9000

This example shows how to configure a three-parameter scheduler in a two-level hierarchical policy.

```
policy-map Bottom-Child
 class A
   bandwidth percent 30
   bandwidth remaining percent 80
   shape average percent 50
 class B
   bandwidth percent 60
   bandwidth remaining percent 10
class class-default
exit

policy-map Top-Parent
 class-default
   shape average 1 mbps
 service-policy Bottom-Child
```

Hierarchical Policing: Examples

Hierarchical Policing: Examples

This example shows a two-level policy with police actions at each level. There are two classes in the top level, one for each customer. Aggregated traffic from each customer is subject to a rate limit as specified by the **police rate** command in the top level. Traffic in different classes in the bottom level is limited by an additional set of police actions to control different types of traffic for each customer.

```
class-map match-any customera
 match vlan 10-14
class-map match-any customerb
 match vlan 15-19
class-map match-any prec1
 match precedence 1
class-map match-any prec3
 match precedence 3
```

```
policy-map parent
 class customera
  service-policy childa
  bandwidth remaining ratio 10
  police rate percent 50
    conform-action transmit
    exceed-action drop
 class customerb
  service-policy childb
  bandwidth remaining ratio 100
  police rate percent 70
    conform-action transmit
    exceed-action drop

policy-map childa
 class prec1
  police rate percent 25
  conform-action transmit
  exceed-action drop
 class prec3
  police rate percent 25
   conform-action transmit
   exceed-action drop

policy-map childb
 class prec1
  police rate percent 30
  conform-action transmit
  exceed-action drop
 class prec3
  police rate percent 30
   conform-action transmit
   exceed-action drop
```

SIP 700 for the ASR 9000

In this example, policers are specified in the policy child in class Prec1 and class Prec3, and also in the class-default in the policy parent. The policers in the child policy, police traffic in class Prec1 at 30 percent (of 50 percent), police traffic in class Prec3 at 60 percent (of 50 percent) and police any other traffic at 10 percent (of 50 percent). Cumulatively, all traffic on the interface is policed at 50 percent of the interface rate by the policer in the parent policy.

```
class-map match-any prec1
    match precedence 1

class-map match-any prec3
    match precedence 3

policy-map parent
    class class-default
       service-policy child
           police rate percent 50
        conform-action transmit
        exceed-action drop
policy-map child
    class prec1
       police rate percent 30
           conform-action transmit
           exceed-action drop
    class prec3
       police rate percent 60
```

```
        conform-action transmit
        exceed-action drop
    class class-default
       police rate percent 10
        conform-action transmit
        exceed-action drop
```

Attaching Service Policies to Physical and Virtual Links: Examples

Physical Link: Example

In this example, the p1 policy is applied to a Gigabit Ethernet interface:

```
interface gigabitethernet 0/2/0/0
service-policy input p1
```

Virtual Link: Example

In this example, the p2 policy is applied to the private virtual circuit (PVC) under a multilink Frame Relay subinterface. A QoS policy can be applied only to a PVC under a Frame Relay subinterface; it cannot be applied directly to a Frame Relay subinterface.

```
interface Multilink0/2/1/0/1.16 point-to-point
 encapsulation frame-relay
 ipv4 address 192.1.1.1 255.255.255.0
 pvc 16
  service-policy output p2
  encap cisco
```

Service Fragment on LACP: Examples

The following example displays the service-fragment premium being created on LACP.

```
policy-map tsqos-port-policy
    class class-default
        shape 500 mbps
    class dscp1
        shape 1 Gbps
        service-fragment premium
    class dscp0
        shape average 100 mbps
        service-fragment sga
```

This example shows the service-fragment premium being referred (at the sub-interface):

```
policy-map tsqos-subif-policy-premium
class class-default
fragment premium
shape 20 mbps
bandwidth remaining ratio 20
service-policy subif-child
end-policy
exit
```

Service Fragment Configurations: Example

This example shows the service-fragment premium being created.

```
policy-map tsqos-port-policy
    class class-default
        shape 500 mbps
    class dscp1
        shape 1 Gbps
        service-fragment premium
        end-policy
      exit
```

This example shows the service-fragment premium being referred (at the sub-interface):

```
policy-map tsqos-subif-policy-premium
    class class-default
        fragment premium
        shape 20 mbps
        bandwidth remaining ratio 20
        service-policy subif-child
        end-policy
       exit
```

Enhanced Hierarchical Ingress Policing: Example

This example shows parent and child policies in which two classes are defined in the child policy. In class AF1, the exceed action is set to an action other than to drop traffic.

If the child-conform-aware command were not configured in the parent policy, the parent policer would drop traffic that matches the conform rate of the child policer but exceeds the conform rate of the parent policer.

When used in the parent policer, the child-conform-aware command prevents the parent policer from dropping any ingress traffic that conforms to the committed rate specified in the child policer.

In this example, class EF in the child policy is configured with a committed rate of 1 Mbps, a conform action and an exceed action. The traffic that is below 1 Mbps is presented to the parent policer with the MPLS EXP bit set to 4, and traffic that exceeds 1 Mbps is dropped.

Class AF1 in the child policy is configured with a committed rate of 1 Mbps, a conform action and an exceed action. The traffic that is below 1 Mbps is presented to the parent policer with the MPLS EXP bit set to 3, and traffic that exceeds 1 Mbps is presented to the parent policer with the MPLS EXP bit set to 2.

With this child policy configuration, the parent policer sees traffic from the child classes as exceeding its committed rate of 2 Mbps. Without the **child-conform-aware** command in the parent policer, the parent polices to 2 Mbps, which can result into dropping some conformed traffic from class EF in the child policy. When the **child-conform-aware** command is configured in the parent policer, the parent policer does not drop any traffic that conforms under the child policy.

```
policy-map parent
  class class-default
    service-policy child
    police rate 2 mbps
      child-conform-aware
      conform-action transmit
      exceed-action drop

policy-map child
  class EF
    police rate 1 mbps
```

```
    conform-action set mpls experimental imposition 4
    exceed-action drop
class AF1
  police rate 1 mbps
    conform-action set mpls experimental imposition 3
    exceed-action set mpls experimental imposition 2
```

Verifying the Configuration of Hierarchical Policies

To verify hierarchical policies, enter any of the following commands in privileged EXEC mode:

show policy-map interface	Displays policy configuration information for all classes configured for all service policies on the specified interface.
show qos interface	Displays QoS information for all classes in the service policy that is attached to the specified interface.
show running-config class-map	Displays the configuration of all class maps configured on the router.
show running-config policy-map	Displays the configuration of all policy maps configured on the router.
show running-config policy-map policy-map-name	Displays the configuration of all classes contained in the policy map you specify.

Additional References

The following sections provide references related to implementing Hierarchical QoS.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Gett Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Mod of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Ro of Cisco Cisco ASR 9000 Series Aggregation Services R Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator found at the following URL and choose a under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html