

Information About QoS for Etherchannels

- Etherchannel with QoS Feature Evolution, on page 1
- Understanding Fragments in Class Definition Statements, on page 2
- Fragments for Gigabit Etherchannel Bundles, on page 3
- QoS: Policies Aggregation MQC, on page 4
- Differences Between the Original Feature and the MQC Support for Multiple Queue AggregationDifferences Between Policy Aggregation—Egress MQC Queuing at Subinterface and the MQC Support for Multiple Queue Aggregation at Main Interface, on page 4
- How to Configure QoS for Etherchannels, on page 5
- Configuration Examples for QoS for Etherchannels, on page 23
- Additional References, on page 25
- Feature Information for Quality of Service for Etherchannel Interfaces, on page 26

Etherchannel with QoS Feature Evolution

An Etherchannel is a port-channel architecture that allows grouping of several physical links to create one logical Ethernet link for the purpose of providing fault tolerance, and high-speed links between switches, routers, and servers. An Etherchannel can be created from between two and eight active Fast, Gigabit, or 10-Gigabit Ethernet ports, with an additional one to eight inactive (failover) ports, which become active as the other active ports fail.

QoS for Etherchannel interfaces has evolved over several Cisco IOS XE releases. It is important to understand what level of support is allowed for your current level of Cisco IOS XE software and underlying Etherchannel configuration. Various combinations of QoS are supported based on how Etherchannel is configured. There are three different modes in which Etherchannel can be configured:

- Etherchannel VLAN-based load balancing via port-channel subinterface encapsulation CLI
- Etherchannel Active/Standby with LACP (no Etherchannel load balancing)
- Etherchannel with LACP with load balancing

Each of these models has specific restrictions regarding which levels of Cisco IOS XE software include support and the possible QoS configurations with each.

The following summarizes the various Etherchannel and QoS configuration combinations that are supported. Example configurations will be provided later in this document. Unless specifically mentioned together, the

combination of service policies in different logical and physical interfaces for a given Etherchannel configuration is not supported.

Etherchannel VLAN-Based Load Balancing via Port-Channel Subinterface Encapsulation CLI

Supported in Cisco IOS XE Release 2.1 or later:

- Egress MQC Queuing Configuration on Port-Channel Subinterface
- Egress MQC Queuing Configuration on Port-Channel Member Link
- QoS Policies Aggregation—Egress MQC Queuing at Subinterface
- Ingress Policing and Marking on Port-Channel Subinterface
- Egress Policing and Marking on Port-Channel Member Link

Supported in Cisco IOS XE Release 2.6 or later:

- QoS Policies Aggregation—MQC Support for Multiple Queue Aggregation at Main Interface - Egress MQC Queuing at Main Interface

Etherchannel Active/Standby with LACP (No Etherchannel Load Balancing)

Supported in Cisco IOS XE 2.4 or later:

- Egress MQC Queuing on Port-Channel Member Link—No Etherchannel Load Balancing

Etherchannel with LACP and Load Balancing

Supported in Cisco IOS XE 2.5 or later:

- Egress MQC Queuing Configuration on Port-Channel Member Link—Etherchannel Load Balancing

Supported in Cisco IOS XE 3.12 or later:

- General MQC QoS support on Port-channel main-interface

  We recommend that as a best practice for QoS, that you use port-channel aggregation—see the "Aggregate EtherChannel Quality of Service" chapter.

Supported in Cisco IOS XE 3.16.3 or later and in Cisco IOS XE Fuji 16.3 or later:

- General MQC QoS support on Port-channel sub-interface

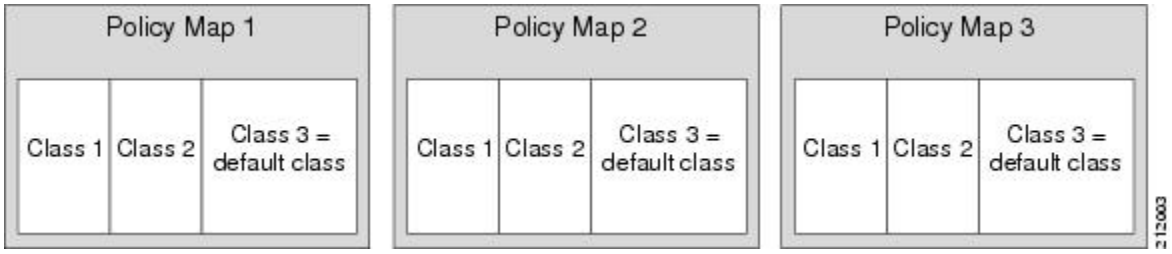
  We recommend that as a best practice for QoS, that you use port-channel aggregation—see the "Aggregate EtherChannel Quality of Service" chapter.

Understanding Fragments in Class Definition Statements

The QoS Policies Aggregation feature introduces the idea of fragments in class definition statements. A default traffic class definition statement can be marked as a fragment within a policy-map. Other policy-maps on the same interface can also define their default traffic class statements as fragments, if desired. A separate policy-map can then be created with a service fragment class definition statement that will be used to apply QoS to all of the fragments as a single group.

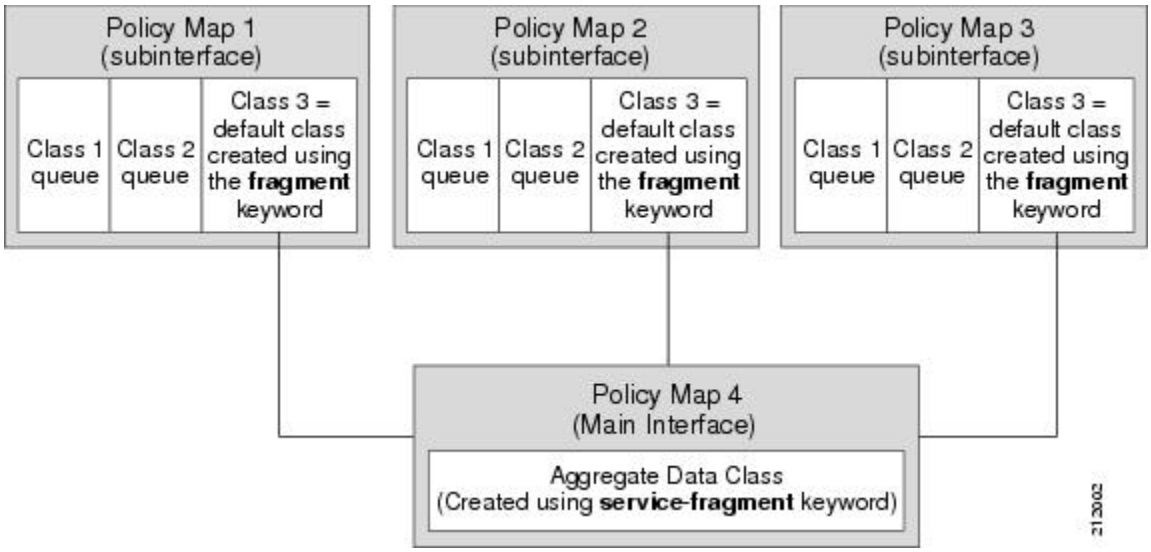
The figure below provides an example of one physical interface with three attached policy-maps that is not using fragments. Note that each policy-map has a default traffic class that can classify traffic only for the default traffic within its own policy-map.

Figure 1: Physical Interface with Policy-Maps—Not Using Fragments



The figure below shows the same configuration configured with fragments, and adds a fourth policy-map with a class definition statement that classifies the fragments collectively. The default traffic classes are now classified as one service fragment group rather than three separate default traffic classes within the individual policy-maps.

Figure 2: Physical Interface with Policy-Maps—Using Fragments



Fragments for Gigabit Etherchannel Bundles

When fragments are configured for Gigabit Etherchannel bundles, the policy-maps that have a default traffic class configured using the **fragment** keyword are attached to the member subinterface links, and the policy-maps that have a traffic class configured with the **service-fragment** keyword to collectively classify the fragments is attached to the physical interface.

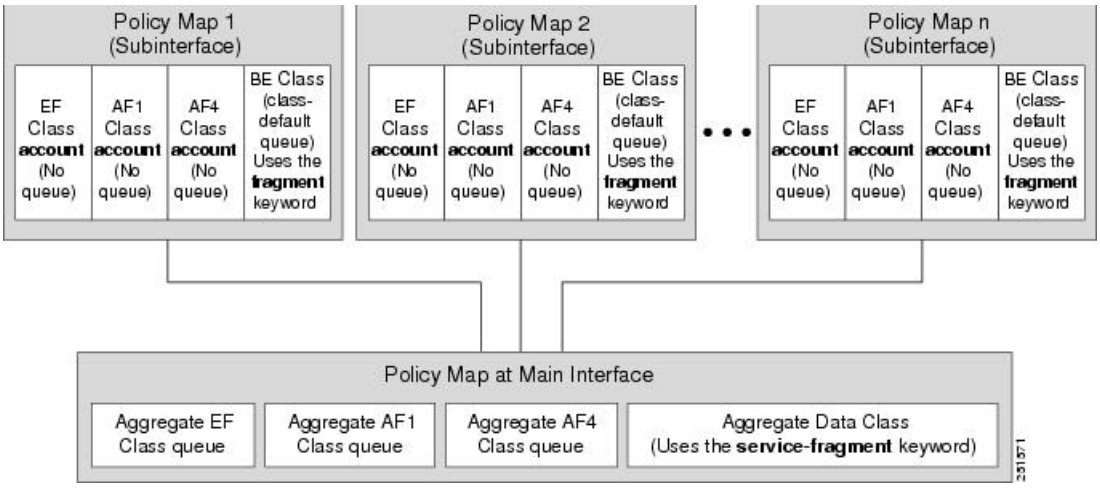
All port-channel subinterfaces configured with fragments that are currently active on a given port-channel member link will use the aggregate service fragment class on that member link. If a member link goes down, the port-channel subinterfaces that must switch to the secondary member link will then use the aggregate service fragment on the new interface.

QoS: Policies Aggregation MQC

The QoS: Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface feature extends the previous support of aggregation of class-default traffic using the **fragment** and **service-fragment** configurations, to other user-defined traffic classes in a subinterface policy-map, such as DSCP-based traffic classes, that are aggregated at the main-interface policy-map as shown in the figure below.

When no queueing is configured on a traffic class in the subinterface policy-map, the **account** command can be used to track queueing drops that occur at the aggregate level for these classes, and can be displayed using the **show policy-map interface** command.

Figure 3: Policy-Map Overview for the MQC Support for Multiple Queue Aggregation at Main Interface Feature



Differences Between the Original Feature and the MQC Support for Multiple Queue AggregationDifferences Between Policy Aggregation—Egress MQC Queuing at Subinterface and the MQC Support for Multiple Queue Aggregation at Main Interface

Although some of the configuration between the “Policy Aggregation – Egress MQC Queuing at Subinterface” scenario and the “MQC Support for Multiple Queue Aggregation at Main Interface - Egress MQC Queuing at Main Interface” scenario appear similar, there are some important differences in the queuing behavior and the internal data handling. See the figure in the “Understanding the QoS: Policies Aggregation MQC” section.

For example, both configurations share and require the use of the **fragment** keyword for the **class class-default** command in the subscriber policy-map, as well as configuration of the **service-fragment** keyword for a user-defined class in the main-interface policy-map to achieve common policy treatment for aggregate traffic. However, the use of this configuration results in different behavior between the original and enhanced QoS policies aggregation implementation:

- In the original implementation using the fragment and service-fragment architecture, all default class traffic and any traffic for classes without defined queueing features at the subinterface goes to the

class-default queue and is aggregated into a common user-defined queue and policy defined at the main policy-map. Subinterface traffic aggregation (for example, from multiple subscribers on the same physical interface) ultimately occurs only for a single class, which is the default class.

- In the enhanced implementation of the MQC Support for Multiple Queue Aggregation at Main Interface feature also using the fragment and service-fragment architecture, all default class traffic also goes to the class-default queue and is aggregated into a common user-defined queue and policy defined at the main policy-map. However, other classes, such as DSCP-based subscriber traffic classes, are also supported for an aggregate policy. These traffic classes do not support any queues or queueing features other than **account** at the subscriber policy-map. The use of the fragment and service-fragment architecture enables these other subscriber traffic classes (from multiple subscribers on the same physical interface) to achieve common policy treatment for aggregate traffic that is defined for those same classes at the main policy-map.

How to Configure QoS for Etherchannels

Configuring Egress MQC Queuing on Port-Channel Subinterface

Before you begin

Traffic classes must be configured using the **class-map** command. A one- or two-level hierarchical policy-map should be configured using previously defined class maps. The port-channel subinterface should have been previously configured with the appropriate encapsulation subcommand to match the select primary and secondary physical interfaces on the Etherchannel. Cisco IOS XE Release 2.1 or later software is required. The global configuration must contain the **port-channel load-balancing vlan-manual** command, or the port-channel main-interface configuration must contain the **load-balancing vlan** command. It is assumed that these commands have already been executed.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *port-channel-number* **.** *subinterface-number*
4. **service-policy output** *policy-map-name*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *port-channel-number . subinterface-number* **Example:** `Device(config)# interface port-channel 1.200`	Specifies the port-channel subinterface that receives the service policy configuration.
Step 4	**service-policy output** *policy-map-name* **Example:** `Device(config-subif)# service-policy output WAN-GEC-sub-Out`	Specifies the name of the service policy that is applied to output traffic.
Step 5	**end** **Example:** `Device(config-subif)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring Egress MQC queuing on Port-Channel Member Links

Before you begin

Traffic classes must be configured using the **class-map** command. A one- or two-level hierarchical policy-map that uses queuing features should be configured using previously defined class maps. The Etherchannel member link interface should already be configured to be part of the channel group (Etherchannel group). No policy-maps that contain queuing commands should be configured on any port-channel subinterfaces. Cisco IOS XE Release 2.1 or later software is required. The global configuration must contain the **port-channel load-balancing vlan-manual** command, or the port-channel main-interface configuration must contain the **load-balancing vlan** command. It is assumed that these commands have already been executed.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface GigabitEthernet** *card/bay/port*
4. **service-policy output** *policy-map-name*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface GigabitEthernet** *card/bay/port* **Example:** `Device(config)# interface GigabitEthernet 0/1/0`	Specifies the member link physical interface that receives the service policy configuration.

Step 4	**service-policy output** *policy-map-name*
	Example:
	`Device(config-if)#`
	`service-policy output WAN-GEC-sub-Out`
		Specifies the name of the service policy that is applied to output traffic for this physical interface that is part of the Etherchannel.

Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring QoS Policies Aggregation—Egress MQC Queuing at Subinterface

Before you begin

Default class traffic from multiple Port-channel subinterfaces can be aggregated into a common policy-map at the main interface when you use the **fragment** keyword at the subinterface **class class-default** configuration, and the **service-fragment** configuration at the main interface class. Queuing occurs at the subinterface for other traffic classes that are defined with queuing features in the subinterface policy-map.

This feature is configured using Modular QoS CLI (MQC). It is most useful in QoS configurations where several policy-maps attached to the same physical interface want aggregated treatment of multiple default traffic classes from multiple port-channel sub-interfaces. Cisco IOS XE Release 2.1 or later software is required. The global configuration must contain the **port-channel load-balancing vlan-manual** command, or the port-channel main-interface must have the **load-balancing vlan** command. It is assumed that these commands have already been executed.

Note This feature is supported when policy-maps are attached to multiple port-channel subinterfaces and the port-channel member link interfaces. This feature cannot be used to collectively classify default traffic classes of policy-maps on different physical interfaces. It can collectively classify all traffic directed toward a given port-channel member link when designated by the **primary** or **secondary** directives on the subinterface **encapsulation** command. All subinterface traffic classes should have queues. However, when a traffic class in the subinterface policy-map is not configured with any queuing feature (commands such as **priority**, **shape**, **bandwidth**, **queue-limit**, **fair-queue**, or **random-detect**), the traffic is assigned to the class-default queue. No classification occurs or is supported at the main interface policy-map for any subinterface traffic classes that do not use the **fragment** and **service-fragment** configuration.

A multistep process is involved with the complete configuration of the QoS Policies Aggregation feature. The following sections detail those steps.

Note the following about attaching and removing a policy-map:

- To configure QoS Policies Aggregation, you must attach the policy-map that contains the **service-fragment** keyword to the main interface first, and then you must attach the policy-map that contains the **fragment** keyword to the subinterface.
- To disable QoS Policies Aggregation, you must remove the policy-map that contains the fragment keyword from the subinterface first, and then you must remove the policy-map that contains the service-fragment keyword from the main interface.

Configuring a Fragment Traffic Class in a Policy-Map

Before you begin

This procedure shows only how to configure the default traffic class as a fragment within a policy-map. It does not include steps on configuring other classes within the policy-map, or other policy-maps on the device.

Example

Note This example shows a sample configuration that is supported in releases prior to Cisco IOS XE Release 2.6.

In the following example, a fragment named BestEffort is created in policy-map subscriber1 and policy-map subscriber 2. In this example, queuing features for other traffic classes are supported at the subinterface policy-map.

```
policy-map subscriber1
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
```

```
  bandwidth remaining ratio 10
policy-map subscriber 2
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
```

Note This example shows a sample configuration that is supported in Cisco IOS XE Release 2.6 and later releases.

The following example also shows how to configure a fragment named BestEffort for the default class in a policy-map on a subinterface using the QoS Policies Aggregation MQC Support for Multiple Queue Aggregation at Main Interface implementation. In this example, notice that queuing features are not supported for the other classes in the policy-map:

```
policy-map subscriber1
 class voice
  set cos 5
  account
 class video
  set cos 4
  account
 class AF1
  account
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
```

After configuring default class statements as fragments in multiple subinterface policy-maps, a separate policy-map with a class statement using the service-fragment keyword must be configured to apply QoS to the class stratements configured as fragments.

What to Do Next

After configuring multiple default class statements as fragments in a policy-map, a separate policy-map with a class statement using the **service-fragment** keyword must be configured to apply QoS to the class statements configured as fragments.

This process is documented in the “Configuring a Service Fragment Traffic Class” section.

Configuring a Service Fragment Traffic Class

Before you begin

This task describes how to configure a service fragment traffic class statement within a policy-map. A service fragment traffic class is used to apply QoS to a collection of default class statements that have been configured previously in other policy-maps as fragments.

This procedure assumes that fragment default traffic classes were already created. The procedure for creating fragment default traffic classes is documented in the "Configuring a Fragment Traffic Class in a Policy-Map" section.

Like any policy-map, the configuration does not manage network traffic until it has been attached to an interface. This procedure does not cover the process of attaching a policy-map to an interface.



Note A service fragment can be used to collectively classify fragments only from the same physical interface. Fragments from different interfaces cannot be classified using the same service fragment.

Only queueing features are allowed in classes where the **service-fragment** keyword is entered, and at least one queueing feature must be entered in classes when the **service-fragment** keyword is used.

A policy-map with a class using the **service-fragment** keyword can be applied only to traffic leaving the interface (policy-maps attached to interfaces using the **service-policy output** command).

A class configured using the **service-fragment** keyword cannot be removed when it is being used to collectively apply QoS to fragments that are still configured on the interface. If you wish to remove a class configured using the **service-fragment** keyword, remove the fragment traffic classes before removing the service fragment.

The **service-fragment** keyword cannot be entered in a child policy-map.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class** *class-name* **service-fragment** *fragment-class-name*
5. **shape average percent** *percent*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:**	Specifies the name of the traffic policy to configure and enters policy-map configuration mode.

	Command or Action	Purpose
	`Device(config)# policy-map BestEffortFragments`	
Step 4	**class** *class-name* **service-fragment** *fragment-class-name* **Example:** `Device(config-pmap)# class data service-fragment BestEffort`	Specifies a class of traffic that is the composite of all fragments matching the *fragment-class-name*. The *fragment-class-name* when defining the fragments in other policy-maps must match the *fragment-class-name* in this command line to properly configure the service fragment class.
Step 5	**shape average percent** *percent* **Example:** `Device(config-pmap-c)# shape average percent 50`	Enters a QoS configuration command. Only queueing features are supported in default traffic classes configured as fragments. The queueing features that are supported are **bandwidth**, **shape**, and **random-detect exponential-weighting-constant**. Multiple QoS queueing commands can be entered.
Step 6	**end** **Example:** `Device(config-pmap-c)# end`	Exits policy-map class configuration mode and returns to privileged EXEC mode.

Examples

Note This example shows a sample configuration that is supported in releases prior to Cisco IOS XE Release 2.6.

In the following example, a policy-map is created to apply QoS to all fragments named BestEffort.

```
policy-map main-interface
 class data service-fragment BestEffort
  shape average 400000000
```

In the following example, two fragments are created and then classified collectively using a service fragment.

```
policy-map subscriber1
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map subscriber 2
```

```
 class voice
  set cos 5
  priority level 1
 class video
  set cos 4
  priority level 2
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
```

Note This example shows a sample configuration that is supported in Cisco IOS XE Release 2.6 and later releases.

The following example shows the creation of two fragments called BestEffort in the subinterface policy-maps, followed by a sample configuration for the **service-fragment** called BestEffort to aggregate the queues at the main interface policy-map:

```
policy-map subscriber1
 class voice
  set cos 5
  account
 class video
  set cos 4
  account
 class AF1
  account
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map subscriber2
 class voice
  set cos 5
  account
 class video
  set cos 4
  account
 class AF1
  account
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
policy-map main-interface
 class voice
  priority level 1
 class video
  priority level 2
 class AF1
  bandwidth remaining ratio 90
 class data service-fragment BestEffort
  shape average 400000000
  bandwidth remaining ratio 1
```

Troubleshooting Tips

Ensure that all class statements that should be part of the same service fragment share the same *fragment-class-name*.

What to Do Next

Attach the service fragment traffic classes to the main physical interfaces.

Attach the fragment traffic classes to the member-link subinterfaces.

Configuring Service Fragments on a Physical Interface Supporting a Gigabit Etherchannel Bundle

Before you begin

This procedure assumes that a service fragment traffic class has already been created. A service fragment traffic class cannot be configured without configuring a fragment class. The procedure for creating a fragment class is documented in the "Configuring a Fragment Traffic Class in a Policy-Map" section. The procedure for creating a service fragment traffic classes is documented in the "Configuring a Service Fragment Traffic Class" section.

These instructions do not provide any details about the options that can be configured for Gigabit Etherchannel member link subinterfaces. These instructions document only the procedure for attaching a policy-map that already has a fragment traffic class to a member link subinterface.



Note For proper behavior, when a port-channel member link goes down, all member links should have the same policy-map applied.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface GigabitEthernet** *card/bay/port*
4. **service-policy output** *service-fragment-class-name*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface GigabitEthernet** *card/bay/port* **Example:** `Device(config)# interface GigabitEthernet 0/1/0`	Specifies the member link physical interface that receives the service-policy configuration.
Step 4	**service-policy output** *service-fragment-class-name* **Example:** `Device(config-if)# service-policy output aggregate-member-link`	Attaches a service policy that contains a service fragment default traffic class to the physical Gigabit Ethernet interface.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Examples

In the following example, the policy-map aggregate-member-link is attached to the physical interface.

```
interface GigabitEthernet1/1/1
 service-policy output aggregate-member-link
!
interface GigabitEthernet1/1/2
 service-policy output aggregate-member-link
```

What to do next

Ensure that the fragment class name is consistent across service-fragment and fragment class definitions. Continue to the “Configuring Fragments on Gigabit Etherchannel Member Link Subinterfaces” section.

Configuring Fragments on Gigabit Etherchannel Member Link Subinterfaces

Before you begin

This procedure assumes that a service fragment traffic class has already been created. A service fragment traffic class cannot be configured without configuring a fragment class. The procedure for creating a fragment class is documented in the “Configuring a Fragment Traffic Class in a Policy-Map” section. The procedure for creating a service fragment traffic class is documented in the “Configuring a Service Fragment Traffic Class” section.

These instructions do not provide any details about the options that can be configured for Gigabit Etherchannel member link subinterfaces. These instructions only document the procedure for attaching a policy-map that already has a fragment traffic class to a member link subinterface.

Fragments cannot be used for traffic on two or more physical interfaces.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *port-channel-interface-number* **.** *port-channel-subinterface-number*
4. **service-policy output** *fragment-class-name*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *port-channel-interface-number* **.** *port-channel-subinterface-number* **Example:** `Device(config)# interface port-channel 1.100`	Enters subinterface configuration mode to configure an Etherchannel member link subinterface.
Step 4	**service-policy output** *fragment-class-name* **Example:** `Device(config-subif)# service-policy output subscriber`	Attaches a service policy that contains a fragment default traffic class to the Etherchannel member link subinterface
Step 5	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Example

In the following example, the service policy named subscriber has a fragment default traffic class and is attached to the port-channel subinterface of an Etherchannel bundle.

```
interface port-channel 1.100
 service-policy output subscriber
```

Configuring Ingress Policing and Marking on Port-Channel Subinterface

Before you begin

Traffic classes must be configured using the **class-map** command. A one- or two-level hierarchical policy-map should be configured using previously defined class maps. The Etherchannel member link interface should already be configured to be part of the channel group (Etherchannel group). Cisco IOS XE Release 2.1 or later software is required. The global configuration must contain the **port-channel load-balancing vlan-manual** command or the port-channel main-interface configuration must contain the **load-balancing vlan** command. It is assumed that these commands have already been executed.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *port-channel-number* **.** *port-channel-interface-number* **.** *sub-interface-number*
4. **service-policy input** *policy-map-name*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *port-channel-number . port-channel-interface-number . sub-interface-number* **Example:** `Device(config)# interface port-channel 1.100.100`	Enters subinterface configuration mode to configure an Etherchannel member link subinterface.
Step 4	**service-policy input** *policy-map-name* **Example:** `Device(config-subif)# service-policy input sub-intf-input`	Specifies the name of the service policy that is applied to input traffic for the port-channel subinterface previously specified.
Step 5	**end** **Example:**	Exits subinterface configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-subif)# end`	

Example

In the following example, the service policy named sub-intf-input is defined and attached to the port-channel subinterface in the input direction.

```
policy-map sub-intf-input
  class voice
    set precedence 5
  class video
    set precedence 6
  class class-default
    set precedence 3
!
interface Port-channel 1.100
 service-policy input sub-intf-input
```

Configuring Egress Policing and Marking on Port-Channel Member Links

Before you begin

Traffic classes must be configured using the **class-map** command. A one- or two-level hierarchical policy-map should be configured using previously defined class maps. The Etherchannel member link interface should already be configured to be part of the channel group (Etherchannel group). Cisco IOS XE Release 2.1 or later software is required. The global configuration must contain the **port-channel load-balancing vlan-manual** command or the port-channel main-interface configuration must contain the **load-balancing vlan** command. It is assumed that these commands have already been executed.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *port-channel-number* **.** *port-channel-interface-number* **.** *sub-interface-number*
4. **service-policy output** *policy-map-name*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *port-channel-number.port-channel-interface-number.sub-interface-number* **Example:** `Device(config)# interface port-channel 1.100.100`	Enters subinterface configuration mode to configure an Etherchannel member link subinterface.

Step 4	**service-policy output** *policy-map-name*
	Example:
	`Device(config-subif)#`
	`service-policy output WAN-GEC-member-Out-police`
		Specifies the name of the service policy that is applied to output traffic for the Etherchannel member link subinterface specified in the previous step.

Step 5	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Example

In the following example, the service policy named WAN-GEC-member-Out-police is defined and attached to the port-channel subinterface in the output direction.

```
policy-map WAN-GEC-member-Out-police
  class voice
    set precedence 5
  class video
    set precedence 6
  class class-default
    set precedence 3
!
interface port-channel 1.100
 service-policy output WAN-GEC-member-Out-police
```

Configuring Policies Aggregation—MQC Support for Multiple Queue Aggregation at Main Interface

Before you begin

This feature is configured using the MQC. It is most useful in QoS configurations where several policy-maps attached to the same physical interface want aggregated treatment of multiple user-defined traffic classes from multiple port-channel subinterfaces. Cisco IOS XE Release 2.6 or later software is required. The global configuration must contain the following command: **port-channel load-balancing vlan-manual** or the main interface of the port-channel being configured must have the following command: **port-channel load-balancing vlan**. It is assumed that these commands have already been executed.

This feature is supported when policy-maps are attached to multiple port-channel subinterfaces and the port-channel member link interfaces. This feature cannot be used to collectively classify default traffic classes of policy-maps on different physical interfaces. It can collectively classify all traffic directed towards a given Port-channel member-link when designated by the **primary** or **secondary** directives on the sub-interface **encapsulation** command. The following items describe the behavior and restrictions on configuring this type of QoS Policy Aggregation with Etherchannel:

- Subinterface traffic classes without configured queuing features do not have queues at the subscriber level
- Default class traffic from multiple subinterfaces can be aggregated into a common policy-map at the main interface when you use the **fragment** keyword at the subinterface **class class-default** configuration, and **service-fragment** configuration at the main interface class
- This configuration additionally enables support for other subinterface traffic classes (such as DSCP-based classes) to be aggregated into a common policy-map at the main interface.
- This feature is enabled by using the **fragment** keyword in the subinterface **class-default** class, and **service-fragment** configuration in the main interface class (this also enables aggregation of the default class.
- Queuing features are not configured at the subinterface policy-map for the other traffic classes.
- Queuing occurs at the main interface policy-map for other subinterface traffic classes as an aggregate.
- Optional tracking of statistics is supported using the **account** command for other traffic classes in the subinterface policy-map.

A multistep process is involved with the complete configuration of QoS multiple queue aggregation at a main interface feature, as follows:

1. Configure default class statements as fragments in multiple subinterface policy-maps as described in the “Configuring a Fragment Traffic Class in a Policy-Map” section.
2. Configure a separate policy-map with a class statement using the **service-fragment** keyword in order to apply QoS to the class statements configured as fragments as described in the “Configuring a Service Fragment Traffic Class” section.
3. Configure service fragment traffic classes and attach them to the main physical interfaces as described in the “Configuring Service Fragments on a Physical Interface Supporting a Gigabit Etherchannel Bundle” section.

4. Configure fragment traffic classes and attach them to the member link subinterfaces as described in the "Configuring Fragments on Gigabit Etherchannel Member Link Subinterfaces" section.

Configuring MQC Queuing on Port-Channel Member Link—No Etherchannel Load Balancing

Before you begin

Traffic classes must be configured using the **class-map** command. A one or two level hierarchical policy-map should be configured using previously defined class maps.

Cisco IOS XE Release 2.4 or later software is required.

The port-channel main interface should also contain the following commands that create an active/standby scenario. Such a configuration will allow only a single interface to be active and forwarding traffic at any time.

- **interface Port-channel1**
- **lacp fast-switchover**
- **lacp max-bundle 1**

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface GigabitEthernet** *card/bay/port*
4. **service-policy output** *policy-map-name*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface GigabitEthernet** *card/bay/port* **Example:**	Specifies the member link physical interface that receives the service policy configuration.

	Command or Action	Purpose
	`Device(config)# interface GigabitEthernet 0/1/0`	
Step 4	**service-policy output** *policy-map-name* **Example:** `Device(config-if)# service-policy output WAN-GEC-member-Out`	Specifies the name of the service policy that is applied to output traffic.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Example

In the following example, the service policy named main-intf is defined and attached to the port-channel member links in the output direction.

```
interface Port-channel 1
 lcap fast-switchover
 lacp max-bundle 1
!
policy-map main-intf
 class voice
  priority
  police cir 10000000
 class video
  bandwidth remaining ratio 10
 class class-default
  bandwidth remaining ratio 3
!
interface GigabitEthernet0/0/0
 channel-group 1 mode active
 service-policy output main-intf
!
interface GigabitEthernet0/0/1
 channel-group 1 mode active
 service-policy output main-intf
```

Configuring MQC Queuing Configuration on Port-Channel Member Link—Etherchannel Load Balancing

Before you begin

Traffic classes must be configured using the **class-map** command. A one- or two-level hierarchical policy-map should be configured using previously defined class maps. The port-channel subinterface should have been previously configured with the appropriate encapsulation subcommand to match the select primary and secondary physical interfaces on the Etherchannel. Cisco IOS XE Release 2.5 or later software is required.

The Etherchannel setup may have multiple active interfaces with flow-based load balancing enabled.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface GigabitEthernet** *card/bay/port*
4. **service-policy output** *policy-map-name*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface GigabitEthernet** *card/bay/port* **Example:** `Device(config)# interface GigabitEthernet 0/1/0`	Specifies the member link physical interface that receives the service policy configuration.

Step 4	**service-policy output** *policy-map-name*
	Example:
	`Device(config-if)#`
	`service-policy output WAN-GEC-member-Out`
		Specifies the name of the service policy that is applied to output traffic.

Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Example

In the following example, the service policy named main-intf is defined and attached to the port-channel member links in the output direction.

```
class voice
 priority
 police cir 10000000
```

```
  class video
   bandwidth remaining ratio 10
  class class-default
   bandwidth remaining ratio 3
!
interface GigabitEthernet0/0/0
 channel-group 1 mode active
 service-policy output main-intf
!
interface GigabitEthernet0/0/1
 channel-group 1 mode active
 service-policy output main-intf
```

Configuration Examples for QoS for Etherchannels

Example: Configuring QoS Policies Aggregation—Egress MQC Queuing at Subinterface

```
port-channel load-balancing vlan-manual
!
class-map match-all BestEffort
!
class-map match-all video
 match precedence 4
!
class-map match-all voice
 match precedence 5
!
policy-map subscriber
 class voice
  priority level 1
 class video
  priority level 2
 class class-default fragment BE
  shape average 100000000
  bandwidth remaining ratios 80

policy-map aggregate-member-link
 class BestEffort service-fragment BE
 shape average 100000000
!
interface Port-channel1
 ip address 209.165.200.225 255.255.0.0
!
interface Port-channel1.100
 encapsulation dot1Q 100
 ip address 209.165.200.226 255.255.255.0
 service-policy output subscriber
!
interface Port-channel1.200
 encapsulation dot1Q 200
 ip address 209.165.200.227 255.255.255.0
 service-policy output subscriber
!
interface Port-channel1.300
 encapsulation dot1Q 300
 ip address 209.165.200.228 255.255.255.0
```

```
 service-policy output subscriber
!
interface GigabitEthernet1/1/1
 no ip address
 channel-group 1 mode on
 service-policy output aggregate-member-link
!
interface GigabitEthernet1/1/2
 no ip address
 channel-group 1 mode on
 service-policy output aggregate-member-link
```

Example: Configuring QoS Policies Aggregation—MQC Support for Multiple Queue Aggregation at Main Interface

```
port-channel load-balancing vlan-manual
!
policy-map subscriber1
 class voice
  set cos 5
  account
 class video
  set cos 4
  account
 class AF1
  account
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
!
policy-map subscriber2
 class voice
  set cos 2
  account
 class video
  set cos 3
  account
 class AF1
  account
 class class-default fragment BestEffort
  shape average 200000000
  bandwidth remaining ratio 10
!
policy-map main-interface-out
 class voice
  priority level 1
 class video
  priority level 2
 class AF1
  bandwidth remaining ratio 90
 class data service-fragment BestEffort
  shape average 400000000
  bandwidth remaining ratio 1
!
interface GigabitEthernet1/1/1
 no ip address
 channel-group 1 mode on
 service-policy output main-interface-out
!
interface GigabitEthernet1/1/2
```

```
 no ip address
 channel-group 1 mode on
 service-policy output main-interface-out
!
interface Port-channel1.100
 encapsulation dot1Q 100
 ip address 10.0.0.1 255.255.255.0
 service-policy output subscriber1
!
interface Port-channel1.200
 encapsulation dot1Q 200
 ip address 10.0.0.2 255.255.255.0
 service-policy output subscriber2
!
interface Port-channel1.300
 encapsulation dot1Q 300
 ip address 10.0.0.4 255.255.255.0
 service-policy output subscriber2
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Command List, All Releases*
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Quality of Service Solutions Command Reference*
Modular Quality of Service Command-Line Interface	"Applying QoS Features Using the MQC" module
Configuring RADIUS-based policing	*Intelligent Services Gateway Configuration Guide*
CISCO ASR 1000 Series software configuration	*Cisco ASR 1000 Series Aggregation Services Routers Software Configuration Guide*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Quality of Service for Etherchannel Interfaces

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Table 1: Feature Information for Quality of Service for Etherchannel Interfaces

Feature Name	Releases	Feature Information
Egress MQC Queuing Configuration on Port-Channel Subinterface	Cisco IOS XE Release 2.1	This feature supports the configuration of Egress MQC queuing on port-channel subinterface. This feature was introduced on Cisco ASR 1000 Series Routers.
Egress MQC Queuing Configuration on Port-Channel Member Link	Cisco IOS XE Release 2.1	This feature supports the configuration of Egress MQC queuing on port-channel member link. This feature was introduced on Cisco ASR 1000 Series Routers.
QoS Policies Aggregation—Egress MQC Queuing at Subinterface	Cisco IOS XE Release 2.1	This feature supports the configuration of QoS Policies Aggregation - Egress MQC queuing at subinterface. This feature was introduced on Cisco ASR 1000 Series Routers.
Ingress Policing and Marking on Port-Channel Subinterface	Cisco IOS XE Release 2.1	This feature supports the configuration of Ingress Policing and Marking on port-channel subinterface. This feature was introduced on Cisco ASR 1000 Series Routers.
Egress Policing and Marking on Port-Channel Member Link	Cisco IOS XE Release 2.1	This feature supports the configuration of Egress policing and marking on port-channel member link. This feature was introduced on Cisco ASR 1000 Series Routers.
Egress MQC Queuing Configuration on Port-Channel Member Link - No Etherchannel Load Balancing	Cisco IOS XE Release 2.4	This feature supports the configuration of Egress MQC Queuing on Port-Channel Member Link - no Etherchannel Load Balancing. This feature was introduced on Cisco ASR 1000 Series Routers.

Feature Name	Releases	Feature Information
Egress MQC Queuing Configuration Supported on Port-Channel Member Link - Etherchannel Load Balancing	Cisco IOS XE Release 2.5	This feature supports the configuration of Egress MQC Queuing on Port-Channel Member Link - Etherchannel Load Balancing. This feature was introduced on Cisco ASR 1000 Series Routers.
QoS Policies Aggregation - MQC Support for Multiple Queue Aggregation at Main Interface - Egress MQC Queuing at Main Interface	Cisco IOS XE Release 2.6	This feature supports the configuration of QoS Policies Aggregation - MQC Support for Multiple Queue Aggregation at Main Interface - Egress MQC Queuing at Main Interface. This feature was introduced on Cisco ASR 1000 Series Routers.