Configuring Application Visibility and Control

Application Visibility and Control (AVC) is a solution for Cisco network devices that provides application-level classification, monitoring, and traffic control to improve business-critical application performance, facilitate capacity management and planning, and reduce network operating costs. The Cisco AVC solution is provided within the Branch and Aggregation routers, Cisco Switches, and Cisco Wireless Controllers and Access points.

For information about AVC on Cisco Switches, see *Configuring Application Visibility and Control in a Wired Network*.

For information about AVC on Cisco Wireless Controllers and Access points, see *Configuring Application Visibility and Control in a Wireless Network*.

Finding Feature Information

Your software release may not support all of the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release.

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

Information About Application Visibility and Control in a Wired Network

Application Visibility and Control (AVC) is a critical part of Cisco's efforts to evolve its Branch and Campus solutions from being strictly packet and connection based to being application-aware and application-intelligent. Application Visibility and Control (AVC) classifies applications using deep packet inspection techniques with the Network-Based Application Recognition (NBAR2) engine. AVC can be configured on wired access ports for standalone switches as well as for a switch stack. NBAR2 can be activated either explicitly on the interface by enabling protocol-discovery or implicitly by attaching a QoS policy that contains **match protocol** classifier. Wired AVC Flexible NetFlow (FNF) can be configured on an interface to provide client, server and application statistics per interface. The record is similar to **application-client-server-stats** traffic monitor which is available in **application-statistics** and **application-performance** profiles in Easy Performance Monitor (Easy perf-mon or ezPM).

Supported AVC Class Map and Policy Map Formats

Supported AVC Class Map Format

Class Map Format	Class Map Example	Direction
match protocol *protocol name*	`class-map match-any NBAR-VOICE` `match protocol ms-lync-audio`	Both ingress and egress

Combination filters	`class-map match-any NBAR-VOICE`
	`match protocol ms-lync-audio`
	`match dscp ef`
		Both ingress and egress

Supported AVC Policy Format

Policy Format	QoS Action
Egress policy based on match protocol filter	Mark and police
Ingress policy based on match protocol filter	Mark and police

The following table describes the detailed AVC policy format with an example:

AVC Policy Format	AVC Policy Example	Direction

Basic set	`policy-map MARKING-IN`
	`class NBAR-MM_CONFERENCING`
	`set dscp af41`
		Ingress and egress

Basic police	`policy-map POLICING-IN`
	`class NBAR-MM_CONFERENCING`
	`police cir 600000`
	`set dscp af41`
		Ingress and egress

Basic set and police	`policy-map webex-policy`
	` class webex-class`
	` set dscp ef`
	` cos police 5000000`
		Ingress and egress

Multiple set and police including default	`policy-map webex-policy`
	` class webex-class`
	` set dscp af31`
	` cos police 4000000`
	` class class-webex-category`
	` set dscp ef`
	` cos police 6000000`
	` class class-default`
	` set dscp <>`
		Ingress and egress

Hierarchical police	`policy-map webex-policy`
	` class webex-class`
	` police 5000000`
	` service-policy client-in-police-only`
	
	` policy-map client-in-police-only`
	` class webex-class`
	` police 100000`
	` class class-webex-category`
	` set dscp ef`
	` cos police 200000`
		Ingress and egress

Hierarchical set and police	`policy-map webex-policy`
	` class class-default`
	` police 1500000`
	` service policy client-up-child`
	` policy-map webex-policy`
	` class webex-class`
	` police 100000`
	` set dscp ef`
	` class class-webex-category`
	` police 200000`
	` set dscp af31`	

Restrictions for Wired Application Visibility and Control

- NBAR based QoS policy configuration is allowed only on wired physical ports. Policy configuration is not supported on virtual interfaces, for example, VLAN, Port-Channel and other logical interfaces.
- NBAR2 based match criteria **match protocol** will be allowed only with marking or policing actions. NBAR2 match criteria will not be allowed in a policy that has queuing features configured.
- 'Match Protocol': up to 255 concurrent different protocols in all policies (8 bits HW limitation).
- NBAR2 attributes based QOS is not supported (**match protocol** attribute).
- AVC is not supported on management port (Gig 0/0).
- IPv6 packet classification is not supported.
- Only IPv4 unicast(TCP/UDP) is supported.
- Web UI: You can configure application visibility and perform application monitoring from the Web UI. Application Control can only be done using the CLI. It is not supported on the Web UI.
- NBAR and ACL logging cannot be configured together on the same switch.
- Protocol-discovery, application-based QoS, and wired AVC FNF cannot be configured together at the same time on the same interface with the non-application-based FNF. However, these wired AVC features can be configured with each other. For example, protocol-discovery, application-based QoS and wired AVC FNF can be configured together on the same interface at the same time.
- In Cisco IOS XE Denali 16.3.2, **show flow monitor** *flow-monitor-name* **statistics** and **show flow monitor** *flow-monitor-name* **cache** commands are not supported for wired AVC. These commands do not display any information specific to wired AVC.
- A single predefined record is supported with wired AVC FNF.
- Attachment should be done only on physical Layer2 (Access/Trunk) and Layer3 ports. Uplink can be attached as long as it is a single uplink and is not part of a port channel.
- Performance: Each switch member is able to handle 500 connections per second (CPS) at less than 50% CPU utilization.
- Scale: Able to handle up to 10,000 bi-directional flows per 48 access ports and 5000 bi-directional flows per 24 access ports. (~200 flows per access port).

How to Configure Application Visibility and Control

Configuring Application Visibility and Control in a Wired Network

To configure application visibility and control on wired ports, follow these steps:

Configuring Visibility :

- Activate NBAR2 engine by enabling protocol-discovery on the interface using the **ip nbar protocol-discovery** command in the interface configuration mode. See Enabling Application Recognition on an interface, on page 5 .

Configuring Control : Configure QoS policies based on application by

1 Creating an AVC QoS policy. See Creating AVC QoS Policy, on page 6 .

2 Applying AVC QoS policy to the interface. See Applying a QoS Policy to the switch port, on page 8 .

Configuring application-based Flexible Netflow :

- Create a flow record by specifying key and non-key fields to the flow. See Creating a Flow Record, on page 9 .
- Create a flow exporter to export the flow record. See Creating a Flow Exporter, on page 13 .
- Create a flow monitor based on the flow record and the flow exporter. See Creating a Flow Monitor, on page 14 .
- Attach the flow monitor to the interface. See Associating Flow Monitor to an interface, on page 16 .

Protocol-Discovery, application-based QoS and application-based FNF are all independent features. They can be configured independently or together on the same interface at the same time.

Enabling Application Recognition on an interface

To enable application recognition on an interface, follow these steps:

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*
3. **ip nbar protocol-discovery**
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Specifies the interface for which you are enabling protocol-discovery and enters interface configuration mode.

	Command or Action	Purpose
Step 3	**ip nbar protocol-discovery** **Example:** `Device(config-if)# ip nbar protocol-discovery`	Enables application recognition on the interface by activating NBAR2 engine.
Step 4	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Creating AVC QoS Policy

To create AVC QoS policy, perform these general steps:

1. Create a class map with match protocol filters.
2. Create a policy map.
3. Apply the policy map to the interface.

Creating a Class Map

You need to create a class map before configuring any match protocol filter. The QoS actions such as marking and policing can be applied to the traffic. The AVC match protocol filters are applied to the wired access ports. For more information about the protocols that are supported, see http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_nbar/prot_lib/config_library/nbar-prot-pack-library.html.

SUMMARY STEPS

1. **configure terminal**
2. **class-map** *class-map-name*
3. **match protocol** *application-name*
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**class-map** *class-map-name* **Example:** `Device(config)# class-map webex-class`	Creates a class map.

Step 3	**match protocol** *application-name*
	
	Example:
	
	`Device(config)# class-map webex-class`
	`Device(config-cmap)# match protocol webex-media`
		Specifies match to the application name.

Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Creating a Policy Map

SUMMARY STEPS

1. **configure terminal**
2. **policy-map** *policy-map-name*
3. **class** [*class-map-name* | **class-default**]
4. **police** *rate-bps burst-byte*
5. **set** {**dscp** *new-dscp* | **cos** *cos-value*}
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**policy-map** *policy-map-name* **Example:** `Device(config)# policy-map webex-policy`	Creates a policy map by entering the policy map name, and enters policy-map configuration mode. By default, no policy maps are defined. The default behavior of a policy map is to set the DSCP to 0 if the packet is an IP packet and to set the CoS to 0 if the packet is tagged. No policing is performed. **Note** To delete an existing policy map, use the **no policy-map** *policy-map-name* global configuration command.

	Command or Action	Purpose
Step 3	**class** [*class-map-name* | **class-default**] **Example:** `Device(config-pmap)# class webex-class`	Defines a traffic classification, and enters policy-map class configuration mode. By default, no policy map and class maps are defined. If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command. A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** is included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**. **Note** To delete an existing class map, use the **no class** *class-map-name* policy-map configuration command.
Step 4	**police** *rate-bps burst-byte* **Example:** `Device(config-pmap-c)# police 100000 80000`	Defines a policer for the classified traffic. By default, no policer is defined. • For *rate-bps,* specify an average traffic rate in bits per second (b/s). The range is 8000 to 10000000000. • For *burst-byte,* specify the normal burst size in bytes. The range is 8000 to 1000000.
Step 5	**set** {**dscp** *new-dscp* | **cos** *cos-value*} **Example:** `Device(config-pmap-c)# set dscp 45`	Classifies IP traffic by setting a new value in the packet. • For **dscp** *new-dscp*, enter a new DSCP value to be assigned to the classified traffic. The range is 0 to 63.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Applying a QoS Policy to the switch port

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*
3. **service-policy input** *policymapname*
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Device(config)# interface Gigabitethernet 1/0/1`	Enters the interface configuration mode.
Step 3	**service-policy input** *policymapname* **Example:** `Device(config-if)# service-policy input MARKING_IN`	Applies local policy to interface.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Configuring Wired AVC Flexible Netflow

Creating a Flow Record

A single flow record can be configured and associated with a flow monitor.

SUMMARY STEPS

1. **configure terminal**
2. **flow record** *flow_record_name*
3. **description** *description*
4. **match ipv4 version**
5. **match ipv4 protocol**
6. **match application name**
7. **match connection client ipv4 address**
8. **match connection server ipv4 address**
9. **match connection server transport port**
10. **match flow observation point**
11. **collect flow direction**
12. **collect connection initiator**
13. **collect connection client counter packets long**
14. **collect connection client counter bytes network long**
15. **collect connection server counter packets long**
16. **collect connection server counter bytes network long**
17. **collect timestamp absolute first**
18. **collect timestamp absolute last**
19. **collect connection new-connections**
20. **end**
21. **show flow record**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**flow record** *flow_record_name* **Example:** `Device(config)# flow record flow-record-1`	Enters flow record configuration mode.
Step 3	**description** *description* **Example:** `Device(config-flow-record)# description flow-record-1`	(Optional) Creates a description for the flow record.

	Command or Action	Purpose
Step 4	**match ipv4 version** **Example:** `Device (config-flow-record)# match ipv4 version`	Specifies a match to the IP version from the IPv4 header.
Step 5	**match ipv4 protocol** **Example:** `Device (config-flow-record)# match ipv4 protocol`	Specifies a match to the IPv4 protocol.
Step 6	**match application name** **Example:** `Device (config-flow-record)# match application name`	Specifies a match to the application name. **Note** This action is mandatory for AVC support, as this allows the flow to be matched against the application.
Step 7	**match connection client ipv4 address** **Example:** `Device (config-flow-record)# match connection client ipv4 address`	Specifies a match to the IPv4 address of the client (flow initiator).
Step 8	**match connection server ipv4 address** **Example:** `Device (config-flow-record)# match connection server ipv4 address`	Specifies a match to the IPv4 address of the server (flow responder).
Step 9	**match connection server transport port** **Example:** `Device (config-flow-record)# match connection server transport port`	Specifies a match to the transport port of the server.
Step 10	**match flow observation point** **Example:** `Device (config-flow-record)# match flow observation point`	Specifies a match to the observation point ID for flow observation metrics.
Step 11	**collect flow direction** **Example:** `Device (config-flow-record)# collect flow direction`	Specifies to collect the direction — Ingress or Egress — of the relevant side — Initiator or Responder — of the bi-directional flow that is specified by the **initiator** keyword in the **collect connection initiator** command in the step below. Depending on the value specified by the **initiator** keyword, the **flow direction** keyword takes the following values : • 0x01 = Ingress Flow • 0x02 = Egress Flow When the **initiator** keyword is set to initiator, the flow direction is specified from the initiator side of the flow. When

	Command or Action	Purpose
		the initiator keyword is set to responder, the flow direction is specified from the responder side of the flow. For wired AVC, the **initiator** keyword is always set to initiator.
Step 12	**collect connection initiator** **Example:** `Device (config-flow-record)# collect connection initiator`	Specifies to collect the side of the flow — Initiator or Responder — relevant to the direction of the flow specified by the **collect flow direction** command. The **initiator** keyword provides the following information about the direction of the flow : • 0x01 = Initiator - the flow source is the initiator of the connection For wired AVC, the **initiator** keyword is always set to initiator.
Step 13	**collect connection client counter packets long** **Example:** `Device (config-flow-record)# collect connection client counter packets long`	Specifies to collect the number of packets sent by the client.
Step 14	**collect connection client counter bytes network long** **Example:** `Device (config-flow-record)# collect connection client counter bytes network long`	Specifies to collect the total number of bytes transmitted by the client.
Step 15	**collect connection server counter packets long** **Example:** `Device (config-flow-record)# collect connection server counter packets long`	Specifies to collect the number of packets sent by the server.
Step 16	**collect connection server counter bytes network long** **Example:** `Device (config-flow-record)# collect connection server counter bytes network long`	Specifies to collect the total number of bytes transmitted by the server.
Step 17	**collect timestamp absolute first** **Example:** `Device (config-flow-record)# collect timestamp absolute first`	Specifies to collect the time, in milliseconds, when the first packet was seen in the flow.
Step 18	**collect timestamp absolute last** **Example:** `Device (config-flow-record)# collect timestamp absolute last`	Specifies to collect the time, in milliseconds, when the most recent packet was seen in the flow.

	Command or Action	Purpose
Step 19	**collect connection new-connections** **Example:** `Device (config-flow-record)# collect connection new-connections`	Specifies to collect the number of connection initiations observed.
Step 20	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.
Step 21	**show flow record** **Example:** `Device # show flow record`	Displays information about all the flow records.

Creating a Flow Exporter

You can create a flow exporter to define the export parameters for a flow.

SUMMARY STEPS

1. **configure terminal**
2. **flow exporter** *flow_exporter_name*
3. **description** *description*
4. **destination** { *hostname* | *ipv4-address* | *ipv6-address* }
5. **option application-table** [**timeout** *seconds*]
6. **end**
7. **show flow exporter**
8. **show flow exporter statistics**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**flow exporter** *flow_exporter_name* **Example:** `Device(config)# flow exporter flow-exporter-1`	Enters flow exporter configuration mode.

	Command or Action	Purpose
Step 3	**description** *description* **Example:** `Device(config-flow-exporter)# description flow-exporter-1`	(Optional) Creates a description for the flow exporter.
Step 4	**destination** { *hostname* | *ipv4-address* | *ipv6-address* } **Example:** `Device (config-flow-exporter)# destination 10.10.1.1`	Specifies the hostname, IPv4 or IPv6 address of the system to which the exporter sends data.
Step 5	**option application-table** [**timeout** *seconds*] **Example:** `Device (config-flow-exporter)# option application-table timeout 500`	(Optional) Configures the application table option for the flow exporter. The **timeout** option configures the resend time in seconds for the flow exporter. The valid range is from 1 to 86400 seconds.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.
Step 7	**show flow exporter** **Example:** `Device # show flow exporter`	Displays information about all the flow exporters.
Step 8	**show flow exporter statistics** **Example:** `Device # show flow exporter statistics`	Displays flow exporter statistics.

Creating a Flow Monitor

You can create a flow monitor and associate it with a flow record.

SUMMARY STEPS

1. **configure terminal**
2. **flow monitor** *monitor-name*
3. **description** *description*
4. **record** *record-name*
5. **exporter** *exporter-name*
6. **cache type normal** { **timeout** {**active** | **inactive**} | **type normal** }
7. **end**
8. **show flow monitor**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**flow monitor** *monitor-name* **Example:** `Device (config)# flow monitor flow-monitor-1`	Creates a flow monitor and enters flow monitor configuration mode.
Step 3	**description** *description* **Example:** `Device (config-flow-monitor)# description flow-monitor-1`	(Optional) Creates a description for the flow monitor.
Step 4	**record** *record-name* **Example:** `Device (config-flow-monitor)# record flow-record-1`	Specifies the name of a record that was created previously.
Step 5	**exporter** *exporter-name* **Example:** `Device (config-flow-monitor)# exporter flow-exporter-1`	Specifies the name of an exporter that was created previously.

Step 6	**cache type normal** { **timeout** {**active** | **inactive**} | **type normal** }
	
	Example:
	`Device (config-flow-monitor)# cache timeout active 1800`
	
	Example:
	`Device (config-flow-monitor)# cache timeout inactive 200`
	
	Example:
	`Device (config-flow-monitor)# cache type normal`
		(Optional) Specifies to configure flow cache parameters.
		Note Only normal cache type is supported. Cache size configuration is not supported. The cache has a constant predefined size of 10,000.

Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.
Step 8	**show flow monitor** **Example:** `Device # show flow monitor`	Displays information about all the flow monitors.

	Command or Action	Purpose
		Note **show flow monitor** *flow-monitor-name* **statistics** and **show flow monitor** *flow-monitor-name* **cache** commands are not supported for wired AVC. These commands do not display any information specific to wired AVC. **show flow exporter statistics** command can be used as a limited alternative to **show flow monitor** *flow-monitor-name* **cache** command for displaying flow monitor cache statistics.

Associating Flow Monitor to an interface

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*
3. **ip flow monitor** *monitor-name* { **input** | **output** }
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Device(config)# interface Gigabitethernet 1/0/1`	Enters the interface configuration mode.
Step 3	**ip flow monitor** *monitor-name* { **input**	**output** } **Example:** `Device (config-if) # ip flow monitor flow-monitor-1 input`	Associates a flow monitor to the interface for input and/or output packets.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

NBAR2 Custom Applications

NBAR2 supports the use of custom protocols to identify custom applications. Custom protocols support protocols and applications that NBAR2 does not currently support.

In every deployment, there are local and specific applications which are not covered by the NBAR2 protocol pack provided by Cisco. Local applications are mainly categorized as:

- Specific applications to an organization
- Applications specific to a geography

NBAR2 provides a way to manually customize such local applications. You can manually customize applications using the command **ip nbar custom** *myappname* in global configuration mode. Custom applications take precedence over built-in protocols. For each custom protocol, user can define a selector ID that can be used for reporting purposes.

There are various types of application customization:

Generic protocol customization

- HTTP
- SSL
- DNS

Composite : Customization based on multiple underlying protocols **– server-name**

Layer3/Layer4 customization

- IPv4 address
- DSCP values
- TCP/UDP ports
- Flow source or destination direction

Byte Offset : Customization based on specific byte values in the payload

HTTP Customization

HTTP customization could be based on a combination of HTTP fields from:

- **cookie** - HTTP Cookie
- **host** - Host name of Origin Server containing resource
- **method** - HTTP method
- **referrer** - Address the resource request was obtained from
- **url** - Uniform Resource Locator path
- **user-agent** - Software used by agent sending the request
- **version** - HTTP version

- **via** - HTTP via field

HTTP Customization

Custom application called MYHTTP using the HTTP host “*mydomain.com” with Selector ID 10.

```
Device# configure terminal
Device(config)# ip nbar custom MYHTTP http host *mydomain.com id 10
```

SSL Customization

Customization can be done for SSL encrypted traffic using information extracted from the SSL Server Name Indication (SNI) or Common Name (CN).

SSL Customization

Custom application called MYSSL using SSL unique-name “mydomain.com” with selector ID 11.

```
Device# configure terminal
Device(config)#ip nbar custom MYSSL ssl unique-name *mydomain.com id 11
```

DNS Customization

NBAR2 examines DNS request and response traffic, and can correlate the DNS response to an application. The IP address returned from the DNS response is cached and used for later packet flows associated with that specific application.

The command **ip nbar custom** *application-name* **dns** *domain-name* **id** *application-id* is used for DNS customization. To extend an existing application, use the command **ip nbar custom** *application-name* **dns domain-name** *domain-name* **extends** *existing-application*.

For more information on DNS based customization, see http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_nbar/configuration/xe-3s/asr1000/qos-nbar-xe-3s-asr-1000-book/nbar-custapp-dns-xe.html .

DNS Customization

Custom application called MYDNS using the DNS domain name “mydomain.com” with selector ID 12.

```
Device# configure terminal
Device(config)# ip nbar custom MYDNS dns domain-name *mydomain.com id 12
```

Composite Customization

NBAR2 provides a way to customize applications based on domain names appearing in HTTP, SSL or DNS.

Composite Customization

Custom application called MYDOMAIN using HTTP, SSL or DNS domain name “mydomain.com” with selector ID 13.

```
Device# configure terminal
Device(config)# ip nbar custom MYDOMAIN composite server-name *mydomain.com id 13
```

L3/L4 Customization

Layer3/Layer4 customization is based on the packet tuple and is always matched on the first packet of a flow.

L3/L4 Customization

Custom application called LAYER4CUSTOM matching IP addresses 10.56.1.10 and 10.56.1.11, TCP and DSCP ef with selector ID 14.

```
Device# configure terminal
Device(config)# ip nbar custom LAYER4CUSTOM transport tcp id 14
Device(config-custom)# ip address 10.56.1.10 10.56.1.11
Device(config-custom)# dscp ef
```

Examples: Monitoring Custom Applications

Show Commands for Monitoring Custom Applications

show ip nbar protocol-id | inc Custom

```
Device# show ip nbar  protocol-id | inc Custom
LAYER4CUSTOM              14              Custom
MYDNS                     12              Custom
MYDOMAIN                  13              Custom
MYHTTP                    10              Custom
MYSSL                     11              Custom
```

show ip nbar protocol-discovery protocol *CUSTOM_APP*

```
WSW-157# show ip nbar  protocol-id MYSSL
Protocol Name               id              type
-----------------------------------------------
MYSSL                     11              Custom
```

NBAR2 Dynamic Hitless Protocol Pack Upgrade

Protocol packs are software packages that update the NBAR2 protocol support on a device without replacing the Cisco software on the device. A protocol pack contains information on applications officially supported by NBAR2 which are compiled and packed together. For each application, the protocol-pack includes information on application signatures and application attributes. Each software release has a built-in protocol-pack bundled with it.

Protocol packs provide the following features:

- They are easy and fast to load.
- They are easy to upgrade to a higher version protocol pack or revert to a lower version protocol pack.
- They do not require the switch to be reloaded.

NBAR2 protocol packs are available for download on Cisco Software Center from this URL: https://software.cisco.com/download/navigator.html .

Prerequisites for the NBAR2 Protocol Pack

Before loading a new protocol pack, you must copy the protocol pack to the flash on all the switch members.

To load a protocol pack, see Examples: Loading the NBAR2 Protocol Pack, on page 21 .

Loading the NBAR2 Protocol Pack

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip nbar protocol-pack** *protocol-pack* [**force**]
4. **exit**
5. **show ip nbar protocol-pack** {protocol-pack | **active**} [**detail**]

DETAILED STEPS

	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	**ip nbar protocol-pack** *protocol-pack* [**force**]
	
	Example:
	`Device(config)# ip nbar protocol-pack flash:defProtoPack`
	
	Example:
	`Device(config)# default ip nbar protocol-pack`
		Loads the protocol pack.
		• Use the **force** keyword to specify and load a protocol pack of a lower version, which is different from the base protocol pack version. This also removes the configuration that is not supported by the current protocol pack on the switch.
		For reverting to the built-in protocol pack, use the following command:

Step 4	**exit** **Example:** `Device(config)# exit`	Returns to privileged EXEC mode.
Step 5	**show ip nbar protocol-pack** {protocol-pack	**active**} [**detail**]	Displays the protocol pack information. • Verify the loaded protocol pack version, publisher, and other details using this command.

	Command or Action	Purpose
	Example: `Device# show ip nbar protocol-pack active`	• Use the *protocol-pack* argument to display information about the specified protocol pack. • Use the **active** keyword to display active protocol pack information. • Use the **detail** keyword to display detailed protocol pack information.

Examples: Loading the NBAR2 Protocol Pack

The following example shows how to load a new protocol pack:

```
Device> enable
Device# configure terminal
Device(config)# ip nbar protocol-pack flash:newDefProtoPack
Device(config)# exit
```

The following example shows how to use the **force** keyword to load a protocol pack of a lower version:

```
Device> enable
Device# configure terminal
Device(config)# ip nbar protocol-pack flash:OldDefProtoPack force
Device(config)# exit
```

The following example shows how to revert to the built-in protocol pack:

```
Device> enable
Device# configure terminal
Device(config)# default ip nbar protocol-pack
Device(config)# exit
```

Monitoring Application Visibility and Control

Monitoring Application Visibility and Control (CLI)

This section describes the new commands for application visibility.

The following commands can be used to monitor application visibility on the and access ports.

Table 1: Monitoring Application Visibility Commands on the

Command	Purpose

show ip nbar protocol-discovery [**interface** *interface-type interface-number*] [**stats**{**byte-count** | **bit-rate** | **packet-count** | **max-bit-rate**}] [**protocol** *protocol-name* | **top-n** *number*]	Displays the statistics gathered by the NBAR Protocol Discovery feature. • (Optional) Enter keywords and arguments to fine-tune the statistics displayed. For more information on each of the keywords, refer to the **show ip nbar protocol-discovery**command in Cisco IOS Quality of Service Solutions Command Reference.
show policy-map interface *interface-type interface-number*	Displays information about policy map applied to the interface.
show platform software fed switch *switch id* **wdavc flows**	Displays statistics about all flows on the specified switch.

Examples: Application Visibility and Control

Examples: Application Visibility and Control Configuration

This example shows how to create class maps with apply match protocol filters for application name:

```
Device# configure terminal
Device(config)# class-map match-any NBAR-VOICE
Device(config-cmap)# match protocol ms-lync-audio
Device(config-cmap)#end
```

This example shows how to create policy maps and define existing class maps for egress QoS:

```
Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class cat-browsing
Device(config-pmap-c)# police 150000
Device(config-pmap-c)# set dscp 12
Device(config-pmap-c)#end
```

This example shows how to create policy maps and define existing class maps for ingress QoS:

```
Device# configure terminal
Device(config)# policy-map test-avc-down
Device(config-pmap)# class cat-browsing
Device(config-pmap-c)# police 200000
Device(config-pmap-c)# set dscp 10
Device(config-pmap-c)#end
```

This example shows how to apply policy maps to a switch port:

```
Device# configure terminal
Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# switchport mode access
Device(config-if)# switchport access vlan 20
Device(config-if)# service-policy type control subscriber POLICING_IN
Device(config-if)#end
```

Show Commands for Viewing the Configuration

show ip nbar protocol-discovery

Displays a report of the Protocol Discovery statistics per interface.

The following is a sample output for the statistics per interface:

```
Deviceqos-cat3k-reg2-r1# show ip nbar protocol-discovery int GigabitEthernet1/0/1
GigabitEthernet1/0/1

Last clearing of "show ip nbar protocol-discovery" counters 00:03:16

                                     Input
Output
                                     -----
------
Protocol                             Packet Count
Packet Count
                                     Byte Count
Byte Count
                                     30sec Bit Rate (bps)
30sec Bit Rate (bps)
                                     30sec Max Bit Rate (bps)
30sec Max Bit Rate (bps)
---------------------- ------------------------
-------------------------------------------------
ms-lyc                               60580
55911
                                     31174777
28774864
                                     3613000
93000
                                     3613000
3437000
Total                                60580
55911
                                     31174777
28774864
                                     3613000
93000
                                     3613000
3437000
```

show policy-map interface

Displays the QoS statistics and the configured policy maps on all interfaces.

The following is a sample output for the policy-maps configured on all the interfaces:

```
Deviceqos-cat3k-reg2-r1# show policy-map int
GigabitEthernet1/0/1

 Service-policy input: MARKING-IN

   Class-map: NBAR-VOICE (match-any)
     718 packets
     Match: protocol ms-lync-audio
       0 packets, 0 bytes
       30 second rate 0 bps
     QoS Set
       dscp ef

   Class-map: NBAR-MM_CONFERENCING (match-any)
     6451 packets
     Match: protocol ms-lync
       0 packets, 0 bytes
       30 second rate 0 bps
     Match: protocol ms-lync-video
       0 packets, 0 bytes
```

```
    30 second rate 0 bps
  QoS Set
    dscp af41

Class-map: class-default (match-any)
  34 packets
  Match: any
```

Basic Troubleshooting(Questions and Answers)

Following are the basic questions and answers for troubleshooting wired Application Visibility and Control:

1 **Question:** My IPv6 traffic is not being classified.

Answer: Currently only IPv4 traffic is supported.

2 **Question:** My multicast traffic is not being classified

Answer: Currently only unicast traffic is supported

3 **Question:** I send ping but I don't see them being classified

Answer: Only TCP/UDP protocols are supported

4 **Question:** Why can't I attach NBAR to an SVI?

Answer: NBAR is only supported on physical interfaces.

5 **Question:** I see that most of my traffic is CAPWAP traffic, why?

Answer: Make sure that you have enabled NBAR on an access port that is not connected to a wireless access port. All traffic coming from AP's will be classified as capwap. Actual classification in this case happens either on the AP or WLC.

6 **Question:** In protocol-discovery, I see traffic only on one side. Along with that, there are a lot of unknown traffic.

Answer: This usually indicates that NBAR sees asymmetric traffic: one side of the traffic is classified in one switch member and the other on a different member. The recommendation is to attach NBAR only on access ports where we see both sides of the traffic. If you have multiple uplinks, you can't attach NBAR on them due to this issue. Similar issue happens if you configure NBAR on an interface that is part of a port channel.

7 **Question:** With protocol-discovery, I see an aggregate view of all application. How can I see traffic distribution over time?

Answer: WebUI will give you view of traffic over time for the last 48 hours.

8 **Question:** I can't configure queue-based egress policy with **match protocol** *protocol-name* command.

Answer: Only **shape** and **set DSCP** are supported in a policy with NBAR2 based classifiers. Common practice is to set DSCP on ingress and perform shaping on egress based on DSCP.

9 **Question:** I don't have NBAR2 attached to any interface but I still see that NBAR2 is activated.

Answer: If you have any class-map with **match protocol** *protocol-name*, NBAR will be globally activated on the stack but no traffic will be subjected to NBAR classification. This is an expected behavior and it does not consume any resources.

10 **Question:** I see some traffic under the default QOS queue. Why?

Answer: For each new flow, it takes a few packets to classify it and install the result in the hardware. During this time, the classification would be 'un-known' and traffic will fall under the default queue.

Additional References for Application Visibility and Control

Related Documents

Related Topic	Document Title
QoS	*NBAR Configuration Guide, Cisco IOS XE 16*
NBAR2 Protocol Pack Hitless Upgrade	*NBAR Configuration Guide, Cisco IOS XE 16*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature History and Information For Application Visibility and Control in a Wired Network

Release	Feature Information
Cisco IOS XE Denali 16.3.2	Wired AVC Flexible NetFlow (FNF) — The feature uses a flow record with an application name as the key, to provide client, server and application statistics, per interface.
Cisco IOS XE Denali 16.3.1	This feature was introduced.

Finding Feature Information

Your software release may not support all of the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release.

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

Information About Application Visibility and Control

Application Visibility and Control (AVC) classifies applications using deep packet inspection techniques with the Network-Based Application Recognition engine, and provides application-level visibility and control (QoS) in wireless networks. After the applications are recognized, the AVC feature enables you to either drop, mark, or police the data traffic.

AVC is configured by defining a class map in a QoS client policy to match a protocol.

Using AVC, we can detect more than 1000 applications. AVC enables you to perform real-time analysis and create policies to reduce network congestion, costly network link usage, and infrastructure upgrades.

Note You can view list of 30 applications in Top Applications in Monitor Summary section of the UI.

Traffic flows are analyzed and recognized using the NBAR2 engine at the access point. For more information about the NBAR2 Protocol Library, see http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_nbar/prot_lib/config_library/nbar-prot-pack-library.html. The specific flow is marked with the recognized protocol or application, such as WebEx. This per-flow information can be used for application visibility using Flexible NetFlow (FNF).

AVC QoS actions are applied with AVC filters in both upstream and downstream directions. The QoS actions supported for upstream flow are drop, mark, and police, and for downstream flow are mark and police. AVC QoS is applicable only when the application is classified correctly and matched with the class map filter in the policy map. For example, if the policy has a filter based on an application name, and the traffic has also been classified to the same application name, then the action specified for this match in the policy will be applied.

Application Visibility and Control Protocol Packs

Protocol packs are a means to distribute protocol updates outside the switch software release trains, and can be loaded on the switch without replacing the switch software.

The Application Visibility and Control Protocol Pack (AVC Protocol Pack) is a single compressed file that contains multiple Protocol Description Language (PDL) files and a manifest file. A set of required protocols can be loaded, which helps AVC to recognize additional protocols for classification on your network. The manifest file gives information about the protocol pack, such as the protocol pack name, version, and some information about the available PDLs in the protocol pack.

The AVC Protocol Packs are released to specific AVC engine versions. You can load a protocol pack if the engine version on the switch platform is the same or higher than the version required by the protocol pack.

Supported AVC Class Map and Policy Map Formats

Supported AVC Class Map Format

Class Map Format	Class Map Example	Direction
match protocol *protocol name*	`class-map match-any webex-class match protocol webex-media`	Both upstream and downstream
match protocol attribute category *category-name*	`class-map match-any IM match protocol attribute category instant-messaging`	Both upstream and downstream
match protocol attribute sub-category *sub-category-name*	`class-map match-any realtimeconferencing match protocol attribute sub-category voice-video-chat-collaboration`	Both upstream and downstream
match protocol attribute application-group *application-group-name*	`class-map match-any skype match protocol attribute application-group skype-group`	Both upstream and downstream
Combination filters	`class-map match-any webex-class match protocol webex match dscp 45 match wlan user-priority 6`	Upstream only

Supported AVC Policy Format

Policy Format	QoS Action
Upstream client policy based on match protocol filter	Mark, police, and drop
Downstream client policy based on match protocol filter	Mark and police

The following table describes the detailed AVC policy format with an example:

AVC Policy Format	AVC Policy Example	Direction

Basic set	`policy-map webex-policy`
	` class webex-class`
	` set dscp ef //or set up,cos`
		Upstream and downstream

Basic police	`policy-map webex-policy`
	` class webex-class`
	` police 5000000`
		Upstream and downstream

Basic set and police	`policy-map webex-policy`
	` class webex-class`
	` set dscp ef //or set up,cos`
	` police 5000000`
		Upstream and downstream

Multiple set and police including default	`policy-map webex-policy`
	` class webex-class`
	` set dscp af31 //or set up,cos`
	` police 4000000`
	` class class-webex-category`
	` set dscp ef //or set up,cos`
	` police 6000000`
	` class class-default`
	` set dscp <>`
		Upstream and downstream

Hierarchical police	`policy-map webex-policy`
	` class webex-class`
	` police 5000000`
	` service-policy client-in-police-only`
	
	` policy-map client-in-police-only`
	` class webex-class`
	` police 100000`
	` class class-webex-category`
	` set dscp ef //or set up,cos`
	` police 6000000`
	` police 200000`
		Upstream and downstream

Hierarchical set and police	`policy-map webex-policy`
	` class class-default`
	` police 1500000`
	` service policy client-up-child`
	` policy-map webex-policy`
	` class webex-class`
	` police 100000`
	` set dscp ef`
	` class class-webex-category`
	` police 200000`
	` set dscp af31`	

Drop action		Upstream only

AVC Policy Format	AVC Policy Example	Direction

	Any of the above examples apply to this format with this additional example:
	`policy-map webex-policy`
	` class webex-class`
	` drop`
	` class netflix`
	` set dscp ef //or set up,cos`
	` police 6000000`
	` class class-default`
	` set dscp <>`	

Prerequisites for Application Visibility and Control

- The access points should be AVC capable.
- For the control part of AVC (QoS) to work, the application visibility feature with FNF has to be configured.

Guidelines for Inter-Device Roaming with Application Visibility and Control

Follow these guidelines to prevent clients from getting excluded due to malformed QoS policies:

- When a new QoS policy is added to the device, a QoS policy with the same name should be added to other device within the same roam or mobility domain.
- When a device is loaded with a software image of a later release, the new policy formats are supported. If you have upgraded the software image from an earlier release to a later release, you should save the configuration separately. When an earlier release image is loaded, some QoS policies might show as not supported, and you should restore those QoS policies to supported policy formats.

Restrictions for Application Visibility and Control

- AVC is supported only on the following access points:
  - Cisco Aironet 1260 Series Access Points
  - Cisco Aironet 1600 Series Access Points
  - Cisco Aironet 2600 Series Access Point
  - Cisco Aironet 2600 Series Wireless Access Points
  - Cisco Aironet 2700 Series Access Point
  - Cisco Aironet 3500 Series Access Points

  - Cisco Aironet 3600 Series Access Points

- AVC is not supported on Cisco Aironet 702W, 702I (128 M memory), and 1530 Series Access Points.
- Dropping or marking of the data traffic (control part) is not supported for software Release 3.3.
- Dropping or marking of the data traffic (control part) is supported in software Release 3E.
- Only the applications that are recognized with application visibility can be used for applying QoS control.
- Multicast traffic classification is not supported.
- Only the applications that are recognized with App visibility can be used for applying QoS control.
- IPv6 including ICMPv6 traffic classifications are not supported.
- Datalink is not supported for NetFlow fields for AVC.
- The following commands are not supported for AVC flow records:
  - **collect flow username**
  - **collect interface** { **input** | **output**}
  - **collect wireless client ipv4 address**
  - **match interface** { **input** | **output**}
  - **match transport igmp type**
- The template timeout cannot be modified on exporters configured with AVC. Even if the template timeout value is configured to a different value, only the default value of 600 seconds is used.
- For the username information in the AVC-based record templates, ensure that you configure the options **records** to get the user MAC address to username mapping. For more information, refer Creating a Flow Exporter (Optional), on page 34.
- When there is a mix of AVC-enabled APs such as 3600, and non-AVC-enabled APs such as 1140, and the chosen policy for the client is AVC-enabled, the policy will not be sent to the APs that cannot support AVC.
- Only ingress AVC statistics are supported. The frequency of statistics updates depends on the number of clients loaded at the AP at that time. Statistics are not supported for very large policy format sizes.
- The total number of flows for which downstream AVC QoS supported per client is 1000.
- The maximum number of flows supported for Catalyst 3850 Series Switch is 48 K.
- These are some class map and policy map-related restrictions. For supported policy formats, see Supported AVC Class Map and Policy Map Formats, on page 27.
  - AVC and non-AVC classes cannot be defined together in a policy in a downstream direction. For example, when you have a class map with match protocol, you cannot use any other type of match filter in the policy map in the downstream direction.
  - Drop action is not applicable for the downstream AVC QoS policy.
  - Match protocol is not supported in ingress or egress for SSID policy.

- Google shares resources among several of their services because of which for some of the traffic it is not possible to say it is unique to one application. Therefore we added google-services for traffic that cannot be distinguished. The behavior you experience is expected.
- AVC is not supported on management port (Gig 0/0).
- NBAR based QoS policy configuration is allowed only on wired physical ports. Policy configuration is not supported on virtual interfaces, for example, VLAN, Port-Channel and other logical interfaces.
- NBAR and NetFlow cannot be configured together at the same time on the same interface.

How to Configure Application Visibility and Control

Configuring Application Visibility and Control (CLI)

To configure Application Visibility, follow these general steps:

1. Create a flow record by specifying keys and non-key fields to the flow.
2. Create an optional flow exporter by specifying the flow record as an option.
3. Create a flow monitor based on the flow record and flow exporter.
4. Configure WLAN to apply flow monitor in IPv4 input or output direction.

To configure Application Control, follow these general steps:

1. Create an AVC QoS policy.
2. Attach AVC QoS policy to the client in one of three ways: configuring WLAN, using ACS or ISE, or adding local policies.

To enable application recognition on an interface, see Enabling Application Recognition on an interface .

Creating a Flow Record

By default, **wireless avc basic** (flow record) is available. When you click **Apply** from the GUI, then the record is mapped to the flow monitor.

Default flow record cannot be edited or deleted. If you require a new flow record, you need to create one and map it to the flow monitor from CLI.

SUMMARY STEPS

1. **configure terminal**
2. **flow record** *flow_record_name*
3. **description** *string*
4. **match ipv4 protocol**
5. **match ipv4 source address**
6. **match ipv4 destination address**
7. **match transport source-port**
8. **match transport destination-port**
9. **match flow direction**
10. **match application name**
11. **match wireless ssid**
12. **collect counter bytes long**
13. **collect counter packets long**
14. **collect wireless ap mac address**
15. **collect wireless client mac address**
16. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**flow record** *flow_record_name*
	
	Example:
	`Device(config)# flow record record1`
	`Device (config-flow-record)#`
		Enters flow record configuration mode.

Step 3	**description** *string* **Example:** `Device(config-flow-record)# description IPv4flow`	(Optional) Describes the flow record as a maximum 63-character string.
Step 4	**match ipv4 protocol** **Example:** `Device (config-flow-record)# match ipv4 protocol`	Specifies a match to the IPv4 protocol.

	Command or Action	Purpose
Step 5	**match ipv4 source address** **Example:** `Device (config-flow-record)# match ipv4 source address`	Specifies a match to the IPv4 source address-based field.
Step 6	**match ipv4 destination address** **Example:** `Device (config-flow-record)# match ipv4 destination address`	Specifies a match to the IPv4 destination address-based field.
Step 7	**match transport source-port** **Example:** `Device (config-flow-record)# match transport source-port`	Specifies a match to the transport layer source-port field.
Step 8	**match transport destination-port** **Example:** `Device (config-flow-record)# match transport destination-port`	Specifies a match to the transport layer destination-port field.
Step 9	**match flow direction** **Example:** `Device (config-flow-record)# match flow direction`	Specifies a match to the direction the flow was monitored in.
Step 10	**match application name** **Example:** `Device (config-flow-record)# match application name`	Specifies a match to the application name. **Note** This action is mandatory for AVC support, as this allows the flow to be matched against the application.
Step 11	**match wireless ssid** **Example:** `Device (config-flow-record)# match wireless ssid`	Specifies a match to the SSID name identifying the wireless network.
Step 12	**collect counter bytes long** **Example:** `Device (config-flow-record)# collect counter bytes long`	Specifies to collect counter fields total bytes.
Step 13	**collect counter packets long** **Example:** `Device (config-flow-record)# collect counter bytes long`	Specifies to collect counter fields total packets.

	Command or Action	Purpose
Step 14	**collect wireless ap mac address** **Example:** `Device (config-flow-record)# collect wireless ap mac address`	Specifies to collect the BSSID with MAC addresses of the access points that the wireless client is associated with.
Step 15	**collect wireless client mac address** **Example:** `Device (config-flow-record)# collect wireless client mac address`	Specifies to collect MAC address of the client on the wireless network. **Note** The **collect wireless client mac address** is mandatory configuration for wireless AVC.
Step 16	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Creating a Flow Exporter (Optional)

You can create a flow export to define the export parameters for a flow. This is an optional procedure for configuring flow parameters.

SUMMARY STEPS

1. **configure terminal**
2. **flow exporter** *flow_exporter_name*
3. **description** *string*
4. **destination** *{hostname | ip-address}*
5. **transport udp** *port-value*
6. **option application-table timeout** *seconds* **(optional)**
7. **option usermac-table timeout** *seconds* **(optional)**
8. **end**
9. **show flow exporter**
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose

Step 2	**flow exporter** *flow_exporter_name*
	
	Example:
	`Device(config)# flow exporter record1`
	`Device (config-flow-exporter)#`
		Enters flow exporter configuration mode.

Step 3	**description** *string* **Example:** `Device(config-flow-exporter)# description IPv4flow`	Describes the flow record as a maximum 63-character string.
Step 4	**destination** *{hostname | ip-address}* **Example:** `Device (config-flow-exporter) # destination 10.99.1.4`	Specifies the hostname or IPv4 address of the system to which the exporter sends data.
Step 5	**transport udp** *port-value* **Example:** `Device (config-flow-exporter) # transport udp 2`	Configures a port value for the UDP protocol.
Step 6	**option application-table timeout** *seconds* **(optional)** **Example:** `Device (config-flow-exporter)# option application-table timeout 500`	(Optional) Specifies application table timeout option. The valid range is from 1 to 86400 seconds.
Step 7	**option usermac-table timeout** *seconds* **(optional)** **Example:** `Device (config-flow-exporter)# option usermac-table timeout 1000`	(Optional) Specifies wireless usermac-to-username table option. The valid range is from 1 to 86400 seconds.
Step 8	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.
Step 9	**show flow exporter** **Example:** `Device # show flow exporter`	Verifies your configuration.
Step 10	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Creating a Flow Monitor

You can create a flow monitor and associate it with a flow record and a flow exporter.

SUMMARY STEPS

1. **configure terminal**
2. **flow monitor** *monitor-name*
3. **description** *description*
4. **record** *record-name*
5. **exporter** *exporter-name*
6. **cache timeout** {**active** | **inactive**} **(Optional)**
7. **end**
8. **show flow monitor**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**flow monitor** *monitor-name* **Example:** `Device (config)# flow monitor flow-monitor-1`	Creates a flow monitor and enters flow monitor configuration mode.
Step 3	**description** *description* **Example:** `Device (config-flow-monitor)# description flow-monitor-1`	Creates a description for the flow monitor.
Step 4	**record** *record-name* **Example:** `Device (config-flow-monitor)# record flow-record-1`	Specifies the name of a recorder that was created previously.
Step 5	**exporter** *exporter-name* **Example:** `Device (config-flow-monitor)# exporter flow-exporter-1`	Specifies the name of an exporter that was created previously.

	Command or Action	Purpose

Step 6	**cache timeout** {**active** | **inactive**} **(Optional)**
	
	Example:
	`Device (config-flow-monitor)# cache timeout active 1800`
	
	`Device (config-flow-monitor)# cache timeout inactive 200`
		Specifies to configure flow cache parameters. You can configure for a time period of 1 to 604800 seconds (optional).
		
		Note To achieve optimal result for the AVC flow monitor, we recommend you to configure the inactive cache timeout value to be greater than 90 seconds.

Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.
Step 8	**show flow monitor** **Example:** `Device # show flow monitor`	Verifies your configuration.

Creating AVC QoS Policy

To create AVC QoS policy, perform these general steps:

1. Create a class map with match protocol filters.
2. Create a policy map.
3. Apply a policy map to the client in one of the following ways:
   a. Apply a policy map over WLAN either from the CLI or GUI.
   b. Apply a policy map through the AAA server (ACS server or ISE) from the CLI.

      For more information, refer to the *Cisco Identity Services Engine User Guide* and *Cisco Secure Access Control System User Guide*.
   c. Apply local policies either from the CLI or GUI.

Creating a Class Map

You need to create a class map before configuring any match protocol filter. The QoS actions such as marking, policing, and dropping can be applied to the traffic. The AVC match protocol filters are applied only for the wireless clients. For more information about the protocols that are supported, see http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_nbar/prot_lib/config_library/nbar-prot-pack-library.html.

SUMMARY STEPS

1. **configure terminal**
2. **class-map** *class-map-name*
3. **match protocol** {*application-name* | **attribute category** *category-name* | **attribute sub-category** *sub-category-name* | **attribute application-group** *application-group-name*}
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**class-map** *class-map-name* **Example:** `Device(config)# class-map webex-class`	Creates a class map.

Step 3	**match protocol** {*application-name* | **attribute category** *category-name* | **attribute sub-category** *sub-category-name* | **attribute application-group** *application-group-name*}
	
	Example:
	`Device(config)# class-map webex-class`
	`Device(config-cmap)# match protocol webex-media`
	
	`Device(config)# class-map class-webex-category`
	`Device(config-cmap)# match protocol attribute category webex-media`
	
	`Device# class-map class-webex-sub-category`
	`Device(config-cmap)# match protocol attribute sub-category webex-media`
	
	`Device# class-map class-webex-application-group`
	`Device(config-cmap)# match protocol attribute application-group webex-media`
		Specifies match to the application name, category name, subcategory name, or application group.

Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Creating a Policy Map

SUMMARY STEPS

1. **configure terminal**
2. **policy-map** *policy-map-name*
3. **class** [*class-map-name* | **class-default**]
4. **police** *rate-bps burst-byte* [**exceed-action** {**drop** | **policed-dscp-transmit**}]
5. **set** {**dscp** *new-dscp* | **cos** *cos-value*}
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy-map-name*
	
	Example:
	`Device(config)# policy-map webex-policy`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.
		The default behavior of a policy map is to set the DSCP to 0 if the packet is an IP packet and to set the CoS to 0 if the packet is tagged. No policing is performed.
		Note To delete an existing policy map, use the **no policy-map** *policy-map-name* global configuration command.

Step 3	**class** [*class-map-name*	**class-default**]
		
		Example:
		`Device(config-pmap)# class-map webex-class`
		`Device(config-pmap-c)#`
			Defines a traffic classification, and enters policy-map class configuration mode.
			By default, no policy map and class maps are defined.
			If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command.
			A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** is included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.
			Note To delete an existing class map, use the **no class** *class-map-name* policy-map configuration command.

Step 4	**police** *rate-bps burst-byte* [**exceed-action** {**drop**	**policed-dscp-transmit**}] **Example:** `Device(config-pmap-c)# police 100000 80000 drop`	Defines a policer for the classified traffic. By default, no policer is defined. • For *rate-bps,* specify an average traffic rate in bits per second (b/s). The range is 8000 to 10000000000. • For *burst-byte,* specify the normal burst size in bytes. The range is 8000 to 1000000.

	Command or Action	Purpose
		• (Optional) Specifies the action to take when the rates are exceeded. Use the **exceed-action drop** keywords to drop the packet. Use the **exceed-action policed-dscp-transmit** keywords to mark down the DSCP value (by using the policed-DSCP map) and to send the packet.
Step 5	**set** {**dscp** *new-dscp* | **cos** *cos-value*} **Example:** `Device(config-pmap-c)# set dscp 45`	Classifies IP traffic by setting a new value in the packet. • For **dscp** *new-dscp*, enter a new DSCP value to be assigned to the classified traffic. The range is 0 to 63.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

What to Do Next

After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Local Policies (CLI)

Configuring Local Policies (CLI)

To configure local policies, complete these procedures:

1. Create a service template.
2. Create an interface template.
3. Create a parameter map.
4. Create a policy map.
5. Apply a local policy on a WLAN.

Creating a Service Template (CLI)

SUMMARY STEPS

1. **configure terminal**
2. **service-template** *service-template-name*
3. **access-group** *acl_list*
4. **vlan** *vlan_id*
5. **absolute-timer** *seconds*
6. **service-policy qos** {**input** | **output**}
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**service-template** *service-template-name*
	
	Example:
	
	`Device(config)# service-template cisco-phone-template`
	`Device(config-service-template)#`
		Enters service template configuration mode.

Step 3	**access-group** *acl_list* **Example:** `Device(config-service-template)# access-group foo-acl`	Specifies the access list to be applied.
Step 4	**vlan** *vlan_id* **Example:** `Device(config-service-template)# vlan 100`	Specifies VLAN ID. You can specify a value from 1 to 4094.
Step 5	**absolute-timer** *seconds* **Example:** `Device(config-service-template)# absolute-timer 20`	Specifies session timeout value for service template. You can specify a value from 1 to 65535.
Step 6	**service-policy qos** {**input** | **output**} **Example:** `Device(config-service-template)# service-policy qos input foo-qos`	Configures QoS policies for the client.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Creating a Parameter Map (CLI)

Parameter map is preferred to use than class map.

SUMMARY STEPS

1. **configure terminal**
2. **parameter-map type subscriber attribute-to-service** *parameter-map-name*
3. *map-index* **map** { **device-type** | **mac-address** | **oui** | **user-role** | **username**} {**eq** | **not-eq** | **regex** *filter-name* }
4. **interface-template** *interface-template-name*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**parameter-map type subscriber attribute-to-service** *parameter-map-name* **Example:** `Device(config)# parameter-map type subscriber attribute-to-service Aironet-Policy-para`	Specifies the parameter map type and name.
Step 3	*map-index* **map** { **device-type**	**mac-address**	**oui**	**user-role**	**username**} {**eq**	**not-eq**	**regex** *filter-name* } **Example:** `Device(config-parameter-map-filter)# 10 map device-type eq "WindowsXP-Workstation"`	Specifies parameter map attribute filter criteria.

Step 4	**interface-template** *interface-template-name*
	
	Example:
	
	`Device(config-parameter-map-filter-submode)# interface-template cisco-phone-template`
	`Device(config-parameter-map-filter-submode)#`
		Enters service template configuration mode.

Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Related Topics

Information About Configuring Local Policies
Restrictions for Configuring Local Policies

Monitoring Local Policies
Examples: Local Policies Configuration

Creating a Policy Map (CLI)

SUMMARY STEPS

1. **configure terminal**
2. **policy-map type control subscriber** *policy-map-name*
3. **event identity-update** {**match-all** | **match-first**}
4. *class_number* **class** {*class_map_name* | **always** } {**do-all** | **do-until-failure** | **do-until-success**}
5. *action-index* **map attribute-to-service table** *parameter-map-name*
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**policy-map type control subscriber** *policy-map-name* **Example:** `Device(config)# policy-map type control subscriber Aironet-Policy`	Specifies the policy map type.
Step 3	**event identity-update** {**match-all**	**match-first**} **Example:** `Device(config-policy-map)# event identity-update match-all`	Specifies match criteria to the policy map.
Step 4	*class_number* **class** {*class_map_name*	**always** } {**do-all**	**do-until-failure**	**do-until-success**} **Example:** `Device(config-class-control-policymap)# 1 class local_policy1_class do-until-success`	Configures the local profiling policy class map number and specifies how to perform the action. The class map configuration mode includes the following command options: • **always—**Executes without doing any matching but return success. • **do-all—**Executes all the actions. • **do-until-failure—**Execute all the actions until any match failure is encountered. This is the default value. • **do-until-success—**Execute all the actions until any match success happens.
Step 5	*action-index* **map attribute-to-service table** *parameter-map-name*	Specifies parameter map table to be used.

	Command or Action	Purpose
	Example: `Device(config-policy-map)# 10 map attribute-to-service table Aironet-Policy-para`	
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Related Topics

Information About Configuring Local Policies
Restrictions for Configuring Local Policies
Monitoring Local Policies
Examples: Local Policies Configuration

Applying a Local Policy for a Device on a WLAN (CLI)

Before You Begin

If the service policy contains any device type-based rules in the parameter map, ensure that the device classifier is already enabled.

Note You should use the **device classification** command to classify the device for it to be displayed correctly on the show command output.

SUMMARY STEPS

1. **configure terminal**
2. **wlan** *wlan-name*
3. **service-policy type control subscriber** *policymapname*
4. **profiling local http (optional)**
5. **profiling radius http (optional)**
6. **no shutdown**
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**wlan** *wlan-name* **Example:** `Device(config)# wlan wlan1`	Enters WLAN configuration mode.
Step 3	**service-policy type control subscriber** *policymapname* **Example:** `Device(config-wlan)# service-policy type control subscriber Aironet-Policy`	Applies local policy to WLAN.
Step 4	**profiling local http (optional)** **Example:** `Device(config-wlan)# profiling local http`	Enables only profiling of devices based on HTTP protocol (optional).
Step 5	**profiling radius http (optional)** **Example:** `Device(config-wlan)# profiling radius http`	Enables profiling of devices on ISE (optional).
Step 6	**no shutdown** **Example:** `Device(config-wlan)# no shutdown`	Specifies not to shut down the WLAN.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Related Topics

Information About Configuring Local Policies
Restrictions for Configuring Local Policies
Monitoring Local Policies
Examples: Local Policies Configuration

Configuring WLAN to Apply Flow Monitor in IPV4 Input/Output Direction

SUMMARY STEPS

1. **configure terminal**
2. **wlan** *wlan-id*
3. **ip flow monitor** *monitor-name* **{input | output}**
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**wlan** *wlan-id* **Example:** `Device (config) # wlan 1`	Enters WLAN configuration submode. For *wlan-id*, enter the WLAN ID. The range is 1 to 64.
Step 3	**ip flow monitor** *monitor-name* **{input | output}** **Example:** `Device (config-wlan) # ip flow monitor flow-monitor-1 input`	Associates a flow monitor to the WLAN for input or output packets.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode. Alternatively, you can also press **Ctrl-Z** to exit global configuration mode.

Monitoring Application Visibility and Control

Monitoring Application Visibility and Control (CLI)

This section describes the new commands for application visibility.

The following commands can be used to monitor application visibility on the and access points.

Table 2: Monitoring Application Visibility Commands on the

Command	Purpose

show avc client *client-mac* **top** *n* **application** [**aggregate** | **upstream** | **downstream**]	Displays information about top "N" applications for the given client MAC.
show avc wlan *ssid* **top** *n* **application** [**aggregate** | **upstream** | **downstream**]	Displays information about top "N" applications for the given SSID.
avc top user[enable | disable]	Enables or disables the information about top "N" application.
show avc wlan *wlan-id* **application** *app name* **top***N* [**aggregate** | **upstream** | **downstream**]	Displays to know network usage information on a per user basis within an application. **Note** On Catalyst 4500E Supervisor Engine 8-E, in the information about top N users that is displayed, the client's MAC address and username are not displayed. This issue occurs only within 90 seconds after the client is disconnected.
show wlan id *wlan-id*	Displays information whether AVC is enabled or disabled on a particular WLAN.
show flow monitor *flow_monitor_name* **cache**	Displays information about flow monitors.
show wireless client mac-address *mac-address* **service-policy** { **input** | **output** }	Displays information about policy mapped to the wireless clients.
show ip nbar protocol-discovery [**interface** *interface-type interface-number*] [**stats**{**byte-count** | **bit-rate** | **packet-count** | **max-bit-rate**}] [**protocol** *protocol-name* | **top-n** *number*]	Displays the statistics gathered by the NBAR Protocol Discovery feature. • (Optional) Enter keywords and arguments to fine-tune the statistics displayed. For more information on each of the keywords, refer to the **show ip nbar protocol-discovery**command in Cisco IOS Quality of Service Solutions Command Reference. **Note** When you configure NBAR, you must enable Protocol Discovery on the interface.
show policy-map target **show policy-map** **show policy-map** *policy-name* **show policy-map interface** *interface-type interface-number*	Displays information about policy map.

Table 3: Clearing Application Visibility Statistics Commands

Command	Purpose

clear avc client *mac* **stats**	Clears the statistics per client.
clear avc wlan *wlan-name* **stats**	Clears the statistics per WLAN.

Examples: Application Visibility and Control

Examples: Application Visibility Configuration

This example shows how to create a flow record, create a flow monitor, apply the flow record to the flow monitor, and apply the flow monitor on a WLAN:

```
Device# configure terminal
Device(config)# flow record fr_v4
Device(config-flow-record)# match ipv4 protocol
Device(config-flow-record)# match ipv4 source address
Device(config-flow-record)# match ipv4 destination address
Device(config-flow-record)# match transport destination-port
Device(config-flow-record)# match flow direction
Device(config-flow-record)# match application name
Device(config-flow-record)# match wireless ssid
Device(config-flow-record)# collect counter bytes long
Device(config-flow-record)# collect counter packets long
Device(config-flow-record)# collect wireless ap mac address
Device(config-flow-record)# collect wireless client mac address
Device(config)#end


Device# configure terminal
Device# flow monitor fm_v4
Device(config-flow-monitor)# record fr_v4
Device(config-flow-monitor)# cache timeout active 1800
Device(config)#end


Device(config)#wlan wlan1
Device(config-wlan)#ip flow monitor fm_v4 input
Device(config-wlan)#ip flow mon fm-v4 output
Device(config)#end
```

Examples: Application Visibility and Control QoS Configuration

This example shows how to create class maps with apply match protocol filters for application name, category, and subcategory:

```
Device# configure terminal
Device(config)# class-map cat-browsing
Device(config-cmap)# match protocol attribute category browsing
Device(config-cmap)#end

Device# configure terminal
Device(config)# class-map cat-fileshare
Device(config-cmap)# match protocol attribute category file-sharing
Device(config-cmap)#end

Device# configure terminal
Device(config)# class-map match-any subcat-terminal
```

```
Device(config-cmap)# match protocol attribute sub-category terminal
Device(config-cmap)#end

Device# configure terminal
Device(config)# class-map match-any webex-meeting
Device(config-cmap)# match protocol webex-meeting
Device(config-cmap)#end
```

This example shows how to create policy maps and define existing class maps for upstream QoS:

```
Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class cat-browsing
Device(config-pmap-c)# police 150000
Device(config-pmap-c)# set dscp 12
Device(config-pmap-c)#end


Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class cat-fileshare
Device(config-pmap-c)# police 1000000
Device(config-pmap-c)# set dscp 20
Device(config-pmap-c)#end


Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class subcat-terminal
Device(config-pmap-c)# police 120000
Device(config-pmap-c)# set dscp 15
Device(config-pmap-c)#end

Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class webex-meeting
Device(config-pmap-c)# police 50000000
Device(config-pmap-c)# set dscp 21
Device(config-pmap-c)#end
```

This example shows how to create policy maps and define existing class maps for downstream QoS:

```
Device# configure terminal
Device(config)# policy-map test-avc-down
Device(config-pmap)# class cat-browsing
Device(config-pmap-c)# police 200000
Device(config-pmap-c)# set dscp 10
Device(config-pmap-c)#end


Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class cat-fileshare
Device(config-pmap-c)# police 300000
Device(config-pmap-c)# set wlan user-priority 2
Device(config-pmap-c)# set dscp 20
Device(config-pmap-c)#end


Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class subcat-terminal
Device(config-pmap-c)# police 100000
Device(config-pmap-c)# set dscp 25
Device(config-pmap-c)#end

Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class webex-meeting
```

```
Device(config-pmap-c)# police 60000000
Device(config-pmap-c)# set dscp 41
Device(config-pmap-c)#end
```

This example shows how to apply defined QoS policy on a WLAN:

```
Device# configure terminal
Device(config)#wlan  alpha
Device(config-wlan)#shut
Device(config-wlan)#end
Device(config-wlan)#service-policy client input test-avc-up
Device(config-wlan)#service-policy client output test-avc-down
Device(config-wlan)#no shut
Device(config-wlan)#end
```

Example: Configuring QoS Attribute for Local Profiling Policy

The following example shows how to configure QoS attribute for a local profiling policy:

```
Device(config)# class-map type control subscriber match-all local_policy1_class
Device(config-filter-control-classmap)# match device-type android
Device(config)# service-template local_policy1_template
Device(config-service-template)# vlan 40
Device(config-service-template)# service-policy qos output local_policy1
Device(config)# policy-map type control subscriber local_policy1
Device(config-event-control-policymap)# event identity-update match-all
Device(config-class-control-policymap)# 1 class local_policy1_class do-until-success
Device(config-action-control-policymap)# 1 activate service-template local_policy1_template
Device(config)# wlan open_auth 9
Device(config-wlan)# client vlan VLAN40
Device(config-wlan)# service-policy type control subscriber local_policy1
```

Additional References for Application Visibility and Control

Related Documents

Related Topic	Document Title
System management commands	*System Management Command Reference Guide, Cisco IOS XE Release 3SE (Cisco WLC 5700 Series)*
Flexible NetFlow configuration	*Flexible NetFlow Configuration Guide, Cisco IOS XE Release 3SE (Cisco WLC 5700 Series)*
Flexible NetFlow commands	*Flexible NetFlow Command Reference, Cisco IOS XE Release 3SE (Cisco WLC 5700 Series)*
QoS configuration	*QoS Configuration Guide, Cisco IOS XE Release 3E (Cisco WLC 5700 Series)*
QoS commands	*QoS Command Reference, Cisco IOS XE Release 3E (Cisco WLC 5700 Series)*

Standards and RFCs

Standard/RFC	Title
None	—

MIBs

MIB	MIBs Link
All supported MIBs for this release.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature History and Information For Application Visibility and Control

Release	Feature Information
Cisco IOS XE 3.3SE	This feature was introduced.
Cisco IOS XE 3E	AVC control with QoS was introduced.