



Configuring Application Visibility and Control in a Wired Network

- [Information About Application Visibility and Control in a Wired Network, on page 1](#)
- [Supported AVC Class Map and Policy Map Formats, on page 2](#)
- [Restrictions for Wired Application Visibility and Control, on page 3](#)
- [How to Configure Application Visibility and Control, on page 4](#)
- [Monitoring Application Visibility and Control, on page 29](#)
- [Examples: Application Visibility and Control Configuration, on page 30](#)
- [Basic Troubleshooting - Questions and Answers, on page 42](#)
- [Additional References for Application Visibility and Control, on page 43](#)
- [Feature History for Application Visibility and Control in a Wired Network, on page 43](#)

Information About Application Visibility and Control in a Wired Network



Note This feature is not supported on the C9500-32C, C9500-32QC, C9500-48Y4C, and C9500-24Y4C models of the Cisco Catalyst 9500 Series Switches.

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. 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

This section describes the supported avc class maps and policy map formats.

Supported AVC Class Map Format

| Class Map Format | Class Map Example | Direction |
|--|--|-------------------------|
| <code>match protocol <i>protocol name</i></code> | <code>class-map match-any NBAR-VOICE match protocol ms-lync-audio</code> | Both ingress and egress |
| Combination filters | <code>class-map match-any NBAR-VOICE match protocol ms-lync-audio match dscp ef</code> | 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 | <code>policy-map MARKING-IN class NBAR-MM_CONFERENCEING set dscp af41</code> | Ingress and egress |
| Basic police | <code>policy-map POLICING-IN class NBAR-MM_CONFERENCEING police cir 600000 set dscp af41</code> | Ingress and egress |
| Basic set and police | <code>policy-map webex-policy class webex-class set dscp ef police 5000000</code> | Ingress and egress |
| Multiple set and police including default | <code>policy-map webex-policy class webex-class set dscp af31 police 4000000 class class-webex-category set dscp ef police 6000000 class class-default set dscp <></code> | Ingress and egress |

| AVC Policy Format | AVC Policy Example | Direction |
|-----------------------------|--|--------------------|
| Hierarchical police | <pre> policy-map webex-policy class webex-class police 500000 service-policy client-in-police-only policy-map client-in-police-only class webex-class police 100000 class class-webex-category set dscp ef police 200000 </pre> | Ingress and egress |
| Hierarchical set and police | <pre> policy-map webex-policy class class-default police 1500000 service-policy client-up-child policy-map client-up-child class webex-class police 100000 set dscp ef class class-webex-category police 200000 set dscp af31 </pre> | |

Restrictions for Wired Application Visibility and Control

- AVC and Encrypted Traffic Analytics (ETA) cannot be configured together at the same time on the same interface.
- NBAR and transmit (Tx) Switched Port Analyzer (SPAN) is not supported on the same interface.
- Only one of the NBAR based QoS mechanisms are allowed to be attached to any port at the same time, either protocol based or attributes based. Only the following two attributes are supported :
 - traffic-class
 - business-relevance
- The legacy WDAVC QoS limitations are still applicable:
 - Only marking and policing are supported.
 - Only physical interfaces are supported.
 - There is a delay in the QoS classification since the application classification is done offline (while the initial packet/s of the flow are meanwhile forwarded before the correct QoS classification).
- 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).
- 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.
To manage and check wired AVC traffic on the Web UI, you must first configure **ip http authentication local** and **ip nbar http-service** commands using the CLI.
- 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.
- Starting with Cisco IOS XE Fuji 16.9.1, up to two wired AVC monitors each with a different predefined record can be attached to an interface at the same time.
- Two new directional flow records - ingress and egress - have been introduced in Cisco IOS XE Fuji 16.9.1, in addition to the two existing legacy flow records.
- Attachment should be done only on physical Layer 2 and Layer 3 ports, and these ports cannot be part of a port channel. Attachment to trunk ports are not supported.
- Performance: Each switch member is able to handle 2000 connections per second (CPS) at less than 50% CPU utilization.
- Scale: Able to handle up to 20,000 bi-directional flows per 48 access ports and 10,000 bi-directional flows per 24 access ports. (~200 flows per access port).
- Wired AVC allows only the fixed set of fields listed in the procedures of this chapter. Other combinations are not allowed. For a regular FNF flow monitor, other combinations are allowed (for the list of supported FNF fields, refer the "Configuring Flexible NetFlow" chapter of the *Network Management Configuration Guide*).
- Starting with Cisco IOS XE 16.12.1 release, a new flow record has been included - the DNS flow record. The DNS flow record is similar to the 5-tuple record and includes the DNS domain name field. It accounts only for DNS related fields. This record doesn't have the interface field as a match field, so the information from all interfaces is aggregated into the same record.

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 the section, "Enabling Application Recognition on an Interface."

Configuring Control : Configure QoS policies based on application by

1. Creating an AVC QoS policy. See the section, "Creating AVC QoS Policy".
2. Applying AVC QoS policy to the interface. See the section, "Applying a QoS Policy to the Switch Port".

Configuring application-based Flexible Netflow :

- Create a flow record by specifying key and non-key fields to the flow.
- Create a flow exporter to export the flow record.
- Create a flow monitor based on the flow record and the flow exporter.
- Attach the flow monitor to the interface.

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:

Procedure

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

Procedure

| | Command or Action | Purpose |
|---------------|--|---|
| Step 1 | terminal Example: Device# <code>configure terminal</code> | Enters global configuration mode. |
| Step 2 | class-map <i>class-map-name</i> Example: Device(config)# <code>class-map webex-class</code> | Creates a class map. |
| Step 3 | match protocol <i>application-name</i> Example: Device(config)# <code>class-map webex-class</code> Device(config-cmap)# <code>match protocol webex-media</code> | Specifies match to the application name. |
| Step 4 | end Example: Device(config)# <code>end</code> | Returns to privileged EXEC mode. Alternatively, you can also press Ctrl-Z to exit global configuration mode. |

Creating a Policy Map

Procedure

| | Command or Action | Purpose |
|---------------|---|-----------------------------------|
| Step 1 | configure terminal Example: Device# <code>configure terminal</code> | Enters global configuration mode. |

| | Command or Action | Purpose |
|--------|--|---|
| Step 2 | <p>policy-map <i>policy-map-name</i></p> <p>Example:</p> <pre>Device(config)# policy-map webex-policy</pre> | <p>Creates a policy map by entering the policy map name, and enters policy-map configuration mode.</p> <p>By default, no policy maps are defined.</p> <p>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.</p> <p>Note To delete an existing policy map, use the no policy-map <i>policy-map-name</i> global configuration command.</p> |
| Step 3 | <p>class [<i>class-map-name</i> class-default]</p> <p>Example:</p> <pre>Device(config-pmap)# class webex-class</pre> | <p>Defines a traffic classification, and enters policy-map class configuration mode.</p> <p>By default, no policy map and class maps are defined.</p> <p>If a traffic class has already been defined by using the class-map global configuration command, specify its name for <i>class-map-name</i> in this command.</p> <p>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.</p> <p>Note To delete an existing class map, use the no class <i>class-map-name</i> policy-map configuration command.</p> |
| Step 4 | <p>police <i>rate-bps burst-byte</i></p> <p>Example:</p> <pre>Device(config-pmap-c)# police 100000 80000</pre> | <p>Defines a policer for the classified traffic.</p> <p>By default, no policer is defined.</p> <ul style="list-style-type: none"> For <i>rate-bps</i>, specify an average traffic rate in bits per second (b/s). The range is 8000 to 10000000000. For <i>burst-byte</i>, specify the normal burst size in bytes. The range is 1000 to 512000000. |
| Step 5 | <p>set { dscp <i>new-dscp</i> cos <i>cos-value</i> }</p> <p>Example:</p> | <p>Classifies IP traffic by setting a new value in the packet.</p> |

| | Command or Action | Purpose |
|---------------|--|--|
| | Device(config-pmap-c) # set dscp 45 | <ul style="list-style-type: none"> 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

Procedure

| | 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

Wired AVC FNF supports two types of predefined flow records — Legacy Bidirectional flow records and Directional flow records (ingress and egress). A total of four different predefined flow records, two bidirectional flow records and two directional flow records, can be configured and associated with a flow monitor. The legacy bidirectional records are client/server application statistics records, and the new directional records are application-stats for input/output.

Bidirectional Flow Records

Flow Record 1 - Bidirectional Flow Record

Procedure

| | Command or Action | Purpose |
|----------------|---|--|
| Step 1 | configure terminal Example: Device# configure terminal | Enters global configuration mode. |
| Step 2 | flow record <i>flow_record_name</i> Example: Device(config)# flow record fr-wdavic-1 | Enters flow record configuration mode. |
| Step 3 | description <i>description</i> Example: Device(config-flow-record)# description fr-wdavic-1 | (Optional) Creates a description for the flow record. |
| 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: | Specifies a match to the observation point ID for flow observation metrics. |

| | Command or Action | Purpose |
|----------------|---|---|
| | Device (config-flow-record) # match flow observation point | |
| 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 : <ul style="list-style-type: none"> • 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 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 : <ul style="list-style-type: none"> • 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 new-connections Example: Device (config-flow-record) # collect connection new-connections | Specifies to collect the number of connection initiations observed. |
| Step 14 | 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 15 | collect connection client counter bytes network long Example: | Specifies to collect the total number of bytes transmitted by the client. |

| | Command or Action | Purpose |
|----------------|---|---|
| | Device (config-flow-record) # collect connection client counter bytes network long | |
| Step 16 | 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 17 | 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 18 | 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 19 | 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. |
| 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. |

Flow Record 2 - Bidirectional Flow Record

Procedure

| | Command or Action | Purpose |
|---------------|---|--|
| Step 1 | configure terminal Example: Device# configure terminal | Enters global configuration mode. |
| Step 2 | flow record <i>flow_record_name</i> Example: | Enters flow record configuration mode. |

| | Command or Action | Purpose |
|----------------|---|--|
| | Device (config) # flow record fr-wdavic-1 | |
| Step 3 | description <i>description</i> Example: Device (config-flow-record) # description fr-wdavic-1 | (Optional) Creates a description for the flow record. |
| 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 client transport port Example: Device (config-flow-record) # match connection client transport port | (Optional) Specifies a match to the connection port of the client as a key field for a flow record. |
| Step 9 | 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 10 | 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 11 | 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. |

| | Command or Action | Purpose |
|---------|---|---|
| Step 12 | <p>collect flow direction</p> <p>Example:</p> <pre>Device(config-flow-record)# collect flow direction</pre> | <p>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 :</p> <ul style="list-style-type: none"> • 0x01 = Ingress Flow • 0x02 = Egress Flow <p>When the initiator keyword is set to initiator, the flow direction is specified from the initiator side of the flow. When 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.</p> |
| Step 13 | <p>collect connection initiator</p> <p>Example:</p> <pre>Device(config-flow-record)# collect connection initiator</pre> | <p>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 :</p> <ul style="list-style-type: none"> • 0x01 = Initiator - the flow source is the initiator of the connection <p>For wired AVC, the initiator keyword is always set to initiator.</p> |
| Step 14 | <p>collect connection new-connections</p> <p>Example:</p> <pre>Device(config-flow-record)# collect connection new-connections</pre> | <p>Specifies to collect the number of connection initiations observed.</p> |
| Step 15 | <p>collect connection client counter packets long</p> <p>Example:</p> <pre>Device(config-flow-record)# collect connection client counter packets long</pre> | <p>Specifies to collect the number of packets sent by the client.</p> |
| Step 16 | <p>collect connection client counter bytes network long</p> <p>Example:</p> | <p>Specifies to collect the total number of bytes transmitted by the client.</p> |

| | Command or Action | Purpose |
|----------------|---|---|
| | <code>Device (config-flow-record) # collect connection client counter bytes network long</code> | |
| Step 17 | collect connection server counter packets long Example: <code>Device (config-flow-record) # collect connection server counter packets long</code> | Specifies to collect the number of packets sent by the server. |
| Step 18 | collect connection server counter bytes network long Example: <code>Device (config-flow-record) # collect connection server counter bytes network long</code> | Specifies to collect the total number of bytes transmitted by the server. |
| Step 19 | collect timestamp absolute first Example: <code>Device (config-flow-record) # collect timestamp absolute first</code> | Specifies to collect the time, in milliseconds, when the first packet was seen in the flow. |
| Step 20 | collect timestamp absolute last Example: <code>Device (config-flow-record) # collect timestamp absolute last</code> | Specifies to collect the time, in milliseconds, when the most recent packet was seen in the flow. |
| Step 21 | end Example: <code>Device (config) # end</code> | Returns to privileged EXEC mode. Alternatively, you can also press Ctrl-Z to exit global configuration mode. |
| Step 22 | show flow record Example: <code>Device# show flow record</code> | Displays information about all the flow records. |

Directional Flow Records

Flow Record 3 - Directional Flow Record - Ingress

Procedure

| | Command or Action | Purpose |
|---------------|---|-----------------------------------|
| Step 1 | configure terminal Example: <code>Device# configure terminal</code> | Enters global configuration mode. |

| | Command or Action | Purpose |
|----------------|---|---|
| Step 2 | flow record <i>flow_record_name</i> Example: Device(config)# flow record fr-wdavic-3 | Enters flow record configuration mode. |
| Step 3 | description <i>description</i> Example: Device(config-flow-record)# description flow-record-1 | (Optional) Creates a description for the flow record. |
| 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 ipv4 source address Example: Device(config-flow-record)# match ipv4 source address | Specifies a match to the IPv4 source address as a key field. |
| Step 7 | match ipv4 destination address Example: Device(config-flow-record)# match ipv4 destination address | Specifies a match to the IPv4 destination address as a key field. |
| Step 8 | match transport source-port Example: Device(config-flow-record)# match transport source-port | Specifies a match to the transport source port as a key field. |
| Step 9 | match transport destination-port Example: Device(config-flow-record)# match transport destination-port | Specifies a match to the transport destination port as a key field. |
| Step 10 | match interface input Example: Device(config-flow-record)# match interface input | Specifies a match to the input interface as a key field. |
| Step 11 | match application name Example: | Specifies a match to the application name. |

| | Command or Action | Purpose |
|----------------|---|--|
| | Device (config-flow-record) # match application name | Note This action is mandatory for AVC support, as this allows the flow to be matched against the application. |
| Step 12 | collect interface output Example: Device (config-flow-record) # collect interface output | Specifies to collect the output interface from the flows. |
| Step 13 | collect counter bytes long Example: Device (config-flow-record) # collect counter bytes long | Specifies to collect the number of bytes in a flow. |
| Step 14 | collect counter packets long Example: Device (config-flow-record) # collect counter packets long | Specifies to collect the number of packets in a flow. |
| Step 15 | 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 16 | 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. |
| Step 17 | end Example: Device (config) # end | Returns to privileged EXEC mode. Alternatively, you can also press Ctrl-Z to exit global configuration mode. |
| Step 18 | show flow record Example: Device # show flow record | Displays information about all the flow records. |

Flow Record 4 - Directional Flow Record - Egress

Procedure

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

| | Command or Action | Purpose |
|----------------|---|---|
| Step 2 | flow record <i>flow_record_name</i> Example: Device(config)# flow record fr-wdavic-4 | Enters flow record configuration mode. |
| Step 3 | description <i>description</i> Example: Device(config-flow-record)# description flow-record-1 | (Optional) Creates a description for the flow record. |
| 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 ipv4 source address Example: Device(config-flow-record)# match ipv4 source address | Specifies a match to the IPv4 source address as a key field. |
| Step 7 | match ipv4 destination address Example: Device(config-flow-record)# match ipv4 destination address | Specifies a match to the IPv4 destination address as a key field. |
| Step 8 | match transport source-port Example: Device(config-flow-record)# match transport source-port | Specifies a match to the transport source port as a key field. |
| Step 9 | match transport destination-port Example: Device(config-flow-record)# match transport destination-port | Specifies a match to the transport destination port as a key field. |
| Step 10 | match interface output Example: Device(config-flow-record)# match interface output | Specifies a match to the output interface as a key field. |
| Step 11 | match application name Example: | Specifies a match to the application name. |

| | Command or Action | Purpose |
|----------------|---|--|
| | Device (config-flow-record) # match application name | Note This action is mandatory for AVC support, as this allows the flow to be matched against the application. |
| Step 12 | collect interface input Example: Device (config-flow-record) # collect interface input | Specifies to collect the input interface from the flows. |
| Step 13 | collect counter bytes long Example: Device (config-flow-record) # collect counter bytes long | Specifies to collect the number of bytes in a flow. |
| Step 14 | collect counter packets long Example: Device (config-flow-record) # collect counter packets long | Specifies to collect the number of packets in a flow. |
| Step 15 | 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 16 | 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. |
| Step 17 | end Example: Device (config) # end | Returns to privileged EXEC mode. Alternatively, you can also press Ctrl-Z to exit global configuration mode. |
| Step 18 | show flow record Example: Device# show flow record | Displays information about all the flow records. |

DNS Flow Record

Flow Record 5 - DNS Flow Record

Procedure

| | Command or Action | Purpose |
|----------------|---|--|
| Step 1 | configure terminal Example: Device# configure terminal | Enters global configuration mode. |
| Step 2 | flow record <i>flow_record_name</i> Example: Device(config)# flow record fr-wdavic-5 | Enters flow record configuration mode. |
| Step 3 | description <i>description</i> Example: Device(config-flow-record)# description flow-record-5 | (Optional) Creates a description for the flow record. |
| 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 client transport port Example: Device(config-flow-record)# match connection client transport port | Specifies a match to the connection port of the client as a key field for a flow record. |
| Step 9 | 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 10 | match connection server transport port Example: | Specifies a match to the transport port of the server. |

| | Command or Action | Purpose |
|----------------|---|---|
| | Device (config-flow-record) # match connection server transport port | |
| Step 11 | collect flow direction Example: Device (config-flow-record) # collect flow direction | <p>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 :</p> <ul style="list-style-type: none"> • 0x01 = Ingress Flow • 0x02 = Egress Flow <p>When the initiator keyword is set to initiator, the flow direction is specified from the initiator side of the flow. When 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.</p> |
| Step 12 | collect timestamp absolute first Example: Device (config-flow-record) # collect timestamp absolute first | <p>Specifies to collect the time, in milliseconds, when the first packet was seen in the flow.</p> |
| Step 13 | collect timestamp absolute last Example: Device (config-flow-record) # collect timestamp absolute last | <p>Specifies to collect the time, in milliseconds, when the most recent packet was seen in the flow.</p> |
| Step 14 | collect connection initiator Example: Device (config-flow-record) # collect connection initiator | <p>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 :</p> <ul style="list-style-type: none"> • 0x01 = Initiator - the flow source is the initiator of the connection <p>For wired AVC, the initiator keyword is always set to initiator.</p> |
| Step 15 | collect connection new-connections Example: Device (config-flow-record) # collect connection new-connections | <p>Specifies to collect the number of connection initiations observed.</p> |

| | Command or Action | Purpose |
|----------------|---|---|
| Step 16 | 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 17 | 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 18 | 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 19 | 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 20 | collect application dns domain-name Example: Device (config-flow-record) # collect application dns domain-name | Configures the use of the DNS Domain-Name as a Collect field for a DNS flow record. |
| Step 21 | 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

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

Procedure

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

| | Command or Action | Purpose |
|---------------|--|--|
| Step 2 | flow exporter <i>flow_exporter_name</i> Example: Device(config)# flow exporter flow-exporter-1 | Enters flow exporter configuration mode. |
| Step 3 | description <i>description</i> Example: Device(config-flow-exporter)# description flow-exporter-1 | (Optional) Creates a description for the flow exporter. |
| Step 4 | destination { <i>hostname</i> <i>ipv4-address</i> <i>ipv6-address</i> } 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 <i>seconds</i>] 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.

Procedure

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

| | Command or Action | Purpose |
|--------|--|---|
| Step 2 | flow monitor <i>monitor-name</i> Example: Device(config)# flow monitor flow-monitor-1 | Creates a flow monitor and enters flow monitor configuration mode. |
| Step 3 | description <i>description</i> Example: Device(config-flow-monitor)# description flow-monitor-1 | (Optional) Creates a description for the flow monitor. |
| Step 4 | record <i>record-name</i> Example: Device(config-flow-monitor)# record flow-record-1 | Specifies the name of a record that was created previously. |
| Step 5 | exporter <i>exporter-name</i> Example: Device(config-flow-monitor)# exporter flow-exporter-1 | Specifies the name of an exporter that was created previously. |
| Step 6 | cache { entries <i>number-of-entries</i> 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. <ul style="list-style-type: none"> • entries <i>number-of-entries</i> — Specifies the maximum number of flow entries in the flow cache in the range from 16 to 65536. Note Only normal cache type is supported. |
| 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. |
| Step 9 | show flow monitor <i>flow-monitor-name</i> Example: Device# show flow monitor flow-monitor-1 | Displays information about the specified wired AVC flow monitor. |

| | Command or Action | Purpose |
|----------------|--|--|
| Step 10 | show flow monitor <i>flow-monitor-name</i> statistics Example: Device# show flow monitor flow-monitor-1 statistics | Displays statistics for wired AVC flow monitor. |
| Step 11 | clear flow monitor <i>flow-monitor-name</i> statistics Example: Device# clear flow monitor flow-monitor-1 statistics | Clears the statistics of the specified flow monitor. Use the show flow monitor flow-monitor-1 statistics command after using the clear flow monitor flow-monitor-1 statistics to verify that all the statistics have been reset. |
| Step 12 | show flow monitor <i>flow-monitor-name</i> cache format table Example: Device# show flow monitor flow-monitor-1 cache format table | Displays flow cache contents in a tabular format. |
| Step 13 | show flow monitor <i>flow-monitor-name</i> cache format record Example: Device# show flow monitor flow-monitor-1 cache format record | Displays flow cache contents in similar format as the flow record. |
| Step 14 | show flow monitor <i>flow-monitor-name</i> cache format csv Example: Device# show flow monitor flow-monitor-1 cache format csv | Displays flow cache contents in CSV format. |

Associating Flow Monitor to an interface

You can attach two different wired AVC monitors with different predefined records to an interface at the same time.

Procedure

| | Command or Action | Purpose |
|---------------|---|--|
| Step 1 | configure terminal Example: Device# configure terminal | Enters global configuration mode. |
| Step 2 | interface <i>interface-id</i> Example: | Enters the interface configuration mode. |

| | Command or Action | Purpose |
|---------------|--|---|
| | Device(config)# interface Gigabitethernet 1/0/1 | |
| Step 3 | ip flow monitor <i>monitor-name</i> { 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/xs-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

```
Device# 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.



Warning When using switch stacking, ensure that each switch has the same Protocol Pack file loaded. If you execute the **ip nbar protocol-pack flash protocol-pack-file** command on the primary switch in the stack, any switch in the stack that does not have the file loaded will be reloaded due to a configuration mismatch.

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

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 [Loading the NBAR2 Protocol Pack, on page 28](#) .

Loading the NBAR2 Protocol Pack

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| Step 1 | enable Example: Device> enable | Enables privileged EXEC mode. <ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • 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: |

| | Command or Action | Purpose |
|---------------|---|--|
| Step 4 | exit Example: Device(config)# exit | Returns to privileged EXEC mode. |
| Step 5 | show ip nbar protocol-pack {protocol-pack active} [detail] Example: Device# show ip nbar protocol-pack active | Displays the protocol pack information. <ul style="list-style-type: none"> • Verify the loaded protocol pack version, publisher, and other details using this command. • Use the <i>protocol-pack</i> 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

This section describes the new commands for application visibility.

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

Table 1: Monitoring Application Visibility Commands on the Switch

| Command | Purpose |
|---------|---------|
|---------|---------|

| | |
|---|---|
| <pre>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]</pre> | <p>Displays the statistics gathered by the NBAR Protocol Discovery feature.</p> <ul style="list-style-type: none"> (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. |
| <pre>show policy-map interface interface-type interface-number</pre> | <p>Displays information about policy map applied to the interface.</p> |
| <pre>show platform software fed switch switch id wdavc flows</pre> | <p>Displays statistics about all flows on the specified switch.</p> |

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 input POLICING_IN
Device(config-if)#end
```

This example shows how to create class maps based on NBAR attributes.

```
Device# configure terminal
Device(config)# class-map match-all rel-relevant
Device(config-cmap)# match protocol attribute business-relevance business-relevant

Device(config)# class-map match-all rel-irrelevant
Device(config-cmap)# match protocol attribute business-relevance business-irrelevant
```

```

Device(config)# class-map match-all rel-default
Device(config-cmap)# match protocol attribute business-relevance default

Device(config)# class-map match-all class--ops-admin-and-rel
Device(config-cmap)# match protocol attribute traffic-class ops-admin-mgmt
Device(config-cmap)# match protocol attribute business-relevance business-relevant

```

This example shows how to create policy maps based on class maps based on NBAR attributes.

```

Device# configure terminal
Device(config)# policy-map attrib--rel-types
Device(config-pmap)# class rel-relevant
Device(config-pmap-c)# set dscp ef
Device(config-pmap-c)# class rel-irrelevant
Device(config-pmap-c)# set dscp af11
Device(config-pmap-c)# class rel-default
Device(config-pmap-c)# set dscp default

Device(config)# policy-map attrib--ops-admin-and-rel
Device(config-pmap)# class class--ops-admin-and-rel
Device(config-pmap-c)# set dscp cs5

```

This example shows how to attach a policy map based on NBAR attributes to a wired port:

```

Device# configure terminal
Device(config)# interface GigabitEthernet1/0/2
Device(config-if)# service-policy input attrib--rel-types

```

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:

```

Device# 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-lync                            60580
55911

```

```

31174777
28774864
3613000
93000
3613000
3437000
Total
55911
60580
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:

```

Device# 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

```

Show Commands for Viewing Attributes-based QoS Configuration**show policy-map interface**

Displays the attribute-based 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:

```
Device# show policy-map interface gigabitEthernet 1/0/2
GigabitEthernet1/0/2

Service-policy input: attrib--rel-types

Class-map: rel-relevant (match-all)
  20 packets
  Match: protocol attribute business-relevance business-relevant
  QoS Set
    dscp ef

Class-map: rel-irrelevant (match-all)
  0 packets
  Match: protocol attribute business-relevance business-irrelevant
  QoS Set
    dscp af11

Class-map: rel-default (match-all)
  14 packets
  Match: protocol attribute business-relevance default
  QoS Set
    dscp default

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

show ip nbar protocol-attribute

Displays all the protocol attributes used by NBAR.

The following shows sample output for some of the attributes:

```
Device# show ip nbar protocol-attribute cisco-jabber-im
  Protocol Name : cisco-jabber-im
    encrypted : encrypted-yes
    tunnel : tunnel-no
    category : voice-and-video
    sub-category : enterprise-media-conferencing
  application-group : cisco-jabber-group
    p2p-technology : p2p-tech-no
    traffic-class : transactional-data
  business-relevance : business-relevant
  application-set : collaboration-apps

Device# show ip nbar protocol-attribute google-services
  Protocol Name : google-services
    encrypted : encrypted-yes
    tunnel : tunnel-no
    category : other
    sub-category : other
  application-group : google-group
  p2p-technology : p2p-tech-yes
```

```

        traffic-class : transactional-data
    business-relevance : default
    application-set : general-browsing
Device# show ip nbar protocol-attribute dns
    Protocol Name : google-services
        encrypted : encrypted-yes
            tunnel : tunnel-no
            category : other
        sub-category : other
    application-group : google-group
    p2p-technology : p2p-tech-yes
    traffic-class : transactional-data
    business-relevance : default
    application-set : general-browsing
Device# show ip nbar protocol-attribute unknown
    Protocol Name : unknown
        encrypted : encrypted-no
            tunnel : tunnel-no
            category : other
        sub-category : other
    application-group : other
    p2p-technology : p2p-tech-no
    traffic-class : bulk-data
    business-relevance : default
    application-set : general-misc

```

Show Commands for Viewing Flow Monitor Configuration

show flow monitor wdavc

Displays information about the specified wired AVC flow monitor.

```
Device # show flow monitor wdavc
```

```
Flow Monitor wdavc:
  Description:      User defined
  Flow Record:     wdavc
  Flow Exporter:   wdavc-exp (inactive)
  Cache:
    Type:           normal (Platform cache)
    Status:         not allocated
    Size:           12000 entries
    Inactive Timeout: 15 secs
    Active Timeout: 1800 secs

```

show flow monitor wdavc statistics

Displays statistics for wired AVC flow monitor.

```
Device# show flow monitor wdavc statistics
  Cache type:      Normal (Platform cache)
  Cache size:      12000
  Current entries: 13

```

```

Flows added:                26
Flows aged:                 13
- Active timeout           ( 1800 secs)    1
- Inactive timeout         (   15 secs)    12
    
```

clear flow monitor wdavc statistics

Clears the statistics of the specified flow monitor. Use the **show flow monitor wdavc statistics** command after using the **clear flow monitor wdavc statistics** to verify that all the statistics have been reset. The following is a sample output of the **show flow monitor wdavc statistics** command after clearing flow monitor statistics.

```

Device# show flow monitor wdavc statistics
Cache type:                Normal (Platform cache)
Cache size:                 12000
Current entries:           0

Flows added:               0
Flows aged:                0
    
```

Show Commands for Viewing Cache Contents

show flow monitor wdavc cache format table

Displays flow cache contents in a tabular format.

```

Device# show flow monitor wdavc cache format table
Cache type:                Normal (Platform cache)
Cache size:                 12000
Current entries:           13

Flows added:               26
Flows aged:                13
- Active timeout           ( 1800 secs)    1
- Inactive timeout         (   15 secs)    12

CONN IPV4 INITIATOR ADDR  CONN IPV4 RESPONDER ADDR  CONN RESPONDER PORT
FLOW OBSPOINT ID  IP VERSION  IP PROT  APP NAME                                flow
-----
-----
-----
64.103.125.147        144.254.71.184
   4294967305         4         17  port dns                                Input
.....
64.103.121.103        10.1.1.2
   4294967305         4         17  layer7 dhcp                             Input
....contd.....
64.103.125.3          64.103.125.97
   4294967305         4         17  layer7 dhcp                             Input
.....
10.0.2.6              157.55.40.149
   4294967305         4         6   layer7 ms-lync                          Input
.....
64.103.126.28        66.163.36.139
    
```

```

        4294967305          4          6 layer7 cisco-jabber-im      Input
    ....contd.....
64.103.125.2              64.103.125.29              68
        4294967305          4          17 layer7 dhcp              Input
    .....
64.103.125.97            64.103.101.181            67
        4294967305          4          17 layer7 dhcp              Input
    .....
192.168.100.6            10.10.20.1                5060
        4294967305          4          17 layer7 cisco-jabber-control Input
    ....contd.....
64.103.125.3              64.103.125.29              68
        4294967305          4          17 layer7 dhcp              Input
    .....
10.80.101.18             10.80.101.6                5060
        4294967305          4          6 layer7 cisco-collab-control Input
    .....
10.1.11.4                 66.102.11.99              80
        4294967305          4          6 layer7 google-services      Input
    ....contd.....
64.103.125.2              64.103.125.97              68
        4294967305          4          17 layer7 dhcp              Input
    .....
64.103.125.29            64.103.101.181            67
        4294967305          4          17 layer7 dhcp              Input
    .....

```

show flow monitor wdavc cache format record

Displays flow cache contents in similar format as the flow record.

```

Device# show flow monitor wdavc cache format record
Cache type:                               Normal (Platform cache)
Cache size:                               12000
Current entries:                          13

Flows added:                              26
Flows aged:                               13
  - Active timeout      ( 1800 secs)      1
  - Inactive timeout    (   15 secs)      12

CONNECTION IPV4 INITIATOR ADDRESS:         64.103.125.147
CONNECTION IPV4 RESPONDER ADDRESS:        144.254.71.184
CONNECTION RESPONDER PORT:                53
FLOW OBSPOINT ID:                         4294967305
IP VERSION:                               4
IP PROTOCOL:                              17
APPLICATION NAME:                         port dns
flow direction:                           Input
timestamp abs first:                      08:55:46.917
timestamp abs last:                       08:55:46.917
connection initiator:                     Initiator
connection count new:                     2

```

```
connection server packets counter:      1
connection client packets counter:     1
connection server network bytes counter: 190
connection client network bytes counter: 106

CONNECTION IPV4 INITIATOR ADDRESS:      64.103.121.103
CONNECTION IPV4 RESPONDER ADDRESS:     10.1.1.2
CONNECTION RESPONDER PORT:             67
FLOW OBSPOINT ID:                     4294967305
IP VERSION:                           4
IP PROTOCOL:                          17
APPLICATION NAME:                      layer7 dhcp
flow direction:                        Input
timestamp abs first:                   08:55:47.917
timestamp abs last:                    08:55:47.917
connection initiator:                  Initiator
connection count new:                  1
connection server packets counter:     0
connection client packets counter:     1
connection server network bytes counter: 0
connection client network bytes counter: 350

CONNECTION IPV4 INITIATOR ADDRESS:      64.103.125.3
CONNECTION IPV4 RESPONDER ADDRESS:     64.103.125.97
CONNECTION RESPONDER PORT:             68
FLOW OBSPOINT ID:                     4294967305
IP VERSION:                           4
IP PROTOCOL:                          17
APPLICATION NAME:                      layer7 dhcp
flow direction:                        Input
timestamp abs first:                   08:55:47.917
timestamp abs last:                    08:55:53.917
connection initiator:                  Initiator
connection count new:                  1
connection server packets counter:     0
connection client packets counter:     4
connection server network bytes counter: 0
connection client network bytes counter: 1412

CONNECTION IPV4 INITIATOR ADDRESS:      10.0.2.6
CONNECTION IPV4 RESPONDER ADDRESS:     157.55.40.149
CONNECTION RESPONDER PORT:             443
FLOW OBSPOINT ID:                     4294967305
IP VERSION:                           4
IP PROTOCOL:                          6
APPLICATION NAME:                      layer7 ms-lync
flow direction:                        Input
timestamp abs first:                   08:55:46.917
timestamp abs last:                    08:55:46.917
connection initiator:                  Initiator
connection count new:                  2
```

```

connection server packets counter:      10
connection client packets counter:     14
connection server network bytes counter: 6490
connection client network bytes counter: 1639

CONNECTION IPV4 INITIATOR ADDRESS:      64.103.126.28
CONNECTION IPV4 RESPONDER ADDRESS:     66.163.36.139
CONNECTION RESPONDER PORT:             443
FLOW OBSPOINT ID:                      4294967305
IP VERSION:                            4
IP PROTOCOL:                           6
APPLICATION NAME:                       layer7 cisco-jabber-im
flow direction:                         Input
timestamp abs first:                    08:55:46.917
timestamp abs last:                     08:55:46.917
connection initiator:                   Initiator
connection count new:                   2
connection server packets counter:      12
connection client packets counter:     10
connection server network bytes counter: 5871
connection client network bytes counter: 2088

CONNECTION IPV4 INITIATOR ADDRESS:      64.103.125.2
CONNECTION IPV4 RESPONDER ADDRESS:     64.103.125.29
CONNECTION RESPONDER PORT:             68
FLOW OBSPOINT ID:                      4294967305
IP VERSION:                            4
IP PROTOCOL:                           17
APPLICATION NAME:                       layer7 dhcp
flow direction:                         Input
timestamp abs first:                    08:55:47.917
timestamp abs last:                     08:55:47.917
connection initiator:                   Initiator
connection count new:                   1
connection server packets counter:      0
connection client packets counter:     2
connection server network bytes counter: 0
connection client network bytes counter: 712

CONNECTION IPV4 INITIATOR ADDRESS:      64.103.125.97
CONNECTION IPV4 RESPONDER ADDRESS:     64.103.101.181
CONNECTION RESPONDER PORT:             67
FLOW OBSPOINT ID:                      4294967305
IP VERSION:                            4
IP PROTOCOL:                           17
APPLICATION NAME:                       layer7 dhcp
flow direction:                         Input
timestamp abs first:                    08:55:47.917
timestamp abs last:                     08:55:47.917
connection initiator:                   Initiator
connection count new:                   1

```

```
connection server packets counter:      0
connection client packets counter:     1
connection server network bytes counter: 0
connection client network bytes counter: 350

CONNECTION IPV4 INITIATOR ADDRESS:      192.168.100.6
CONNECTION IPV4 RESPONDER ADDRESS:     10.10.20.1
CONNECTION RESPONDER PORT:             5060
FLOW OBSPOINT ID:                     4294967305
IP VERSION:                            4
IP PROTOCOL:                           17
APPLICATION NAME:                      layer7 cisco-jabber-control
flow direction:                        Input
timestamp abs first:                   08:55:46.917
timestamp abs last:                    08:55:46.917
connection initiator:                   Initiator
connection count new:                   1
connection server packets counter:     0
connection client packets counter:     2
connection server network bytes counter: 0
connection client network bytes counter: 2046

CONNECTION IPV4 INITIATOR ADDRESS:      64.103.125.3
CONNECTION IPV4 RESPONDER ADDRESS:     64.103.125.29
CONNECTION RESPONDER PORT:             68
FLOW OBSPOINT ID:                     4294967305
IP VERSION:                            4
IP PROTOCOL:                           17
APPLICATION NAME:                      layer7 dhcp
flow direction:                        Input
timestamp abs first:                   08:55:47.917
timestamp abs last:                    08:55:47.917
connection initiator:                   Initiator
connection count new:                   1
connection server packets counter:     0
connection client packets counter:     2
connection server network bytes counter: 0
connection client network bytes counter: 712

CONNECTION IPV4 INITIATOR ADDRESS:      10.80.101.18
CONNECTION IPV4 RESPONDER ADDRESS:     10.80.101.6
CONNECTION RESPONDER PORT:             5060
FLOW OBSPOINT ID:                     4294967305
IP VERSION:                            4
IP PROTOCOL:                           6
APPLICATION NAME:                      layer7 cisco-collab-control
flow direction:                        Input
timestamp abs first:                   08:55:46.917
timestamp abs last:                    08:55:47.917
connection initiator:                   Initiator
connection count new:                   2
```

```

connection server packets counter:      23
connection client packets counter:     27
connection server network bytes counter: 12752
connection client network bytes counter: 8773

CONNECTION IPV4 INITIATOR ADDRESS:      10.1.11.4
CONNECTION IPV4 RESPONDER ADDRESS:     66.102.11.99
CONNECTION RESPONDER PORT:             80
FLOW OBSPOINT ID:                      4294967305
IP VERSION:                             4
IP PROTOCOL:                            6
APPLICATION NAME:                       layer7 google-services
flow direction:                         Input
timestamp abs first:                   08:55:46.917
timestamp abs last:                    08:55:46.917
connection initiator:                   Initiator
connection count new:                   2
connection server packets counter:      3
connection client packets counter:      5
connection server network bytes counter: 1733
connection client network bytes counter: 663

CONNECTION IPV4 INITIATOR ADDRESS:      64.103.125.2
CONNECTION IPV4 RESPONDER ADDRESS:     64.103.125.97
CONNECTION RESPONDER PORT:             68
FLOW OBSPOINT ID:                      4294967305
IP VERSION:                             4
IP PROTOCOL:                            17
APPLICATION NAME:                       layer7 dhcp
flow direction:                         Input
timestamp abs first:                   08:55:47.917
timestamp abs last:                    08:55:53.917
connection initiator:                   Initiator
connection count new:                   1
connection server packets counter:      0
connection client packets counter:      4
connection server network bytes counter: 0
connection client network bytes counter: 1412

CONNECTION IPV4 INITIATOR ADDRESS:      64.103.125.29
CONNECTION IPV4 RESPONDER ADDRESS:     64.103.101.181
CONNECTION RESPONDER PORT:             67
FLOW OBSPOINT ID:                      4294967305
IP VERSION:                             4
IP PROTOCOL:                            17
APPLICATION NAME:                       layer7 dhcp
flow direction:                         Input
timestamp abs first:                   08:55:47.917
timestamp abs last:                    08:55:47.917
connection initiator:                   Initiator
connection count new:                   1

```



```

connection server packets counter:      0
connection client packets counter:     1
connection server network bytes counter: 0
connection client network bytes counter: 350

```

show flow monitor wdvac cache format csv

Displays flow cache contents in CSV format.

```

Device# show flow monitor wdvac cache format csv
Cache type: Normal (Platform cache)
Cache size: 12000
Current entries: 13

Flows added: 26
Flows aged: 13
- Active timeout ( 1800 secs) 1
- Inactive timeout ( 15 secs) 12

```

```

CONN IPV4 INITIATOR ADDR,CONN IPV4 RESPONDER ADDR,CONN RESPONDER PORT,FLOW
OBSPOINT ID,IP VERSION,IP
PROT,APP NAME,flow dirn,time abs first,time abs last,conn initiator,conn
count new,conn server packets
cnt,conn client packets cnt,conn server network bytes cnt,conn client
network bytes cnt
64.103.125.147,144.254.71.184,53,4294967305,4,17,port
dns,Input,08:55:46.917,08:55:46.917,Initiator,2,1,1,190,106
64.103.121.103,10.1.1.2,67,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,1,0,350
64.103.125.3,64.103.125.97,68,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:53.917,Initiator,1,0,4,0,1412
10.0.2.6,157.55.40.149,443,4294967305,4,6,layer7 ms-
lync,Input,08:55:46.917,08:55:46.917,Initiator,2,10,14,6490,1639
64.103.126.28,66.163.36.139,443,4294967305,4,6,layer7 cisco-jabber-
im,Input,08:55:46.917,08:55:46.917,Initiator,2,12,10,5871,2088
64.103.125.2,64.103.125.29,68,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,2,0,712
64.103.125.97,64.103.101.181,67,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,1,0,350
192.168.100.6,10.10.20.1,5060,4294967305,4,17,layer7 cisco-jabber-
control,Input,08:55:46.917,08:55:46.917,Initiator,1,0,2,0,2046
64.103.125.3,64.103.125.29,68,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,2,0,712
10.80.101.18,10.80.101.6,5060,4294967305,4,6,layer7 cisco-collab-
control,Input,08:55:46.917,08:55:47.917,Initiator,2,23,27,12752,8773
10.1.11.4,66.102.11.99,80,4294967305,4,6,layer7 google-
services,Input,08:55:46.917,08:55:46.917,Initiator,2,3,5,1733,663
64.103.125.2,64.103.125.97,68,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:53.917,Initiator,1,0,4,0,1412
64.103.125.29,64.103.101.181,67,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,1,0,350

```

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 switch 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 |
|--|--|
| For complete syntax and usage information for the commands used in this chapter. | <i>Command Reference (Catalyst 9500 Series Switches)</i> |

Feature History for Application Visibility and Control in a Wired Network

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

| Release | Feature | Feature Information |
|--------------------------------|--|--|
| Cisco IOS XE Everest 16.5.1a | Application Visibility and Control in a Wired Network | 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. Support for this feature was introduced only on the C9500-12Q, C9500-16X, C9500-24Q, C9500-40X models of the Cisco Catalyst 9500 Series Switches. |
| Cisco IOS XE Fuji 16.8.1a | Wired Application Visibility and Control (Wired AVC) Attribute-based QoS (EasyQoS) | Support for defining QoS classes and policies based on Network-Based Application Recognition (NBAR) attributes instead of specific protocols, was made available, with a few limitations. Only business-relevance and traffic-class are the supported NBAR attributes. |
| Cisco IOS XE Gibraltar 16.12.1 | DNS flow record | Support for DNS flow record was introduced. DNS flow record uses the DNS Domain-Name as the collect field for defining the flow record. |
| Cisco IOS XE Amsterdam 17.3.1 | Interoperability of Application Visibility and Control and Encrypted Traffic Analytics | Support for interoperability of Application Visibility and Control and Encrypted Traffic Analytics on the same port was introduced. |

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to <http://www.cisco.com/go/cfn>.

