Application Visibility and Control QoS

This document provides detailed understanding about Application Visibility and Control (AVC) QoS and its configuration.

Application Visibility and Control QoS

Application Visibility and Control (AVC) QoS is an application-aware traffic management feature that

- leverages the granular visibility of applications that Cisco AVC solution provides
- identifies, classifies, and prioritizes network traffic based on the specific application, and
- allows network administrators to drop, mark, or rate-limit multiple applications to optimize network performance.

Cisco AVC is a network management solution that

- classifies applications using deep packet inspection (DPI) techniques
- enables application-aware network policies on wired access ports, and
- provides detailed client, server, and application-level traffic statistics.

Deep packet inspection (DPI): AVC uses Network-Based Application Recognition (NBAR2) to recognize applications through Layers 4 to 7. 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.

Benefits of Application Visibility and Control

- Enhanced intelligence: Evolves branch and campus solutions from strictly packet and connection-based management to application-aware, application-intelligent systems.
- Granular visibility: Provides detailed statistics per interface, allowing for better insight into network usage patterns.

  You can configure wired AVC Flexible NetFlow (FNF) on an interface to provide client, server and application statistics per interface.
- Flexible deployment: Supports configuration on wired access ports for standalone switches as well as for a switch stack.
- Performance monitoring: Offers traffic monitoring capabilities similar to Easy Performance Monitor (ezPM) to track application performance and client-server interactions.

  The record is similar to **application-client-server-stats** traffic monitor which is available in **application-statistics** and **application-performance** profiles in ezPM.

Restrictions for Application Visibility and Control

Interface and port limitations

These interface and port limitations apply to NBAR-based QoS policy configuration:

- Configuration is allowed only on wired physical ports. It is supported on Layer 2 access and trunk ports and Layer 3 routed ports.
- Policy configuration is not supported on virtual interfaces like VLAN and other logical interfaces.
- Policy configuration is not supported on port-channel member ports and virtual interfaces like SVIs or sub-interfaces.
- Policy configuration is not supported on management port (Gig 0/0).
- Policies cannot be attached to trunk ports.

Traffic and protocol support

- Use only IPv4 unicast (TCP/UDP) traffic for classification.
- When using `match application name` in a Netflow record, only IPv4 unicast (TCP/UDP) is supported.

IPv6 or multicast traffic is not supported.

Co-existence and operational restrictions

- Only two wired AVC monitors each with a different predefined record can be attached to an interface at the same time.
- Does not support configuring NBAR and transmit (Tx) Switched Port Analyzer (SPAN) on the same interface.
- Does not support configuring NBAR and ACL logging 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.
- You can enable both AVC and Encrypted Traffic Analytics (ETA) on the same port only for IPv4 unicast traffic.
- You cannot configure FNF on an interface when both AVC and ETA are configured on the interface.
- When exporting AVC flow records with an external collector, the interface input field is not included as a match field, so the information from all interfaces is aggregated into the same record.

QoS policy and classification restrictions

- NBAR2-based match criteria, **match protocol**, is allowed only with marking or policing actions. NBAR2 match criteria is not allowed in a policy that has queuing features configured.
- Limit the number of concurrent protocols to 255 across all policies. The hardware limitation is 8 bit.
- 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.
- Use only the `traffic-class` and `business-relevance` attributes for attribute-based QoS.
- Account for a slight delay in QoS classification, as application identification occurs offline while initial packets of the flow are forwarded.

Management and UI restrictions

- Use the Command Line Interface (CLI) for all application control tasks, as 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.

Performance and scalability

- Monitor switch performance, as each member handles up to 2000 connections per second at less than 50% CPU utilization.
- Ensure your deployment stays within the supported limits of 20,000 bi-directional flows per 48 access ports and 10,000 bi-directional flows per 24 access ports. Approximately, 200 flows per access port.
- For wired AVC traffic, the system supports four AVC flow monitors per direction, interface, and IPv4/IPv6 protocol.

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
match protocol *protocol name*	`class-map match-any NBAR-VOICE` `match protocol ms-lync-audio`	ingress

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

Supported AVC policy format

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

The 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

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

Basic set and police	`policy-map webex-policy`
	` class webex-class`
	` set dscp ef`
	` police 5000000`
		ingress

Multiple set and police including default	`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 <>`
		ingress

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`
	`  police 200000`
		ingress

AVC Policy Format	AVC Policy Example	Direction

Hierarchical set and police	`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`
		ingress

Match and collect fields in a wired AVC monitor

You can use any combination of the match and collect fields listed here in a wired AVC monitor.

Application and flow information

- application name
- flow direction
- connection initiator
- flow observation point

IP address information

IPv4:

- IPv4 version
- IPv4 protocol
- IPv4 source address
- IPv4 destination address
- Connection client IPv4 address
- Connection server IPv4 address

IPv6:

- IPv6 version
- IPv6 protocol
- IPv6 source address
- IPv6 destination address
- connection client IPv6 address
- connection server IPv6 address

Transport layer information

- transport TCP flags
- transport source-port
- transport destination-port
- connection client transport port
- connection server transport port

Interface information

- interface input
- interface output

The fields `interface input` and `interface output` can each be used as a match field in a record. But both the fields cannot be used as match fields in the same record. You can used both the fields as collect fields or one as a collect field and one as a match field in the same record.

Supported collect only fields

- connection client counter packets long
- connection client counter bytes network long
- connection server counter packets long
- connection server counter bytes network long
- counter bytes long
- counter packets long
- timestamp absolute first
- timestamp absolute last
- connection new-connections

Extracted fields

- application dns domain-name
- application http host
- application ssl common-name

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.

Local application categories

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, and
- applications specific to a geography

Application customization types

NBAR2 provides a way to manually customize local applications. 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: customization based on multiple underlying protocols
- Layer3 or 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

Customize local applications using NBAR2

Use this procedure to manually customize local applications using NBAR2.

You can manually customize applications using the command **ip nbar custom** *myappname* in global configuration mode.

Procedure

Step 1 Customize general protocols such as HTTP, SSL, or DNS.

Example:

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
```

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

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

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

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
```

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.

To extend an existing application, use the command **ip nbar custom** *application-name* **dns domain-nam** *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.

Step 2 Customize composite or multiple underlying protocols.

Example:

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
```

Step 3 Customize Layer3 or Layer4.

Example:

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
```

L3/L4 customization is based on the packet tuple and is always matched on the first packet of a flow.

Step 4 Monitor custom applications.

Example:

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

```
MYHTTP                    10            Custom
MYSSL                     11            Custom

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 pack features

Protocol packs provide these 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.

Load the NBAR2 protocol pack

Use this procedure to load and activate the NBAR2 protocol pack to expand application recognition capabilities beyond the built-in protocols.

NBAR2 protocol packs contain updated protocol definitions and signatures that enhance the ability to identify and classify network applications. Loading a protocol pack updates the device with the latest application recognition patterns.

Procedure

Step 1 Load the protocol pack.

Example:

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

Here, flash:defProtoPack is the *protocol-pack-location*.

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 this command:

Example:

```
Device(config)# default ip nbar protocol-pack
Device(config)# exit
```

Step 2 Display the protocol pack information and verify the loaded protocol pack version, publisher, and other details.

Example:

```
Device# show ip nbar protocol-pack active
```

- **protocol-pack** keyword: displays information about the specified protocol pack
- **active** keyword: displays active protocol pack information
- **detail** keyword: displays detailed protocol pack information

The NBAR2 protocol pack is successfully loaded and activated, providing enhanced application recognition capabilities with the latest protocol definitions.

Configure Application Visibility and Control QoS

Use this procedure to monitor and control network applications by implementing visibility and control mechanisms on wired network interfaces.

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

Procedure

Step 1 Enable application recognition on an interface, on page 11

Step 2 Configure application-based QoS, on page 11.

Step 3 Configure application-based flexible Netflow.

Enable application recognition on an interface

Use this procedure to enable application recognition on a network interface using the NBAR2 engine.

Procedure

Step 1 Specify the interface for which you are enabling protocol discovery and enter the interface configuration mode.

Example:

```
Device# configure terminal
Device(config)# interface gigabitethernet 1/0/1
```

Step 2 Enable application recognition on the interface by activating NBAR2 engine.

Example:

```
Device(config-if)# ip nbar protocol-discovery
Device(config-if)# end
```

Application recognition is now enabled on the specified interface, allowing the NBAR2 engine to discover and classify application protocols on network traffic.

Configure application-based QoS

Use this procedure to configure application-based QoS on a switch port interface to control traffic flow and prioritization.

QoS policies must be applied to switch port interfaces to enforce traffic management rules. This configuration enables the switch to classify, mark, and prioritize traffic according to the defined policy map.

Policy maps are used to define QoS actions that are applied to classified traffic. You can configure policing to limit traffic rates and set DSCP or CoS values to mark packets for appropriate treatment by network devices.

The class map is created and configured with the specified match criteria for traffic classification.

Procedure

Step 1 Create a class map with match protocol filters.

Example:

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

Example:

This example shows how to match on attributes:

```
Device# configure terminal
Device(config)# class-map match-all audio
Device(config-cmap)# match protocol attribute traffic-class voip-telephony
Device(config-cmap)# match protocol attribute business-relevance business-relevant
Device(config-cmap)# end
```

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, refer http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_nbar/prot_lib/config_library/nbar-prot-pack-library.html.

Step 2 Create a policy map to define traffic classification and apply QoS actions such as policing and marking to network traffic.

a) Create a policy map.

Example:

```
Device# configure terminal
Device(config)# policy-map webex-policy
```

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.

To delete an existing policy map, use the **no policy-map** *policy-map-name* global configuration command.

b) Define a traffic classification.

Example:

```
Device(config-pmap)#  class webex-class
```

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**.

To delete an existing class map, use the **no class** *class-map-name* policy-map configuration command.

c) Define a policer for the classified traffic.

Example:

```
Device(config-pmap-c)# police 100000 80000
```

By default, no policer is defined.

Specify these values:

- 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 1000 to 512000000.

d) Classify IP traffic by setting a new value in the packet.

Example:

```
Device(config-pmap-c)# set dscp 45
Device(config-pmap-c)# end
```

For **dscp** *new-dscp*, enter a new DSCP value to be assigned to the classified traffic. The range is 0 to 63.

Step 3 Apply the QoS policy to a switch port to enforce traffic marking, shaping, and prioritization rules on incoming traffic.

Example:

```
Device# configure terminal
Device(config)#  interface Gigabitethernet 1/0/1
Device(config-if)# service-policy input webex-policy
Device(config-if)# end
```

Configure wired AVC flexible Netflow

Use this procedure to collect and export application-level statistics.

Procedure

Step 1 Create a flow record

a) Create a bidirectional flow record, on page 13
b) Create directional flow records, on page 15
c) Create a DNS flow record, on page 17

Wired AVC flexible Netflow (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.

Step 2 Create a flow exporter, on page 18.

Step 3 Create a flow monitor, on page 19.

Step 4 Associate flow monitor to an interface, on page 21.

Create a flow record

Create a bidirectional flow record

Use this procedure to create a bidirectional flow record that captures traffic flow information including application names, client and server details, and connection statistics for AVC analysis.

Flow records define the key fields and values that are collected from network traffic. A bidirectional flow record captures data from both directions of a connection, providing comprehensive visibility into application traffic patterns and usage statistics.

Wired AVC flexible Netflow 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.

Procedure

Step 1 Create a flow record along with a description.

Example:

```
Device(config)# flow record fr-wdavc-1
Device(config-flow-record)# description fr-wdavc-1
```

Step 2 Specify the respective match criteria.

Example:

```
Device(config-flow-record)# match ipv4 version
Device(config-flow-record)# match ipv4 protocol
Device(config-flow-record)# match application name
Device(config-flow-record)# match connection client ipv4 address
Device(config-flow-record)# match connection server ipv4 address
Device(config-flow-record)# match connection server transport port
Device(config-flow-record)# match flow observation point
```

Note
Configuring **match application name** is mandatory for AVC support, as this allows the flow to be matched against the application.

The client is the flow initiator and the server is the flow responder.

The **match flow observation point** specifies a match to the observation point ID for flow observation metrics.

Step 3 Specify the collection parameters.

Example:

```
Device(config-flow-record)# collect flow direction
Device(config-flow-record)# collect connection initiator
Device(config-flow-record)# collect connection new-connections
Device(config-flow-record)# collect connection client counter packets long
Device(config-flow-record)# collect connection client counter bytes network long
Device(config-flow-record)# collect connection server counter packets long
Device(config-flow-record)# collect connection server counter bytes network long
Device(config-flow-record)# collect timestamp absolute first
Device(config-flow-record)# collect timestamp absolute last
Device(config-flow-record)# end
```

Various collection parameters include:

- **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. Depending on the value specified by the **initiator** keyword, the **flow direction** keyword takes one of these values:

- 0x01 = Ingress Flow
- 0x02 = Egress Flow

When the connection side is set to **initiator**, the flow direction is specified from the initiator side of the flow. When the connection side is set to **responder**, the flow direction is specified from the responder side of the flow. For wired AVC, the connection side is always set to **initiator**.

- **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 tthis information about the direction of the flow:
  - 0x01 = Initiator - the flow source is the initiator of the connection
- **connection new-connections**: specifies to collect the number of connection initiations observed.
- **connection client counter packets long**: specifies to collect the number of packets sent by the client.
- **connection client counter bytes network long**: specifies to collect the total number of bytes transmitted by the client.
- **connection server counter packets long**: specifies to collect the number of packets sent by the server.
- **connection server counter bytes network long**: specifies to collect the total number of bytes transmitted by the server.
- **timestamp absolute first**: specifies to collect the time, in milliseconds, when the first packet was seen in the flow.
- **timestamp absolute last**: specifies to collect the time, in milliseconds, when the most recent packet was seen in the flow.

Step 4 Verify information about all the flow records.

Example:

```
Device# show flow record
```

What to do next

Similarly, configure the ingress and egress directional flow records and DNS flow record.

Create directional flow records

Use this procedure to create separate directional flow records—to capture and analyze ingress and egress network traffic patterns—with application identification capabilities.

You can create separate flow record for ingress and egress traffic by specifying a match to the input or output interface, respectively. Also specify to collect from the output or input interface, respectively.

Procedure

Step 1 Create a flow record along with a description.

Example:

```
Device(config)# flow record fr-wdavc-3
Device(config-flow-record)# description flow-record-3
```

Step 2 Specify the respective match criteria.

Example:

```
Device(config-flow-record)# match ipv4 version
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 source-port
Device(config-flow-record)# match transport destination-port
Device(config-flow-record)# match interface input
Device(config-flow-record)# match application name
```

Note
Configuring **match application name** is mandatory for AVC support, as this allows the flow to be matched against the application.

Use **match interface input** for creating ingress flow record and **match interface output** for creating egress flow record.

Step 3 Specify the collection parameters.

Example:

```
Device(config-flow-record)# collect interface output
Device(config-flow-record)# collect counter bytes long
Device(config-flow-record)# collect counter packets long
Device(config-flow-record)# collect timestamp absolute first
Device(config-flow-record)# collect timestamp absolute last
Device(config-flow-record)# end
```

For creating ingress flow record, specify **collect interface output** and for egress flow record, specify **collect interface input**.

Various collection parameters include:

- **collect interface output**: specifies to collect the output interface from the flows.

  collect interface input: specifies to collect the input interface from the flows.

- **collect counter bytes long**: specifies to collect the number of bytes in a flow.
- **collect counter packets long**: specifies to collect the number of packets in a flow.
- **collect timestamp absolute first**: specifies to collect the time, in milliseconds, when the first packet was seen in the flow.
- **collect timestamp absolute last**: specifies to collect the time, in milliseconds, when the most recent packet was seen in the flow.

Step 4 Verify information about all the flow records.

Example:

```
Device# show flow record
```

Create a DNS flow record

Use this procedure to create a flow record specifically designed to monitor DNS traffic, collecting domain names and connection metrics for network analysis and security purposes.

DNS flow records are used to track domain name resolution patterns and collect detailed statistics about DNS connections. This configuration is essential for network monitoring, security analysis, and application visibility and control (AVC) implementations.

Procedure

Step 1 Create a flow record along with a description.

Example:

```
Device(config)# flow record fr-wdavc-5
Device(config-flow-record)# description flow-record-5
```

Step 2 Specify the respective match criteria.

Example:

```
Device(config-flow-record)# match ipv4 version
Device(config-flow-record)# match ipv4 protocol
Device(config-flow-record)# match application name
Device(config-flow-record)# match connection client ipv4 address
Device(config-flow-record)# match connection client transport port
Device(config-flow-record)# match connection server ipv4 address
Device(config-flow-record)# match connection server transport port
```

Note
Configuring **match application name** is mandatory for AVC support, as this allows the flow to be matched against the application.

Step 3 Specify the collection parameters.

Example:

```
Device(config-flow-record)# collect flow direction
Device(config-flow-record)# collect connection initiator
Device(config-flow-record)# collect connection new-connections
Device(config-flow-record)# collect connection client counter packets long
Device(config-flow-record)# collect connection client counter bytes network long
Device(config-flow-record)# collect connection server counter packets long
Device(config-flow-record)# collect connection server counter bytes network long
Device(config-flow-record)# collect timestamp absolute first
Device(config-flow-record)# collect timestamp absolute last
Device(config-flow-record)# end
```

Various collection parameters include:

- **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. Depending on the value specified by the **initiator** keyword, the **flow direction** keyword takes one of these values:

- 0x01 = Ingress Flow
- 0x02 = Egress Flow

When the connection side is set to **initiator**, the flow direction is specified from the initiator side of the flow. When the connection side is set to **responder**, the flow direction is specified from the responder side of the flow. For wired AVC, the connection side is always set to **initiator**.

- **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 tthis information about the direction of the flow:
  - 0x01 = Initiator - the flow source is the initiator of the connection
- **connection new-connections**: specifies to collect the number of connection initiations observed.
- **connection client counter packets long**: specifies to collect the number of packets sent by the client.
- **connection client counter bytes network long**: specifies to collect the total number of bytes transmitted by the client.
- **connection server counter packets long**: specifies to collect the number of packets sent by the server.
- **connection server counter bytes network long**: specifies to collect the total number of bytes transmitted by the server.
- **timestamp absolute first**: specifies to collect the time, in milliseconds, when the first packet was seen in the flow.
- **timestamp absolute last**: specifies to collect the time, in milliseconds, when the most recent packet was seen in the flow.

Step 4 Configures the use of the DNS Domain-Name as a Collect field for a DNS flow record.

Example:

```
Device(config-flow-record)# collect application dns domain-name
Device(config)# end
```

Step 5 Verify information about all the flow records.

Example:

```
Device# show flow record
```

Create a flow exporter

Use this procedure to create a flow exporter to define how flow data is exported to a collector system.

Flow exporters are used to send flow monitoring data to external systems for analysis. You need to specify the destination and configure export parameters to establish proper data collection.

Procedure

Step 1 Specify a name and description for the flow exporter.

Example:

```
Device# configure terminal
Device(config)# flow exporter flow-exporter-1
Device(config-flow-exporter)# description flow-exporter-1
```

Step 2 Specify the destination—hostname, IPv4 or IPv6 address—of the system to which the exporter sends data.

Example:

```
Device(config-flow-exporter)# destination 10.10.1.1
```

Step 3 (Optional) Configure the application table option for the flow exporter.

Example:

```
Device(config-flow-exporter)# option application-table timeout 500
Device(config)# end
```

The timeout option configures the resend time in seconds for the flow exporter. The valid range is from 1 to 86400 seconds.

Step 4 (Optional) Verify flow exporter configuration.

a) Display information about all the flow exporters.

Example:

```
Device# show flow exporter
```

b) Display flow exporter statistics.

Example:

```
Device# show flow exporter statistics
```

The flow exporter is now configured and ready to send flow data to the specified destination system for monitoring and analysis.

Create a flow monitor

Use this procedure to create a flow monitor to collect, analyze, and export network traffic flow data for monitoring and troubleshooting purposes.

You can create a flow monitor and associate it with a flow record. Flow monitors are used to collect network traffic statistics and can be configured with various cache parameters and export options.

Procedure

Step 1 Create a flow monitor and specify a description.

Example:

```
Device# configure terminal
Device(config)# flow monitor flow-monitor-1
Device(config-flow-monitor)# description flow-monitor-1
```

Step 2 Associate the flow monitor with a flow record that was created previously.

Example:

```
Device(config-flow-monitor)# record flow-record-1
```

Step 3 Associate the flow monitor with a flow exporter that was created previously.

Example:

```
Device(config-flow-monitor)# exporter flow-exporter-1
```

Step 4 (Optional) Configure flow cache parameters.

Example:

```
Device(config-flow-monitor)# cache type normal
Device(config-flow-monitor)# cache timeout active 1800
Device(config-flow-monitor)# cache timeout inactive 200
Device(config)# end
```

The entries option specifies the maximum number of flow entries in the flow cache in the range from 16 to 65536. Only normal cache type is supported.

Step 5 (Optional) Verify flow monitor configuration.

a) Display information about all the flow monitors.

Example:

```
Device# show flow monitor
```

b) Display information about the specified wired AVC flow monitor.

Example:

```
Device# show flow monitor flow-monitor-1
```

c) Display statistics for wired AVC flow monitor.

Example:

```
Device# show flow monitor flow-monitor-1 statistics
```

Step 6 Clear the statistics of the specified flow monitor.

Example:

```
Device# clear flow monitor flow-monitor-1 statistics
```

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 7 Display flow cache contents.

- In tabular format:

  ```
  Device# show flow monitor flow-monitor-1 cache format table
  ```

- In similar format as the flow record:

```
Device# show flow monitor flow-monitor-1 cache format record
```

- In CSV format:

```
Device# show flow monitor flow-monitor-1 cache format csv
```

What to do next

Associate flow monitor to an interface.

Associate flow monitor to an interface

Use this procedure to associate a flow monitor to an interface to enable packet flow monitoring and analysis.

You can attach two different wired AVC monitors with different predefined records to an interface at the same time. This configuration enables flow monitoring for network traffic analysis and application visibility.

Procedure

Step 1 Specify the interface and enter the interface configuration mode.

Example:

```
Device(config)# interface Gigabitethernet 1/0/1
```

Step 2 Associate a flow monitor to the interface for input or output packets, or both.

Example:

```
Device(config-if)# ip flow monitor flow-monitor-1 input
```

The available direction options are: **input** and **output**.

Monitor Application Visibility and Control

Use this procedure to monitor and verify application visibility statistics, flows, and policy configurations to ensure proper Application Visibility and Control (AVC) operation.

AVC provides network administrators with detailed insights into application usage and performance. These monitoring commands help verify AVC functionality and troubleshoot issues on switches and access ports.

Procedure

Step 1 Display the statistics gathered by the NBAR Protocol Discovery feature.

Example:

```
Device# show ip nbar protocol-discovery
```

Optionally, use additional keywords to fine-tune the statistics displayed:

- **interface** *interface-type interface-number*: specifies a particular interface
- **stats** {**byte-count** | **bit-rate** | **packet-count** | **max-bit-rate**}: displays specific statistics types
- **protocol** *protocol-name*: filters by protocol name
- **top-n** *number*: displays top n protocols

Step 2 Display information about policy map applied to the interface.

Example:

```
Device# show policy-map interface interface-type interface-number
```

Step 3 Display statistics about all flows on the specified switch.

Example:

```
Device# show platform software fed switch switch-id wdavc flows
```

Verify application visibility and control in a wired network

Use this procedure to verify the configuration and functionality of application visibility and control in a wired network.

Procedure

Step 1 Verify the configuration.

a) Display a report of the Protocol Discovery statistics per interface.

Example:

```
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                               60580                           55911
                                    31174777                        28774864
                                    3613000                         93000
                                    3613000                         3437000
```

b) Display the QoS statistics and the configured policy maps on all interfaces.

Example:

```
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
```

Step 2 Verify attributes-based QoS configuration.

a) Display the attribute-based QoS statistics and the configured policy maps on all interfaces.

Example:

```
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
```

b) Display all the protocol attributes used by NBAR.

Example:

```
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
```

Step 3 Verify flow monitor configuration.

a) Display information about the specified wired AVC flow monitor.

Example:

```
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
```

b) Display statistics for wired AVC flow monitor.

Example:

```
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
```

You can clear the statistics of the specified flow monitor. Use the **show flow monitor wdavc statistics** command after using the **clear flow monitor wdavc statistics** command to verify that all the statistics have been reset.

This is a sample output 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
```

Step 4 View cache content.

- Display 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 dirn ...................
-----------------------  ------------------------  -------------------
  ----------------  ----------  -------  ---------------------------
 ---------
64.103.125.147            144.254.71.184
53        4294967305           4       17  port dns
   Input     ...................
64.103.121.103            10.1.1.2
67        4294967305           4       17  layer7 dhcp
   Input     ....contd...........
64.103.125.3              64.103.125.97
68        4294967305           4       17  layer7 dhcp
   Input     ...................
10.0.2.6                  157.55.40.149
443        4294967305           4        6  layer7 ms-lync
    Input     ...................
64.103.126.28             66.163.36.139
443        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     .....................
```

- Display 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
```

- Display flow cache contents in CSV format.

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

Troubleshooting guidelines for Application Visibility and Control

Follow these guidelines to resolve common issues and ensure optimal performance when configuring wired Application Visibility and Control:

- Avoid Control and Provisioning of Wireless Access Points (CAPWAP) classification issues: Ensure that NBAR is enabled only on access ports that are not connected to wireless access points. Traffic originating from access points is classified as CAPWAP, as actual classification occurs at the AP or WLC level.
- Prevent asymmetric traffic visibility: Attach NBAR only to access ports where both sides of the traffic flow are visible. Asymmetric traffic—where one side is classified on a different switch member—results in incomplete classification and unknown traffic reports. Do not attach NBAR to uplinks or interfaces that are part of a port channel.
- View traffic distribution over time: Use the WebUI to access a historical view of traffic distribution for the last 48 hours.

- Manage egress policy limitations: The queue-based egress policies are not supported with **match protocol** commands. Use **set DSCP** on ingress and perform shaping on egress based on the DSCP value.
- Understand global NBAR activation: NBAR is globally activated on the stack if any class map uses the **match protocol** command. This is expected behavior and does not consume system resources if no traffic is subjected to classification.
- Account for initial flow classification latency: Expect some traffic to appear in the default QoS queue for new flows. It takes a few packets to classify a flow and install the result in the hardware; during this interval, the classification remains unknown.