AVC Configuration

Revised: July 29, 2013, OL-29700-01

This chapter addresses Cisco AVC configuration and includes the following topics:

Recent Configuration Enhancements

Table 4-1 describes configuration features added in recent releases, and limitations.

Table 4-1 Configuration Features

Feature	Introduced in Release	Information/Limitations
Easy Performance Monitor "express" method of provisioning monitors	IOS XE 3.10	For information, see Easy Performance Monitor, page 4-19

Feature	Introduced in Release	Information/Limitations
Support for configuring 40 fields for each FNF record	IOS XE 3.10	For limitations, see: Downgrading to an IOS XE Version that Does Not Support More than 32 Fields, page 6-2
CLI field aliases	IOS XE 3.10	For limitations, see: Removing Aliases before Downgrading from Cisco IOS XE 3.10, page 6-2

Unified Policy CLI

Beginning with Cisco IOS XE release 3.8, monitoring configuration is done using performance-monitor unified monitor and policy.

```
policy-map type performance-monitor <policy-name>
   [no] parameter default account-on-resolution
   class <class-map name>
      flow monitor <monitor-name> [sampler <sampler name>]
      [sampler <sampler name>]
      monitor metric rtp
```

Usage Guidelines

- Support for:
  - Multiple flow monitors under a class-map.
  - Up to 5 monitors per attached class-map.
  - Up to 256 classes per performance-monitor policy.
- No support for:
  - Hierarchical policy.
  - Inline policy.
- Metric producer parameters are optional.
- Account-on-resolution (AOR) configuration causes all classes in the policy-map to work in AOR mode, which delays the action until the class-map results are finalized (the application is determined by NBAR2).

Attach policy to the interface using following command:

```
interface <interface-name>
  service-policy type performance-monitor <policy-name> {input|output}
```

Metric Producer Parameters

Metric producer-specific parameters are optional and can be defined for each metric producer for each class-map.

```
monitor metric rtp
    clock-rate {type-number| type-name | default} rate
    max-dropout number
    max-reorder number
```

```
min-sequential number
ssrc maximum number
```

Reacts

The **react** CLI defines the alerts applied to a flow monitor. Applying reacts on the device requires punting the monitor records to the route processor (RP) for alert processing. To avoid the performance reduction of punting the monitor records to the RP, it is preferable when possible to send the monitor records directly to the Management and Reporting system and apply the network alerts in the Management and Reporting system.

```
react <id> [media-stop|mrv|rtp-jitter-average|transport-packets-lost-rate]
```

NetFlow/IPFIX Flow Monitor

Flow monitor defines monitor parameters, such as record, exporter, and other cache parameters.

```
flow monitor type performance-monitor <monitor-name>
   record <name | default-rtp | default-tcp>
   exporter <exporter-name>
   history size <size> [timeout <interval>]
   cache entries <num>
   cache timeout {{active | inactive | synchronized} <value> | event transaction end}
   cache type {permanent | normal | immediate}
   react-map <react-map-name>
```

Usage Guidelines

- The **react-map** CLI is allowed under the class in the policy-map. In this case, the monitor must include the exporting of the class-id in the flow record. The route processor (RP) correlates the class-id in the monitor with the class-id where the react is configured.
- Applying history or a react requires punting the record to the RP.
- Export on the “event transaction end” is used to export the records when the connection or transaction is terminated. In this case, the records are not exported based on timeout. Exporting on the event transaction end should be used when detailed connection/transaction granularity is required, and has the following advantages:
  - Sends the record close to the time that it has ended.
  - Exports only one record on true termination.
  - Conserves memory in the cache and reduces the load on the Management and Reporting system.
  - Enables exporting multiple transactions of the same flow. (This requires a protocol pack that supports multi-transaction.)

NetFlow/IPFIX Flow Record

The flow record defines the record fields. With each Cisco IOS release, the Cisco AVC solution supports a more extensive set of metrics.

The sections that follow list commonly used AVC-specific fields as of release IOS XE 3.8, organized by functional groups. These sections do not provide detailed command reference information, but highlight important usage guidelines.

In addition to the fields described below, a record can include any NetFlow field supported by the platform.

A detailed description of NetFlow fields appears in the *Cisco IOS Flexible NetFlow Command Reference*.

Note In this release, the record size is limited to 40 fields (key and non-key fields or match and collect fields).

L3/L4 Fields

The following are L3/L4 fields commonly used by the Cisco AVC solution.

```
[collect | match] connection [client|server] [ipv4|ipv6] address
[collect | match] connection [client|server] transport port
[collect | match] [ipv4|ipv6] [source|destination] address
[collect | match] transport [source-port|destination-port]
[collect | match] [ipv4|ipv6] version
[collect | match] [ipv4|ipv6] protocol
[collect | match] routing vrf [input|output]
[collect | match] [ipv4|ipv6] dscp
[collect | match] ipv4 ttl
[collect | match] ipv6 hop-limit
collect           transport tcp option map
collect           transport tcp window-size [minimum|maximum|sum]
collect           transport tcp maximum-segment-size
```

Usage Guidelines

The client is determined according to the initiator of the connection.

The **client** and **server** fields are bi-directional. The **source** and **destination** fields are uni-directional.

L7 Fields

The following are L7 fields commonly used by the Cisco AVC solution.

```
[collect | match] application name [account-on-resolution]
collect application http url
collect application http uri statistics
collect application http host
collect application http user-agent
collect application http referer
collect application rtsp host-name
collect application smtp server
collect application smtp sender
collect application pop3 server
collect application nntp group-name
collect application sip source
collect application sip destination
```

Usage Guidelines

- The application ID is exported according to RFC-6759.

- Account-On-Resolution configures FNF to collect data in a temporary memory location until the record key fields are resolved. After resolution of the record key fields, FNF combines the temporary data collected with the standard FNF records. Use the **account-on-resolution** option when the field used as a key is not available at the time that FNF receives the first packet.

  The following limitations apply when using Account-On-Resolution:

  - Flows ended before resolution are not reported.
  - FNF packet/octet counters, timestamp, and TCP performance metrics are collected until resolution. All other field values are taken from the packet that provides resolution or the following packets.
- For information about extracted fields, including the formats in which they are exported, see Appendix C, "DPI/L7 Extracted Fields".

Interfaces and Directions

The following are interface and direction fields commonly used by the Cisco AVC solution:

```
[collect | match] interface [input|output]
[collect | match] flow direction
collect connection initiator
```

Counters and Timers

The following are counter and timer fields commonly used by the Cisco AVC solution:

```
collect          connection client counter bytes   [long]
collect          connection client counter packets [long]
collect          connection server counter bytes   [long]
collect          connection server counter packets [long]
collect          counter packets [long]
collect          counter bytes   [long]
collect          counter bytes rate
collect          connection server counter responses
collect          connection client counter packets retransmitted
collect          connection transaction duration            {sum, min, max}
collect          connection transaction counter complete
collect          connection new-connections
collect          connection sum-duration
collect          timestamp sys-uptime first
collect          timestamp sys-uptime last
```

TCP Performance Metrics

The following are fields commonly used for TCP performance metrics by the Cisco AVC solution:

```
collect          connection delay network to-server         {sum, min, max}
collect          connection delay network to-client         {sum, min, max}
collect          connection delay network client-to-server  {sum, min, max}
collect          connection delay response to-server        {sum, min, max}
collect          connection delay response to-server histogram
                                        [bucket1 ... bucket7 | late]
collect          connection delay response client-to-server {sum, min, max}
collect          connection delay application               {sum, min, max}
```

Usage Guidelines

The following limitations apply to TCP performance metrics in AVC for IOS XE 3.10:

- All TCP performance metrics must observe bi-directional traffic.
- The policy-map must be applied in both directions.

Figure 4-1 provides an overview of network response time metrics.

Figure 4-1 Network Response Times

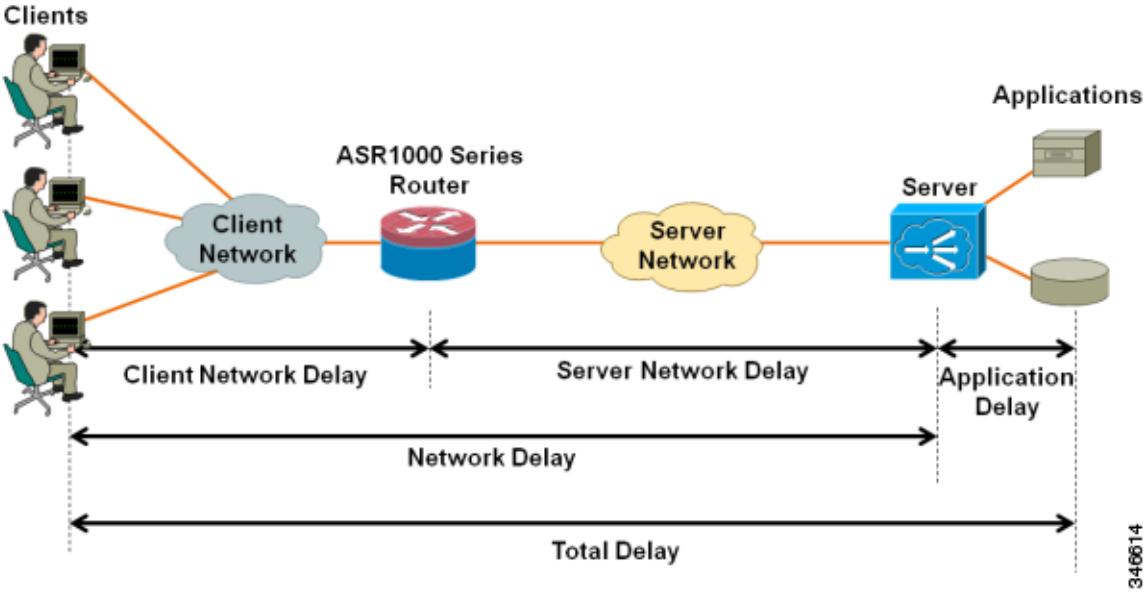
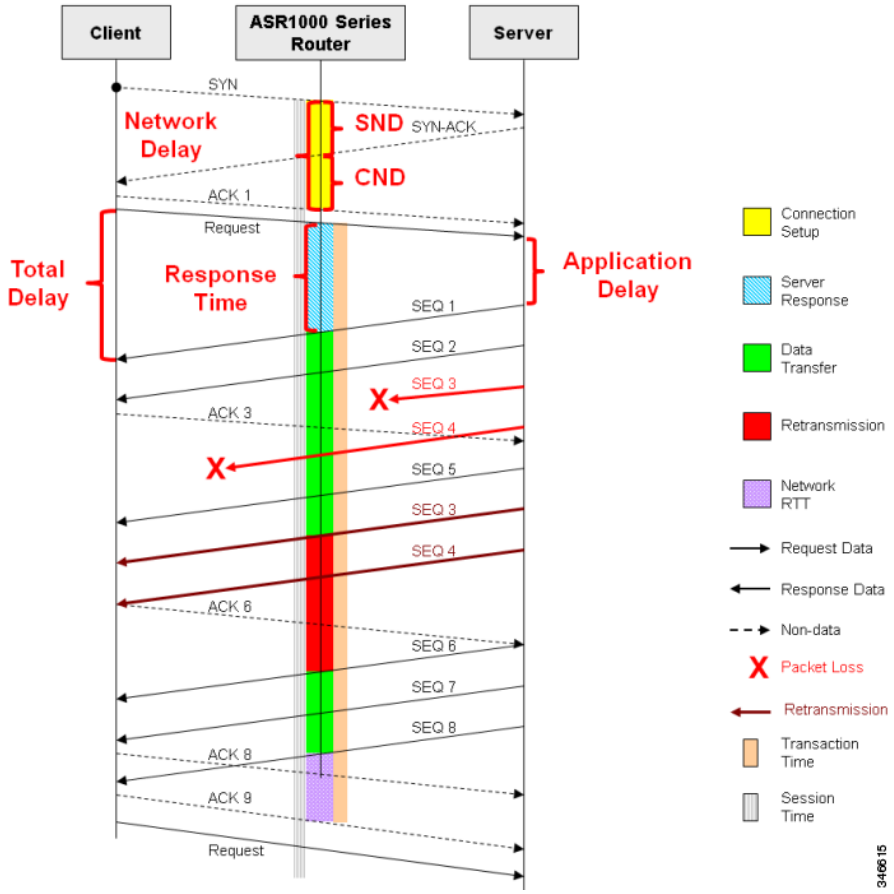


Figure 4-2 provides details of network response time metrics.

Figure 4-2 Network Response Time Metrics in Detail



Media Performance Metrics

The following are fields commonly used for media performance metrics by the Cisco AVC solution:

```
[collect | match] match transport rtp ssrc
collect  transport rtp payload-type
collect  transport rtp jitter mean sum
collect  transport rtp jitter [minimum | maximum]
collect  transport packets lost counter
collect  transport packets expected counter
collect  transport packets lost counter
collect  transport packets lost rate
collect  transport event packet-loss counter
collect  counter packets dropped
collect  application media bytes counter
collect  application media bytes rate
collect  application media packets counter
collect  application media packets rate
collect  application media event
collect  monitor event
```

Usage Guidelines

Some of the media performance fields require punt to the route processor (RP). For more information, see *Cisco Application Visibility and Control Field Definition Guide for Third-Party Customers*.

L2 Information

The following are L2 fields commonly used by the Cisco AVC solution:

```
[collect | match] datalink [source-vlan-id | destination-vlan-id]
[collect | match] datalink mac [source | destination] address [input | output]
```

WAAS Interoperability

The following are WAAS fields commonly used by the Cisco AVC solution:

```
[collect | match] services waas segment [account-on-resolution]
collect           services waas passthrough-reason
```

Usage Guidelines

Account-On-Resolution configures FNF to collect data in a temporary memory location until the record key fields are resolved. After resolution of the record key fields, FNF combines the temporary data collected with the standard FNF records. Use this option (**account-on-resolution**) when the field used as a key is not available at the time that FNF receives the first packet.

The following limitations apply when using Account-On-Resolution:

- Flows ended before resolution are not reported.
- FNF packet/octet counters, timestamp and TCP performance metrics are collected until resolution. All other field values are taken from the packet that provides resolution or the following packets.

Classification

The following are classification fields commonly used by the Cisco AVC solution:

```
[collect | match]  policy performance-monitor classification hierarchy
```

Usage Guidelines

Use this field to report the matched class for the performance-monitor policy-map.

NetFlow/IPFIX Option Templates

NetFlow option templates map IDs to string names and descriptions:

```
flow exporter my-exporter
  export-protocol ipfix
  template data timeout <timeout>
  option interface-table timeout <timeout>
  option vrf-table timeout <timeout>
  option sampler-table timeout <timeout>
  option application-table timeout <timeout>
  option application-attributes timeout <timeout>
  option sub-application-table timeout <timeout>
```

```
  option c3pl-class-table timeout <timeout>
  option c3pl-policy-table timeout <timeout>
```

NetFlow/IPFIX Show commands

Use the following commands to show or debug NetFlow/IPFIX information:

```
show flow monitor type performance-monitor [<name> [cache [raw]]]
show flow record type performance-monitor
show policy-map type performance-monitor [<name> | interface]
```

NBAR Attribute Customization

Use the following commands to customize the NBAR attributes:

```
[no] ip nbar attribute-map <profile name>
   attribute category <category>
   attribute sub-category <sub-category>
   attribute application-group <application-group>
   attribute tunnel <tunnel-info>
   attribute encrypted <encrypted-info>
   attribute p2p-technology <p2p-technology-info>
[no] ip nbar attribute-set <protocol-name> <profile name>
```

Note These commands support all attributes defined by the NBAR2 Protocol Pack, including custom-category, custom-sub-category, and custom-group available in Protocol Pack 3.1.

NBAR Customize Protocols

Use the following commands to customize NBAR protocols and assign a protocol ID. A protocol can be matched based on HTTP URL/Host or other parameters:

```
ip nbar custom <protocol-name> [http {[url <urlregexp>]  [host <hostregexp>]}] [offset
[format value]] [variable field-name field-length] [source | destination] [tcp | udp ]
[range start end | port-number ] [id <id>]
```

Packet Capture Configuration

Use the following commands to enable packet capture:

```
policy-map type packet-services <policy-name>
    class <class-name>
        capture limit packet-per-sec <pps> allow-nth-pak <np> duration <duration>
                   packets <packets>  packet-length <len>
        buffer size <size> type <type>

interface <interface-name>
   service-policy type packet-services <policy-name> [input|output]
```

QoS Metrics

This section describes how to configure Flexible NetFlow (FNF) monitors to include Quality of Service (QoS) metrics.

Background

FNF Monitors

Flexible NetFlow (FNF) enables monitoring traffic on router interfaces. FNF monitors are configured for a specific interface to monitor the traffic on that interface. At defined intervals, the monitor sends collected traffic data to a "collector," which can be a component within the router or an external component.

Beginning with Cisco AVC for IOS XE release 3.9, FNF records include new fields for QoS metrics.

QoS

QoS configuration is based on **class maps** and **policy maps**. Class maps categorize traffic; policy maps determine how to handle the traffic. Based on the policy identified for each packet, the packet is placed into a specific **QoS queue**, which determines the priority and pattern of transmission. Each queue is identified by a Queue ID field.

For additional information about QoS, visit: http://www.cisco.com/go/qos

Exported Metrics

AVC enables configuration of QoS Packet Drop and QoS Class Hierarchy monitors on an interface, using one or more of the following QoS metrics, which can be included in exported FNF records:

- Queue ID—Identifies a QoS queue.
- Queue Packet Drops—Packets dropped (on the monitored interface) per QoS queue, due to a QoS policy that limits resources available to a specific type of traffic.
- Class Hierarchy—Class hierarchy of the reported flow. The class hierarchy is determined by the QoS policy map and determines the traffic priority.

QoS Packet Drop Monitor Output in Exported Record

When a QoS Packet Drop monitor is configured, the FNF record includes packet drop data per QoS queue in the following format:

Queue id	Queue packet drops
1	100
2	20

QoS Class Hierarchy Information Included in Exported Record

QoS class hierarchy information is exported using the following FNF fields:

- Hierarchy policy for each flow (defined by the policy map)
- Queue ID for each flow

This section provides an example of a QoS policy map configuration, followed by the information provided in an FNF record for three flows governed by this configuration.

The example includes two levels of policy map hierarchy. In the example, the `service-policy P11` statement in **bold** type creates a hierarchy with the P11 policy map as a child of the P1 policy map.

Note QoS class hierarchy reporting supports a hierarchy of five levels.

Based on the configuration, the following applies to a packet with, for example, a DSCP value of "ef" in the IP header:

1. The C1 class definition includes the packet by the `match any` statement.
2. The C11 class definition includes the packet by the `match ip dscp ef` statement.
3. Because the packet is included in class C1, policy map P1 defines the policy for the packet with the `shaping average` statement.
4. Policy map P1 invokes policy map P11 for class C1 with the `service-policy P11` statement.
5. Because the packet is included in class C11, policy map P11 assigns the packet to a queue which has been allocated 10% of remaining bandwidth.

```
class-map match-all C1
    match any
class-map match-all C11
    match ip dscp ef
class-map match-all C12
    match ip dscp cs2
!
policy-map P11
    class C11
        bandwidth remaining percent 10
    class C12
        bandwidth remaining percent 70
    class class-default
        bandwidth remaining percent 20

policy-map P1
    class C1
        shaping average 16000000
        service-policy P11
```

Table 4-2 shows an example of the information provided in an FNF record for three flows governed by this configuration.

Table 4-2 QoS Class Hierarchy Information in the FNF record

Flow	Hierarchy	Queue id
Flow 1	P1, C1, C11	1
Flow 2	P1, C1, C11	1
Flow 3	P1, C1, C12	2

In Table 4-2, policy and class information is shown using the true policy and class names, such as P1 and C1. However, the FNF record exports policy and class names using numerical identifiers in place of policy and class names. The monitor periodically outputs a "policy option template" and a "class option template" indicating the policy names and class names that correspond to the numbers used in the

exported FNF records. These option templates are defined in the exporter configuration, using statements such as the following, which create the option templates and indicate the time interval at which the monitor outputs the option template information:

```
option c3pl-class-table timeout <timeout>
option c3pl-policy-table timeout <timeout>
```

Configuration

Enabling QoS Metric Collection

Enabling

To enable the QoS metrics collection feature for the platform, enter global configuration mode using `configure terminal`, then use the following QoS configuration command. The command causes QoS to begin collecting QoS metrics for FNF.

Note Enabling QoS metrics collection requires resetting all performance monitors on the device.

```
platform qos performance-monitor
```

Verifying

To verify that QoS metrics collection is enabled, use the following command:

```
show platform hardware qfp active feature qos config global
```

The following is an example of the output of the command:

```
Marker statistics are: disabled
Match per-filter statistics are: disabled
Match per-ace statistics are: disabled
Performance-Monitor statistics are: enabled
```

Configuring a QoS Packet Drop Monitor

A QoS Packet Drop monitor can only export the Queue ID and Queue Packet Drop fields. It cannot be combined with other monitors to export additional fields. At the given reporting interval, the monitor reports only on queues that have dropped packets (does not report value of 0).

Step 1: Create the QoS Packet Drop FNF Monitor

Use the following FNF configuration to create a QoS Packet Drop monitor. The process specifies a flow record of type "qos-record" and attaches the record to a monitor of type "qos-monitor." In the steps that follow, the qos-monitor is attached to the desired interface.

Note Ensure that QoS metrics collection is enabled. See Enabling QoS Metric Collection, page 4-12.

```
flow record qos-record
    match policy qos queue index
    collect policy qos queue drops
flow monitor qos-monitor
    exporter my-exporter
    record qos-record
```

Step 2: Configure the QoS Policy

The following example shows configuration of a QoS policy map. It includes a hierarchy of three policies: avc, avc-parent, and avc-gparent. Note that avc-gparent includes avc-parent, and avc-parent includes avc.

```
policy-map avc
    class prec4
        bandwidth remaining ratio 3
    class class-default
        bandwidth remaining ratio 1
policy-map avc-parent
    class class-default
        shape average 10000000
        service-policy avc
policy-map avc-gparent
    class class-default
        shape average 100000000
        service-policy avc-parent
```

Step 3: Attach the FNF Monitor and QoS Policy to an Interface

Use the following to attach the monitor to the desired interface. For *<interface>*, specify the interface type—for example: GigabitEthernet0/2/1

Specify the IP address of the interface in IPv4 or IPv6 format.

```
interface <interface>
    ip address <interface_IP_address>
    ip flow monitor qos-monitor output
    service-policy output avc-gparent
```

Verifying the QoS Packet Drop Monitor Configuration

This section provides commands that are useful for verifying or troubleshooting a QoS Packet Drop Monitor configuration.

Verifying that the Monitor is Allocated

Use the following command to verify that the QoS monitor exists:

```
show flow monitor
```

Use the following commands to verify additional monitor details:

```
show flow monitor qos-monitor
show flow monitor qos-monitor cache
show flow monitor qos-monitor statistics
show platform hardware qfp active feature fnf client flowdef name qos-record
show platform hardware qfp active feature fnf client monitor name qos-monitor
```

Verifying QoS queues and Class-Hierarchies

The following **show** commands display the statistics that QoS has collected. “gigX/X/X” refers to the interface for which the monitor has been configured.

```
show policy-map int gigX/X/X
show platform hardware qfp active feature qos queue output all
```

Verifying FNF-QOS FIA Activation

Use the following **show** command to verify that the FNF-QoS FIA (feature activation array) is enabled on the interface (GigabitEthernet0/2/1 in this example):

```
show platform hardware qfp active interface if-name GigabitEthernet0/2/1
```

Verifying the FNF Monitor and Record

Use the following **debug** commands to verify that the FNF monitor and record have been created:

```
debug platform software flow flow-def errors
debug platform software flow monitor errors
debug platform software flow interface errors


debug platform hardware qfp active feature fnf server trace
debug platform hardware qfp active feature fnf server info
debug platform hardware qfp active feature fnf server error
```

Configuring a QoS Class Hierarchy Monitor

In contrast to the QoS Packet Drop monitor, a QoS Class Hierarchy monitor can be combined with another monitor to export additional metrics.

Step 1: Create the QoS Class Record

The following example configuration creates a QoS class record. The process specifies a record of type "qos-class-record." The example specifies "ipv4 source" and "ipv4 destination" addresses, but you can configure the record to match according to other criteria.

Note Ensure that QoS metrics collection is enabled. See Enabling QoS Metric Collection, page 4-12.

```
flow record qos-class-record
    match ipv4 source address
    match ipv4 destination address
    collect counter bytes
    collect counter packets
    collect policy qos classification hierarchy
    collect policy qos queue index
```

Step 2: Create the QoS Class Hierarchy Monitor

Use the following FNF configuration to create a QoS Class Hierarchy monitor. The process specifies a monitor of type "class-hier-monitor." In the steps that follow, the monitor is attached to the desired interface.

```
flow monitor class-hier-monitor
    exporter my-exporter
    record qos-class-record
```

Step 3: Attach the QoS Class Hierarchy Monitor to an Interface

Use the following to attach the monitor to the desired interface. For <*interface*>, specify the interface type—for example: GigabitEthernet0/2/1

Specify the IP address of the interface in IPv4 or IPv6 format.

Note Attaching the service-policy to the interface, as indicated by the "service-policy" statement below, is a required step.

```
interface <interface>
    ip address <interface_IP_address>
    ip flow monitor class-hier-monitor output
    service-policy output avc-gparent
```

Verifying the QoS Class Hierarchy Monitor Configuration

This section provides commands that are useful for verifying or troubleshooting a QoS Class Hierarchy Monitor configuration.

Verifying that the Monitor is Allocated

Use the following command to verify that the QoS monitor exists:

```
show flow monitor
```

Use the following commands to verify additional details:

```
show flow monitor class-hier-monitor
show flow monitor class-hier-monitor cache
show flow monitor class-hier-monitor statistics

show platform hardware qfp active feature fnf client flowdef name qos-class-record
show platform hardware qfp active feature fnf client monitor name qos-monitor
```

Verifying FNF-QOS FIA Activation

In the following feature invocation array (FIA) verification example, the interface is GigabitEthernet0/2/1.

```
show platform hardware qfp active interface if-name GigabitEthernet0/2/1
```

Verifying the FNF Monitor and Record

Use the following **debug** commands to verify that the FNF monitor and record have been created:

```
debug platform software flow flow-def errors
debug platform software flow monitor errors
debug platform software flow interface errors

debug platform hardware qfp active feature fnf server trace
debug platform hardware qfp active feature fnf server info
debug platform hardware qfp active feature fnf server error
```

Connection/Transaction Metrics

Beginning with Cisco AVC for IOS XE release 3.9, Flexible NetFlow (FNF) monitors can report on individual transactions within a flow. This enables greater resolution for traffic metrics. This section describes how to configure connection and transaction metrics, including **transaction-id** and **connection id**, for FNF monitors. The connection/transaction monitoring feature is referred to as "Multi-transaction."

Note The Multi-transaction feature requires an NBAR protocol pack that supports the feature. The protocol pack provided with Cisco AVC for IOS XE release 3.9 and later protocol packs support this feature.

Introduction

Flexible NetFlow (FNF) monitors typically report traffic metrics per flow. (A flow is defined as a connection between a specific source address/port and destination address/port.) A single flow can include multiple HTTP transactions. Enabling the Multi Transaction feature for a monitor enables reporting metrics for each transaction individually.

You can configure the FNF record to identify the flow or the flow+transaction, using one of the following two metrics:

- connection id—A 4-byte metric identifying the flow.
- transaction-id—An 8-byte metric composed of two parts:
  - MSB—Identifies the flow and is equivalent to the connection id metric.
  - LSB—Identifies the transaction. The value is a sequential index of the transaction, beginning with 0.

Configuration

The following subsections describe the following for the Multi-transaction feature:

- Requirements, page 4-16
- Configuring Exporter, Record, and Monitor in Native FNF Mode, page 4-17
- Configuring Exporter, Record, and Monitor in Performance Monitor Mode, page 4-18
- Verifying and Troubleshooting the Configuration, page 4-18

Requirements

The following requirements apply when using the Multi-transaction feature:

- The record configuration must use **match**, not **collect**.
- Specify only "connection id" or "transaction-id," but not both.
- Include "application name" in the record.
- Include "cache timeout event transaction-end" which specifies that the record is transmitted immediately and not stored in the monitor cache.

Configuring Exporter, Record, and Monitor in Native FNF Mode

Use the following to configure exporter, record, and monitor.

The "match connection" statement shown in **bold** below must be one of the following:

- `match connection id`
- `match connection transaction-id`

```
flow exporter <exporter_name>
    destination <exporter_address>
    transport <exporter_port>

flow record <record_name>
    match connection <id or transaction-id>
    collect <specify_metric>
    collect application name

flow monitor <monitor_name>
    record  <record_name>
    exporter <exporter_name>
    cache timeout event transaction-end

interface <interface>
    ip flow monitor <monitor_name> input
```

Example

In the following example:

- Exporter: mul_trans_exporter
- Record: mul_trans_record_1
- Specifies the **transaction-id** metric for the FNF record, as shown in **bold**. Alternatively, you can specify the **connection id** metric.
- The `collect` statements specify the following metrics: `counter packets, application name`
- Monitor: mul_trans_monitor_1
- Interface: GigabitEthernet0/0/2

```
flow exporter mul_trans_exporter
    destination 64.128.128.128
    transport udp 2055

flow record mul_trans_record_1
    match connection transaction-id
    collect counter packets
    collect application name

flow monitor er_mul_trans_monitor_1
    record  mul_trans_record_1
    exporter mul_trans_exporter
    cache timeout event transaction-end

interface GigabitEthernet0/0/2
    ip flow monitor mul_trans_monitor_1 input
```

Configuring Exporter, Record, and Monitor in Performance Monitor Mode

Flexible Netflow (FNF) performance monitor (perf-monitor) mode enables configuring monitors with advanced filtering options that filter data before reporting it. Options for configuring filtering include IP access list, policy-map, and so on.

The following perf-monitor example configures a monitor and specifies the **transaction-id** metric for the FNF record, as shown in **bold**. Alternatively, you can specify the **connection id** metric.

Note See Configuring Exporter, Record, and Monitor in Native FNF Mode, page 4-17 for additional configuration information.

```
ip access-list extended mt_perf_acl
    permit ip any any

class-map match-all mt_perf_class
    match access-group name mt_perf_acl
    match protocol http

flow exporter mt_perf_exporter
    destination 64.128.128.128
    transport udp 2055

flow record type performance-monitor mt_perf_record
    match connection transaction-id
    collect counter packets
    collect application name
    collect application http url

flow monitor type performance-monitor mt_perf_monitor
    record mt_perf_record
    exporter mt_perf_exporter
    cache type normal
    cache timeout event transaction-end

policy-map type performance-monitor mt_perf_policy
    parameter default account-on-resolution
    class mt_perf_class
    flow monitor mt_perf_monitor

interface GigabitEthernet0/0/2
    service-policy type performance-monitor input mt_perf_policy
```

Verifying and Troubleshooting the Configuration

This section describes commands useful for verification and troubleshooting the FNF configuration. There are subsections for:

- Native or Performance Monitor Mode, page 4-19
- Native FNF Mode, page 4-19
- Performance Monitor Mode, page 4-19

Note For information about the **show** commands in the sections below, see the FNF command reference guide: http://www.cisco.com/en/US/docs/ios-xml/ios/fnetflow/command/fnf-cr-book.html

Native or Performance Monitor Mode

Verifying Multi-transaction Status

Display the Multi-transaction status:

```
show plat soft nbar statistics | inc is_multi_trs_enable
```

If Multi-transaction is enabled, the value is: `is_multi_trs_enable==1`

Native FNF Mode

Validating the Configuration

Use the following **show** commands to validate the configuration.

```
show flow exporter <exporter_name> templates
show flow monitor <monitor_name>
show platform hardware qfp active feature fnf client flowdef name <record_name>
show platform hardware qfp active feature fnf client monitor name <monitor_name>
```

Viewing Collected FNF Data and Statistics

Use the following **show** commands to view the collected FNF data and statistics.

```
show flow monitor <monitor_name> cache
show flow monitor <monitor_name> statistics
show flow exporter <exporter_name> statistics
show platform hardware qfp active feature fnf datapath aor
```

Performance Monitor Mode

Validating the Configuration

Use the following **show** commands to validate the configuration.

```
show flow exporter <exporter_name> templates
show flow record type performance-monitor <record_name>
show platform hardware qfp active feature fnf client monitor name <monitor_name>
```

Viewing Collected FNF Data and Statistics

Use the following **show** commands to view the FNF collected data and statistics.

```
show performance monitor cache monitor <monitor_name> detail
show flow exporter <exporter_name> statistics
show platform hardware qfp active feature fnf datapath aor
```

Easy Performance Monitor

Overview

The Easy Performance Monitor (“Easy perf-mon” or “ezPM”) feature provides an “express” method of provisioning monitors. This new mechanism adds functionality and does not affect the existing methods for provisioning monitors.

Easy perf-mon does not provide the full flexibility of the traditional perf-mon configuration model. Easy perf-mon provides “profiles” that represent typical deployment scenarios. After selecting a profile and specifying a small number of parameters, Easy perf-mon provides the remaining provisioning details.

For additional information about configuring Easy perf-mon, see:
Easy Performance Monitor

Application Experience Profile

For Cisco IOS XE release 3.10, Easy perf-mon includes one profile, called "Application Experience," and five different traffic monitors, described in Table 4-3. Future releases will provide additional options.

Table 4-3 Application Experience Traffic Monitors

	Monitor Name	Default Traffic Classification
1	Application-Response-Time (ART)	All TCP
2	URL	HTTP applications
3	Media	RTP applications over UDP
4	Conversation-Traffic-Stats	Remaining traffic not matching other classifications
5	Application-Traffic-Stats	DNS and DHT

Users can override a small set of parameters in each of the traffic monitors, as described in Table 4-4. For an example of how to configure parameters in the Application Experience profile, see Easy Performance Monitor—Application Experience Profile Configuration, page 4-22.

Table 4-4 Application Experience Traffic Monitors: Parameters

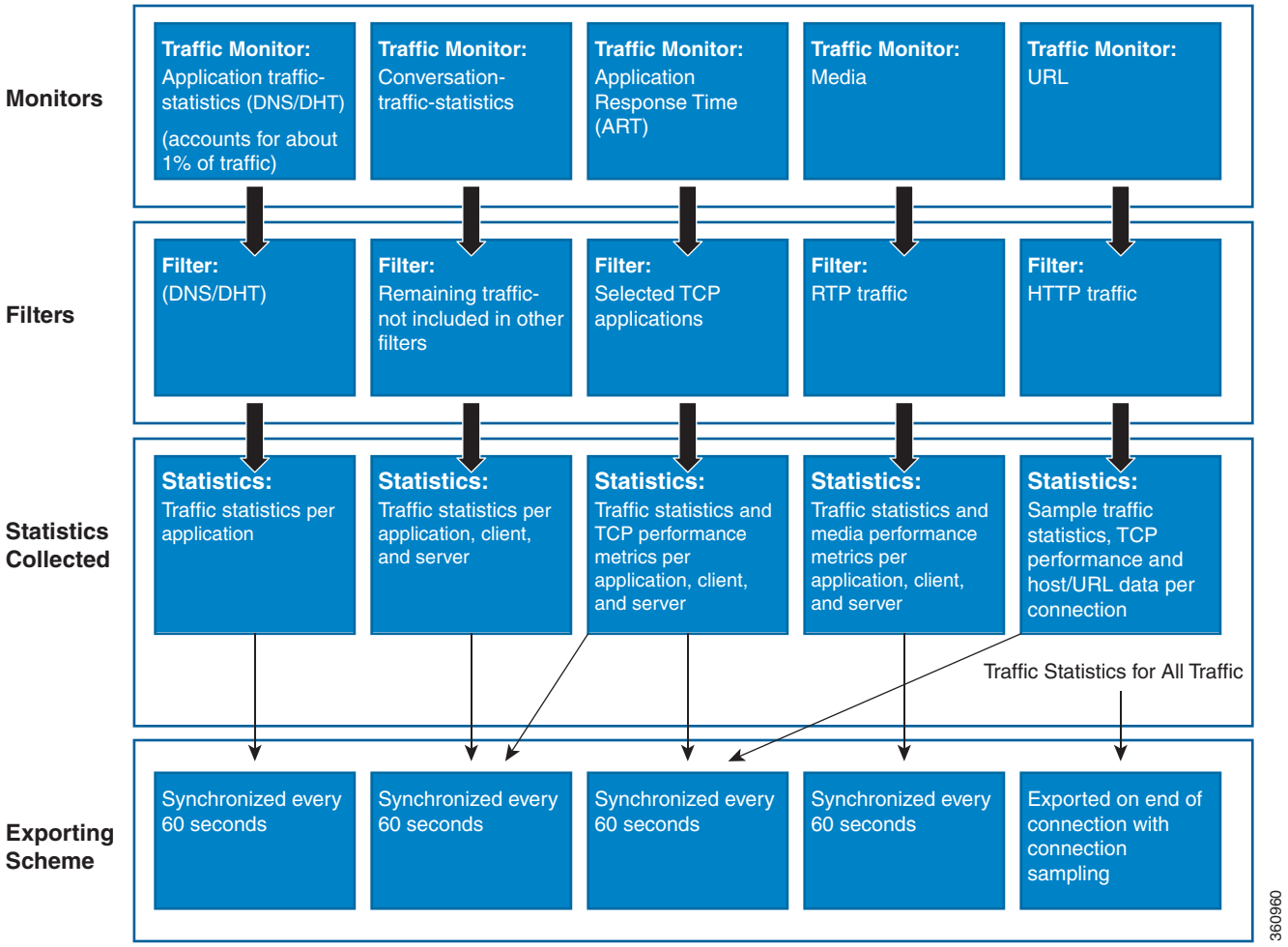
		Configurable Parameters			
	Monitor Name	IPv4 / IPv6	In / Out	Traffic Class	Cache Size
1	Application-Response-Time (ART)	Y	N	Class and Application only	Y
2	URL	Y	N	Class and Application only	Y
3	Media	Y	Y	Class and Application only	Y
4	Conversation-Traffic-Stats	Y	N	N	Y
5	Application-Traffic-Stats	N	N	N	Y

The Application Experience profile implements the improved data exporting model, which is optimized for maximum performance, exporting the maximum possible amount of available information for monitored traffic. Based on the requirements of the reports that have been defined:

- For each type of traffic, the exported record contains all of the collected data required for the defined reports, with the required granularity.
- Exported records do not contain unnecessary data, such as data redundant with previously exported records or data that is not required for the defined reports.
- Exported records include server information.

Figure 4-3 illustrates how the "Application Experience" profile exports different types of traffic statistics.

Figure 4-3 Export Model—Application Experience Profile



CLI Field Aliases

AVC for Cisco IOS XE release 3.10 introduces aliases to the monitor configuration syntax. Using the **all** alias simplifies configuration statements. The **all** refers to the set of all fields possible for a given statement. For example, "`collect connection delay` **`all`**" configures all fields that are possible to configure by the "`collect connection delay`" statement.

The following are examples:

```
collect connection delay all
collect connection transaction all
collect connection client all
collect connection server all
collect connection delay response to-server histogram all
```

Caution When using aliases, see Removing Aliases before Downgrading from Cisco IOS XE 3.10, page 6-2 before downgrading from Cisco IOS XE release 3.10 or later.

Configuration Examples

This section contains AVC configuration examples. These examples provide a general view of a variety of configuration scenarios. Configuration is flexible and supports different types of record configurations.

Easy Performance Monitor Configuration

The following Easy Performance Monitor (Easy perf-mon) configuration example activates all traffic monitors in the profile and attaches the policy-maps, both ingress and egress, to the GigabitEthernet0/0/1 interface:

```
!
! Easy performance monitor context
! --------------------------------
!
performance monitor context my-avc profile application-experience
   exporter destination 1.2.3.4 source GigabitEthernet0/0/1 port 4739
   traffic-monitor all
!
!
! Interface attachments
! ---------------------
interface GigabitEthernet0/0/1
  performance monitor context my-avc
```

Easy Performance Monitor—Application Experience Profile Configuration

The following Easy Performance Monitor (Easy perf-mon) “Application Experience” profile configuration example activates three traffic monitors, and specifies monitoring only IPv4 traffic. The context is then attached to two interfaces:

```
!
! Easy performance monitor context
! --------------------------------
!
performance monitor context my-visibility profile application-experience
   exporter destination 1.2.3.4 source GigabitEthernet0/0/1 port 4739

   traffic-monitor application-response-time ipv4
   traffic-monitor conversation-traffic-stats ipv4
   traffic-monitor media ipv4
!
! Interface attachments
! ---------------------
interface GigabitEthernet0/0/1
  performance monitor context my-visibility
interface GigabitEthernet0/0/2
  performance monitor context my-visibility
```

Conversation Based Records—Omitting the Source Port

The monitor configured in the following example sends traffic reports based on conversation aggregation. For performance and scale reasons, it is preferable to send TCP performance metrics only for traffic that requires TCP performance measurements. It is recommended to configure two similar monitors:

- One monitor includes the required TCP performance metrics. In place of the line shown in **bold** in the example below (collect <any TCP performance metric>), include a line for each TCP metric for the monitor to collect.
- One monitor does not include TCP performance metrics.

The configuration is for IPv4 traffic. Similar monitors should be configured for IPv6.

```
flow record type performance-monitor conversation-record
    match services waas segment account-on-resolution
    match connection client ipv4 (or ipv6) address
    match connection server ipv4 (or ipv6) address
    match connection server transport port
    match ipv4 (or ipv6) protocol
    match application name account-on-resolution
    collect interface input
    collect interface output
    collect connection server counter bytes long
    collect connection client counter bytes long
    collect connection server counter packets long
    collect connection client counter packets long
    collect connection sum-duration
    collect connection new-connections
    collect policy qos class hierarchy
    collect policy qos queue id
    collect <any TCP performance metric>


flow monitor type performance-monitor conversation-monitor
    record conversation-record
    exporter my-exporter
    history size 0
    cache type synchronized
    cache timeout synchronized 60
    cache entries <cache size>
```

HTTP URL

The monitor configured in the following example sends the HTTP host and URL. If the URL is not required, the host can be sent as part of the conversation record (see Conversation Based Records—Omitting the Source Port, page 4-23).

```
flow record type performance-monitor url-record
    match transaction-id
    collect application name
    collect connection client ipv4 (or ipv6) address
    collect routing vrf input
    collect application http url
    collect application http host
    <other metrics could be added here if needed.
     For example bytes/packets to calculate BW per URL
     Or performance metrics per URL>
```

```
flow monitor type url-monitor
    record url-record
    exporter my-exporter
    history size 0
    cache type normal
    cache timeout event transaction-end
    cache entries <cache size>
```

HTTP URI

The **uri statistics** command enables exporting the first level of a parsed URI address. The command exports the value in the URI statistics field, which contains the depth 1 URI value, followed by a URI hit count value of 1.

Note For Cisco ASR 1000 Series platforms, the URI hit count value is always 1 because the URI statistics field can only be configured per connection or transaction.

If no backslash exists at all after the URL, a zero length field is exported.

If the depth 1 value of the parsed URI exceeds a maximum number of characters, the value is truncated to the maximum length.

Note This command must be configured with either the **connection id** or **transaction-id** commands.

Configuration Example

```
flow record er_uri_stat_record_1
    match connection transaction-id
    collect application name
    collect counter packets
    collect application http uri statistics
```

Example of Exported Value—Typical Address

Address: http://usr:pwd@www.test.com:81/dir/dir.2/index.htm?q1=0&&test1&test2=value#top

The **uri statistics** command exports: **/dir:1**

- **/dir** is the URI depth 1 level value.
- The "**:**" indicates a null character, followed by a URI hit count value of **1**.

Example of Exported Value—No Backslash after URL

Address: http://usr:pwd@www.test.com

The **uri statistics** command exports a zero length field.

Application Traffic Statistics

The monitor configured in the following example collects application traffic statistics:

```
flow record type performance-monitor application-traffic-stats
    match ipv4 protocol
    match application name account-on-resolution
    match ipv4 version
    match flow direction
```

```
    collect connection initiator
    collect counter packets
    collect counter bytes long
    collect connection new-connections
    collect connection sum-duration

flow monitor type application-traffic-stats
    record application-traffic-stats
    exporter my-exporter
    history size 0
    cache type synchronized
    cache timeout synchronized 60
    cache entries <cache size>
```

Media RTP Report

The monitor configured in the following example reports on media traffic:

```
flow record type performance-monitor media-record
    match ipv4(or ipv6) protocol
    match ipv4(or ipv6) source address
    match ipv4(or ipv6) destination address
    match transport source-port
    match transport destination-port
    match transport rtp ssrc
    match routing vrf input
    collect transport rtp payload-type
    collect application name
    collect counter packets long
    collect counter bytes long
    collect transport rtp jitter mean sum
    collect transport rtp payload-type
    collect <other media metrics>

flow monitor type media-monitor
    record media-record
    exporter my-exporter
    history size 10 // default history
    cache type synchronized
    cache timeout synchronized 60
    cache entries <cache size>
```

QoS Example 1: Control and Throttle Traffic

The following QoS configuration example illustrates how to control and throttle the peer-to-peer (P2P) traffic in the network to 1 megabit per second:

```
class-map match-all p2p-class-map
  match protocol attribute sub-category p2p-file-transfer

policy-map p2p-attribute-policy
  class p2p-class-map
    police 1000000
interface Gig0/0/3
    service-policy input p2p-attribute- policy
```

QoS Example 2: Assigning Priority and Allocating Bandwidth

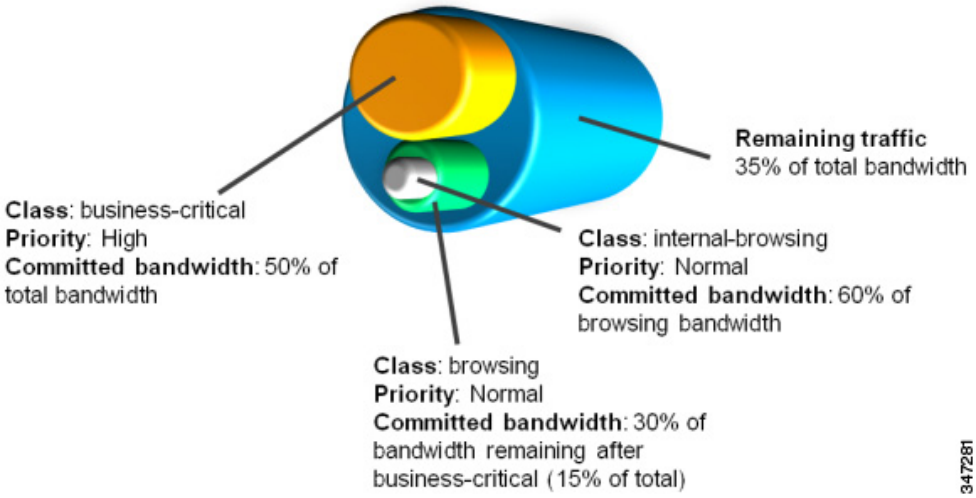
The following QoS configuration example illustrates how to allocate available bandwidth on the eth0/0 interface to different types of traffic. The allocations are as follows:

- Business-critical Citrix application traffic for "access-group 101" users receives highest priority, with 50% of available bandwidth committed and traffic assigned to a priority queue. The `police` statement limits the bandwidth of business-critical traffic to 50% in the example.
- Web browsing receives a committed 30% of the remaining bandwidth after the business-critical traffic. This is a commitment of 15% of the total bandwidth available on the interface.
- Internal browsing, as defined by a specific domain (myserver.com in the example), receives a committed 60% of the browsing bandwidth.
- All remaining traffic uses the remaining 35% of the total bandwidth.

The policy statements commit minimum bandwidth in the percentages described for situations of congestion. When bandwidth is available, traffic can receive more than the "committed" amount. For example, if there is no business-critical traffic at a given time, more bandwidth is available to browsing and other traffic.

Figure 4-4 illustrates the priority and bandwidth allocation for each class. "Remaining traffic" refers to all traffic not specifically defined by the class mapping.

Figure 4-4 Bandwidth allocation



In class-map definition statements:

- **match-all** restricts the definition to traffic meeting all of the “match” conditions that follow. For example, the “business-critical” class only includes Citrix protocol traffic from IP addresses in “access-group 101.”
- **match-any** includes traffic meeting one or more of the “match” conditions that follow.

```
class-map match-all business-critical
    match protocol citrix
    match access-group 101
class-map match-any browsing
    match protocol attribute category browsing

class-map match-any internal-browsing
    match protocol http url “*myserver.com*”

policy-map internal-browsing-policy
    class internal-browsing
        bandwidth remaining percent 60

policy-map my-network-policy
    class business-critical
        priority
        police cir percent 50
    class browsing
        bandwidth remaining percent 30
        service-policy internal-browsing-policy

interface eth0/0
    service-policy output my-network-policy
```