Configuring Packet Capture

Prerequisites for Configuring Packet Capture

Packet capture is supported on Cisco Catalyst 9600 Series Switches

The following sections provide information about the prerequisites for configuring packet capture.

Prerequisites for Configuring Wireshark

- Wireshark is supported only on switches running DNA Advantage
- Before starting a Wireshark capture process, ensure that CPU usage is moderate and that sufficient memory (at least 200 MB) is available. The CPU usage during Wireshark capture depends on how many packets match the specified conditions. It also depends on the intended actions for the matched packets (store, decode and display, or both).

Prerequisites for Configuring Embedded Packet Capture

The Embedded Packet Capture (EPC) software subsystem consumes CPU and memory resources during its operation. You must have adequate system resources for different types of operations. The following table provides some guidelines for using the system resources.

Table 1: System Requirements for the EPC Subsystem

System Resources	Requirements
Hardware	CPU utilization requirements are platform-dependent.

System Resources	Requirements
Memory	The DRAM stores the packet buffer. The size of the packet buffer is user specified.
Disk space	Packets can be exported to external devices. No intermediate storage on flash disk is required.

Restrictions for Configuring Packet Capture

The following sections provide information about the restrictions for configuring packet capture.

Restrictions for Configuring Wireshark

- Wireshark doesn't support Global packet capture.
- Wireshark doesn't support limiting circular file storage by file size.
- If you delete the file used by an active capture session, the capture session can't create a new file. All further packets captured are lost. You need to restart the capture point.
- File limit is limited to the size of the flash in DNA Advantage.
- Wireshark can't capture packets on a destination SPAN port.
- Wireshark stops capturing when one of the attachment points (interfaces) attached to a capture point stops working. For example, if the device that is associated with an attachment point is unplugged from the device. To resume capturing, restart the capture manually.
- Wireshark doesn't capture packets dropped by floodblock.
- A Wireshark class map allows only one ACL (IPv4, IPv6, or MAC).
- ACL logging and Wireshark are incompatible. Once you activate Wireshark, it takes priority. All traffic, including that captured by ACL logging on any ports, is redirected to Wireshark. We recommended that you deactivate ACL logging before starting Wireshark. Otherwise, Wireshark traffic will be contaminated by ACL logging traffic.
- If you capture both PACL and RACL on the same port, only one copy is sent to the CPU. If you capture a DTLS-encrypted CAPWAP interface, two copies are sent to Wireshark, one encrypted and the other decrypted. The same behavior occurs if we capture a Layer 2 interface carrying DTLS-encrypted CAPWAP traffic. The core filter is based on the outer CAPWAP header.
- The CLI for configuring Wireshark requires that the feature is executed only from EXEC mode. Actions that usually occur in configuration submode (such as defining capture points), are handled at the EXEC mode instead. All key commands are not NVGEN’d and aren’t synchronized to the standby Supervisor in NSF and SSO scenarios.

Embedded Wireshark is supported with the following limitations:

- Capture filters and display filters aren't supported.
- Active capture decoding isn’t available.

- The output format is different from previous releases.
- A Wireshark session with either a longer duration limit or no capture duration (using a terminal with no auto-more support using the **term len 0** command) may make the console or terminal unusable.
- Packet length range as a filter for packet capture isn't supported for non- IPv4/IPv6 packets and fragmented packets.
- Packet length range as a filter can't be used in addition with any other filters.

Restrictions for Configuring Embedded Packet Capture

The following restrictions are applicable to all models of the Cisco Catalyst 9600 Series Switches:

- Layer 2 EtherChannels aren't supported.
- You can't use VRFs, management ports, and private VLANs as attachment points.
- Embedded Packet Capture (EPC) isn't supported on logical ports, which includes port channels, switch virtual interfaces (SVIs), and subinterfaces. It's supported only on physical ports.
- A VLAN interface that is in shutdown state doesn't support EPC.
- If you change interface from switch port to routed port (Layer 2 to Layer 3) or the opposite way, you must delete the capture point and create a new one, once the interface comes back up. Stop/start the capture point won't work.
- Packets captured in the output direction of an interface might not reflect the changes made by the device rewrite. This includes TTL, VLAN tag, CoS, checksum, MAC addresses, DSCP, precedent, UP, etc.
- Even though the minimum configurable duration for packet capture is 1 second, packet capture works for a minimum of 2 seconds.
- It's not possible to modify a capture point parameter when a capture is already active or has started.
- EPC captures multicast packets only on ingress and doesn't capture the replicated packets on egress.
- The Rewrite information of both ingress and egress packets aren't captured.
- CPU-injected packets are considered control plane packets. Therefore, these types of packets won't be captured on an interface egress capture.
- Control plane packets aren't rate limited and performance impacting. Use filters to limit control plane packet capture.
- DNA Advantage supports decoding of protocols such as Control and Provisioning of Wireless Access Points (CAPWAP).
- You can define up to eight capture points, but only one can be active at a time. Stop one before you can start the other.
- MAC filter won't capture IP packets even if it matches the MAC address. This applies to all interfaces (Layer 2 switch port, Layer 3 routed port).
- MAC ACL is only used for non-IP packets such as ARP. It won't be supported on a Layer 3 port or SVI.
- MAC filter can't capture Layer 2 packets (ARP) on Layer 3 interfaces.

- VACL doesn't support IPv6-based ACLs.
- EPC cannot capture based on the underlying routing protocols in MPLS packets.

The following restrictions are applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2):

- EPC is not supported on Layer 2 ports.
- EPC is not suppored on port channel member interface.
- EPC is not supported on the control-plane interface.
- EPC is not supported with Ethernet-over-MPLS (EoMPLS).
- Hardware drop statistics are not available as no CPU queue is allocated for ACL logging.
- The following behavior is applicable for ACL logging with EPC on ingress and egress:
  - On ingress, if ACL logging, QoS ACL, and EPC ACL are configured at the same time, EPC ACL will not take priority.

    The following behaviors are applicable if both ACL logging and EPC ACLs are enabled:
    - ACL with deny logging: If a packet matches both EPC ACL and ACL with deny logging, then two copies will be sent to the CPU, one from EPC and one from ACL log.
    - ACL with permit logging: If a packet matches both EPC ACL and ACL with permit logging, only the copy from ACL log will be sent to the CPU.
  - On egress, if both EPC and ACL logging are configured, irrespective of the order in which they are configured, ACL logging will take priority and EPC will not work.

Information About Packet Capture

The Packet Capture feature is an onboard packet capture facility that allows network administrators to capture packets flowing to, through, and from the device. You can analyze them locally or save and export them for offline analysis by using tools such as Wireshark and Embedded Packet Capture (EPC). This feature simplifies network operations by allowing devices to become active participants in the management and operation of the network. This feature facilitates troubleshooting by gathering information about the packet format. This feature also facilitates application analysis and security.

Embedded Packet Capture with Wireshark is supported on DNA Advantage.

About Wireshark

Wireshark is a packet analyzer program that supports multiple protocols and presents information in a text-based user interface.

Wireshark dumps packets to a file using a well-known format called .pcap, and is applied or enabled on individual interfaces. You specify an interface in EXEC mode along with the filter and other parameters. The Wireshark application is applied only when you enter a **start** command, and is removed only when Wireshark stops capturing packets either automatically or manually.

Capture Points

A capture point is the central policy definition of the Wireshark feature. The capture point describes all the characteristics associated with a given instance of Wireshark. These include which packets to capture, where to capture them from, what to do with the captured packets, and when to stop. Capture points can be modified after creation, and don't become active until explicitly activated with a **start** command. This process is termed activating the capture point or starting the capture point. Capture points are identified by name and can also be manually or automatically deactivated or stopped.

Multiple capture points can be defined.

In case of stacked systems, the capture point is activated on the active member. A switchover terminates any active packet capture session and you have to restart the session.

Attachment Points

An attachment point is a point in the logical packet process path associated with a capture point. An attachment point is an attribute of the capture point. Packets that impact an attachment point are tested against capture point filters; packets that match are copied and sent to the associated Wireshark instance of the capture point. A specific capture point can be associated with multiple attachment points, with limits on mixing attachment points of different types. Some restrictions apply when you specify attachment points of different types. Attachment points are directional (input or output or both) with the exception of the Layer 2 VLAN attachment point, which is always bidirectional.

In case of stacked systems, the attachment points on all stack members are valid. EPC captures the packets from all the defined attachment points. However these packets are processed only on the active member.

Filters

Filters are attributes of a capture point that identify and limit the subset of traffic traveling through the attachment point of a capture point. These are copied and passed to Wireshark. Wireshark displays a packet, if it passes through an attachment point, and all the filters associated with the capture point.

A capture point has the following types of filters:

- Core system filter—The core system filter is applied by hardware, and its match criteria is limited by hardware. This filter determines whether to copy the hardware-forwarded traffic to software for Wireshark purposes.
- Capture filter—Wireshark applies the capture filter. The match criteria are more granular than those supported by the core system filter. Packets that pass the core filter but fail the capture filter are copied. They are sent to the CPU/software, but are discarded by the Wireshark process. The capture filter syntax matches that of the display filter.

Note Wireshark on the doesn't use the syntax of the capture filter.

- Display filter—Wireshark applies the display filter. Its match criteria are similar to the criteria of the capture filter. Packets that fail the display filter aren’t displayed.

Core System Filter

You can specify core system filter match criteria by using the class map or ACL, or explicitly by using the CLI.

In some installations, you need to obtain authorization to modify the device configuration, which can lead to extended delays if the approval process is lengthy. This can limit the ability of network administrators to monitor and analyze traffic. To address this situation, Wireshark supports explicit specification of core system filter match criteria from the EXEC mode CLI. The disadvantage is that the match criteria that you can specify is a limited subset of what class map supports, such as MAC, IP source and destination addresses, ether-type, IP protocol, and TCP/UDP source and destination ports.

If you prefer to use configuration mode, you can define ACLs or have class maps refer capture points to them. Explicit and ACL-based match criteria are used internally to construct class maps and policy maps.

Note that ACL and class map configuration are part of the system and not aspects of the Wireshark feature.

Display Filter

With the display filter, you can direct Wireshark to further narrow the set of packets to display when decoding and displaying from a .pcap file.

Actions

You can invoke Wireshark on live traffic or on a previously existing .pcap file. When invoked on live traffic, it can perform four types of actions on packets that pass its display filters:

- Captures to buffer in memory to decode and analyze and store.
- Stores to a .pcap file
- Decodes and displays
- Stores and displays

The decode and display action is applicable only when invoked on a .pcap file.

Storage of Captured Packets to Buffer in Memory

You can store packets in the capture buffer in memory. You can use the packets for subsequent decoding, analysis, or storage to a .pcap file.

The capture buffer can be in linear or circular mode. In linear mode, when the buffer is full it discards new packets. In circular mode, if the buffer is full, it discards the older packets to accommodate the new packets. Although you can clear the buffer when needed, this mode is used for debugging network traffic. However, it's not possible to clear the contents of the buffer alone without deleting it. Stop the current captures and restart the capture again for this to take effect.

Note If you have more than one capture that is storing packets in a buffer, clear the buffer before starting a new capture to avoid memory loss.

Storage of Captured Packets to a .pcap File

Note When you use Wireshark on switches in a stack, you can store packet captures only on flash or USB flash devices connected to the active switch.

For example, if flash1 is connected to the active switch, and flash2 is connected to the secondary switch, only flash1 can be used to store packet captures.

Attempts to store packet captures on devices other than flash or USB flash devices connected to the active switch result in errors.

Wireshark can store captured packets to a .pcap file. You can locate the capture file on the following storage devices:

- Device on-board flash storage (flash:)
- USB drive

Note Attempts to store packet captures on unsupported devices or devices not connected to the active switch result in errors.

When configuring a Wireshark capture point, you can associate a filename. When you activate the capture point, Wireshark creates a file with the specified name and writes packets to it. If the file exists at the time of creation of the capture point, Wireshark asks you if the file can be overwritten. If the file exists at the time of activating the capture point, Wireshark will overwrite the existing file. You can assoicate only one capture point with a given filename.

If the destination of the Wireshark writing process is full, Wireshark fails with partial data in the file. Ensure that there's sufficient space in the file system before you start the capture session.

You can reduce the required storage space by retaining only a segment, instead of the entire packet. Typically, you don't require details beyond the first 64 bytes or 128 bytes. The default behavior is to store the entire packet.

To avoid possible packet drops when processing and writing to the file system, Wireshark can optionally use a memory buffer to temporarily hold packets as they arrive. You can sepcify the memory buffer size when the capture point is associated with a .pcap file.

Packet Decoding and Display

Wireshark can decode and display packets to the console. This functionality is possible for capture points applied to live traffic and for capture points applied to a previously existing .pcap file.

Note Decoding and displaying packets may be CPU intensive.

Wireshark can decode and display packet details for a wide variety of packet formats. The details are displayed by entering the **monitor capture name start** command with one of the following keyword options, which place you into a display and decode mode:

- Brief—Displays one line per packet (the default).
- Detailed—Decodes and displays all the fields of all the packets whose protocols are supported. Detailed modes require more CPU than the other two modes.
- (hexadecimal) dump—Displays one line per packet as a hexadecimal dump of the packet data and the printable characters of each packet.

When you enter the **capture** command with the decode and display option, the Wireshark output is returned to Cisco IOS and displayed on the console unchanged.

Live Traffic Display

Wireshark receives copies of packets from the core system. Wireshark applies its display filters to discard uninteresting packets, and then decodes and displays the remaining packets.

.pcap File Display

Wireshark can decode and display packets from a previously stored .pcap file and direct the display filter to selectively displayed packets.

Packet Storage and Display

Functionally, this mode is a combination of the previous two modes. Wireshark stores packets in the specified .pcap file and decodes and displays them to the console. Only the core filters are applicable here.

Wireshark Capture Point Activation and Deactivation

After you define a Wireshark capture point with its attachment points, filters, actions, and other options, you must activate it. Until you activate the capture point, it doesn't actually capture packets.

Before you activate a capture point, some functional checks are performed. A capture point can't be activated if a core system filter or attachment points aren't defined. Attempting to activate a capture point that doesn't meet these requirements generates an error.

The display filters are specified as needed.

After you activate Wireshark capture points, you can deactivate them in multiple ways. You can halt a capture point that is storing only packets to a .pcap file manually. You can also configure it with time or packet limits, after which the capture point halts automatically.

When you activate a Wireshark capture point, a fixed rate policer is applied automatically in the hardware. This ensures the CPU isn't flooded with Wireshark directed packets. The disadvantage of the rate policer is that you can't capture contiguous packets beyond the established rate even if more resources are available.

The set packet capture rate is 1000 packets per sec (pps). The 1000 pps limit is applied to the sum of all attachment points. For example, if we have a capture session with three attachment points, the rates of all three attachment points together is policed to 1000 pps.

Note Policer isn't supported for control-plane packet capture. When activating control-plane capture points, you need to be extra cautious, so that it does not flood the CPU.

Wireshark Features

This section describes how Wireshark features function in the device environment:

- Redirection features—In the input direction, features traffic redirected by Layer 3 (such as PBR and WCCP) are logically later than Layer 3 Wireshark attachment points. Wireshark captures these packets even though they might later be redirected out another Layer 3 interface. Symmetrically, output features redirected by Layer 3 (such as egress WCCP) are logically prior to Layer 3 Wireshark attachment points, and Wireshark won't capture them.
- SPAN—Wireshark can't capture packets on interface configured as a SPAN destination.
- SPAN—Wireshark is able to capture packets on interfaces configured as a SPAN source in the ingress direction, and may be available for egress direction too.

Guidelines for Configuring Wireshark

- During Wireshark packet capture, hardware forwarding happens concurrently.
- Because packet forwarding typically occurs in hardware, packets aren't copied to the CPU for software processing. For Wireshark packet capture, packets are copied and delivered to the CPU, which causes an increase in CPU usage.
- You might experience high CPU (or memory) usage if:
  - You leave a capture session enabled and unattended for a long period, resulting in unanticipated bursts of traffic.
  - You launch a capture session with ring files or capture buffer and leave it unattended for a long time, resulting in performance or system health issues.
- To avoid high CPU usage, do the following:
  - Attach only relevant ports.
  - Use a class map, and secondarily, an access list to express match conditions. If neither is viable, use an explicit, in-line filter.
  - Adhere closely to the filter rules. Restrict the traffic type (such as, IPv4 only) with a restrictive, rather than relaxed ACL, which elicits unwanted traffic.
  - When using Wireshark to capture live traffic, consider applying a QoS policy temporarily to limit the actual traffic until the capture process concludes.
- Always limit packet capture to either a shorter duration or a smaller packet number. The parameters of the capture command enable you to specify the following:
  - Capture duration
  - Number of packets captured
  - File size
  - Packet segment size
- During a capture session, watch for high CPU usage and memory consumption due to Wireshark that may impact device performance or health. If these situations arise, stop the Wireshark session immediately.

- Run a capture session without limits if you know that little traffic matches the core filter.
- Writing to flash disk is a CPU-intensive operation. If the capture rate is insufficient, you may want to use a buffer capture.
- Avoid decoding and displaying packets from a .pcap file for a large file. Instead, transfer the .pcap file to a PC and run Wireshark on the PC.
- If you plan to store packets to a storage file, ensure that sufficient space is available before beginning a Wireshark capture process.
- To avoid packet loss, consider the following:
  - Use store-only (when you don't specify the display option) while capturing live packets. Do not use it for decode and display, which is an CPU-intensive operation (especially in detailed mode).
  - If you have more than one capture that is storing packets in a buffer, clear the buffer before starting a new capture to avoid memory loss.
- If you want to decode and display live packets in the console window, ensure that you bound the Wireshark session by a short capture duration.
- The core filter can be an explicit filter, access list, or class map. Specifying a newer filter of these types replaces the existing one.
- No specific order applies when defining a capture point. You can define capture point parameters in any order, provided that CLI allows this. The Wireshark CLI allows as many parameters as possible on a single line. This limits the number of commands required to define a capture point.
- All parameters except attachment points take a single value. Generally, you can replace the value with a new one by reentering the command. After user confirmation, the system accepts the new value and overrides the older one. A **no** form of the command is unnecessary to provide a new value, but it's necessary to remove a parameter.
- The action you want to perform determines which parameters are mandatory. The Wireshark CLI allows you to specify or modify any parameter prior to entering the **start** command. When you enter the **start** command, Wireshark will start only after determining that all mandatory parameters have been provided.
- If the file exists at the time of creation of the capture point, Wireshark aks you if the file can be overwritten. If the file exists at the time of activating the capture point, Wireshark overwrites the existing file.
- You can terminate a Wireshark session with an explicit **stop** command or by entering **q** in automore mode. The session could terminate itself automatically when it meets a stop condition such as duration or packet capture limit is met. It can terminate if an internal error occurs, or resource is full (specifically if disk is full in file mode).
- Dropped packets won't be shown at the end of the capture. However, only the count of dropped and oversized packets are displayed.

Default Wireshark Configuration

The table below shows the default Wireshark configuration.

Feature	Default Setting
Duration	No limit

Feature	Default Setting
Packets	No limit
Packet-length	No limit (full packet)
File size	No limit
Ring file storage	No
Buffer storage mode	Linear

About Embedded Packet Capture

EPC provides an embedded systems management facility that helps in tracing and troubleshooting packets. This feature allows network administrators to capture data packets flowing through, to, and from a Cisco device. The network administrator may define the capture buffer size and type (circular, or linear) and the maximum number of bytes of each packet to capture. You can throttle the packet capture rate using further administrative controls. For example, You can filter the packets using an Access Control List. You can further define the controls by specifying a maximum packet capture rate or by specifying a sampling interval.

Prior to Cisco IOS XE Amsterdam 17.2.1 , EPC isn't supported on an interface in shutdown state. Starting from Cisco IOS XE Amsterdam 17.2.1 , EPC is supported on an interface in shutdown state. This is useful in capturing packets on an interface as it's being brought up.

Benefits of Embedded Packet Capture

- Ability to capture IPv4 and IPv6 packets in the device, and also capture non-IP packets with MAC filter or match any MAC address.
- Extensible infrastructure for enabling packet capture points. A capture point is a traffic transit point where a packet is captured and associated with a buffer.
- Facility to export the packet capture in packet capture file (PCAP) format suitable for analysis using any external tool.
- Methods to decode data packets captured with varying degrees of detail.

Packet Data Capture

Packet data capture is the capture of data packets that are then stored in a buffer. You can define packet data captures by providing unique names and parameters.

You can perform the following actions on the capture:

- Activate captures at any interface.
- Apply access control lists (ACLs) or class maps to capture points.

Note Network Based Application Recognition (NBAR) and MAC-style class map is not supported.

- Destroy captures.

- Specify buffer storage parameters such as size and type. The size ranges from 1 MB to 100 MB. The default option for the buffer is linear and the other option for the buffer is circular.
- Specify match criteria that includes information about the protocol, IP address or port address.

How to Configure Packet Capture

The following sections provide information on configuring packet capture.

How to Configure Wireshark

To configure Wireshark, perform these basic steps.

1. Define a capture point.
2. Add or modify the parameters of the capture point.
3. Activate or deactivate a capture point.
4. Delete the capture point when you're no longer using it.

Defining a Capture Point

The example in this procedure defines a simple capture point. If you choose, you can define a capture point and all of its parameters with one instance of the **monitor capture** command.

Note Define an attachment point, direction of capture, and core filter to have a functional capture point.

Follow these steps to define a capture point.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**monitor capture** {*capture-name*} {**interface** *interface-type interface-id* | **control-plane**} {**in** | **out** | **both**} **Example:** `Device# monitor capture mycap interface GigabitEthernet1/0/1 in`	Defines the capture point, specifies the attachment point with which the capture point is associated, and specifies the direction of the capture. The keywords have these meanings: • *capture-name*—Specifies the name of the capture point to be defined (mycap is used in the example). Capture Name should be less than or equal to eight characters. Only

	Command or Action	Purpose
		alphanumeric characters and underscore (_) is permitted. • (Optional) **interface** *interface-type interface-id*—Specifies the attachment point with which the capture point is associated (GigabitEthernet1/0/1 is used in the example). **Note** Optionally, you can define multiple attachment points and all the parameters for this capture point with this one command instance. These parameters are discussed in the instructions for modifying capture point parameters. Range support is also available both for adding and removing attachment points. Use one of the following for *interface-type*: • **GigabitEthernet**—Specifies the attachment point as GigabitEthernet. • **vlan**—Specifies the attachment point as a VLAN. **Note** Only ingress capture (in) is allowed when using this interface as an attachment point. • (Optional) **control-plane**—Specifies the control plane as an attachment point. • **in** | **out** | **both**—Specifies the direction of capture.
Step 3	**monitor capture** {*capture-name*} [**match** {**any** | **ipv4 any any** | **ipv6**} **any any**}] **Example:** `Device# monitor capture mycap interface GigabitEthernet1/0/1 in match any`	Defines the core system filter. The keywords have these meanings: • *capture-name*—Specifies the name of the capture point to be defined (mycap is used in the example). • **match**—Specifies a filter. The first filter defined is the core filter.

	Command or Action	Purpose
		Note If a capture point doesn't have a core system filter or attachment points defined, it can't be activated. Attempting to activate a capture point that doesn't meet these requirements generates an error. • **ipv4**—Specifies an IP version 4 filter. • **ipv6**—Specifies an IP version 6 filter.

Step 4	**show monitor capture** {*capture-name*} [**parameter**]
	Example:
	`Device# show monitor capture mycap parameter`
	`monitor capture mycap interface GigabitEthernet1/0/1 in`
	`monitor capture mycap match any`
		Displays the capture point parameters defined in Step 2 and confirms that you defined a capture point.

Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

You can add more attachment points, modify the parameters of your capture point, then activate it. If you want to use your capture point just as it is, you can now activate it.

Note You can't change the parameters of a capture point using the methods presented in this topic.

If you enter an incorrect capture name, or an invalid/non existing attachment point, the switch shows errors, for example, "*Capture Name should be less than or equal to 8 characters. Only alphanumeric characters and underscore (_) is permitted*" and "*% Invalid input detected at '^' marker*" respectively.

Adding or Modifying Capture Point Parameters

Although listed in sequence, you can execute the steps to specify values for the parameters in any order. You can also specify them in one, two, or several lines. Except for attachment points, which can be multiple, you can replace any value with a more recent value by redefining the same option. You need to confirm interactively when certain parameters already specified are modified.

Starting with the Cisco IOS XE Amsterdam 17.3.x release, you can use packet length range and ether type as parameters for packet capture.

Follow these steps to modify the parameters of a capture point.

Before you begin

You must define a capture point before you can use these instructions.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**monitor capture** {*capture-name*} **match** {**any** | **mac** *mac-match-string* | **ipv4** {**any** | **host** | **protocol**} {**any** | **host**} | **ipv6** {**any** | **host** | **protocol**} {**any** | **host**}} **Example:** `Device# monitor capture mycap match ipv4 any any`	Defines the core system filter (**ipv4 any any**), defined either explicitly, through ACL or through a class map. You can specify a range of packet lengths for packet capture by using the **monitor capture** *capture-name* **interface** *interface-id* {**in** | **out** | **both**} **match pktlen-range max** *packet-length-in-bytes* **min** *packet-length-in-bytes* command in the EXEC configuration mode. You can define the core system filter through an ACL. You can configure the Ethertype of a protocol in the ACL. You can configure the same ACL in Wireshark to enable the capture of packets with a specific Ethertype.
Step 3	**monitor capture** {*capture-name*} **limit** {[**duration** *seconds*] [**packet-length** *size*] [**packets** *num*]} **Example:** `Device# monitor capture mycap limit duration 60 packet-len 400`	Specifies the session limit in seconds (60), packets captured, or the packet segment length retained by Wireshark (400).
Step 4	**monitor capture** {*capture-name*} **file** {**location** *filename*} **Example:**	Specifies the file association, if the capture point intends to capture packets rather than only display them.

	Command or Action	Purpose
	`Device# monitor capture mycap file location flash:mycap.pcap`	**Note** If the file exists, confirm if it can be overwritten.
Step 5	**monitor capture** {*capture-name*} **file** {**buffer-size** *size*} **Example:** `Device# monitor capture mycap file buffer-size 100`	Specifies the size of the memory buffer used by Wireshark to handle traffic bursts.

Step 6	**show monitor capture** {*capture-name*}[**parameter**]
	Example:
	`Device# show monitor capture mycap parameter`
	`monitor capture mycap interface GigabitEthernet1/0/1 in`
	`monitor capture mycap match ipv4 any any`
	`monitor capture mycap limit duration 60 packet-len 400`
	`monitor capture point mycap file location bootdisk:mycap.pcap`
	`monitor capture mycap file buffer-size 100`
		Displays the capture point parameters that you defined previously.

Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Modifying Parameters

Associating or Disassociating a Capture File

```
Device# monitor capture point mycap file location flash:mycap.pcap
Device# no monitor capture mycap file
```

Specifying a Memory Buffer Size for Packet Burst Handling

```
Device# monitor capture mycap buffer size 100
```

Defining an Explicit Core System Filter to Match Both IPv4 and IPv6

```
Device# monitor capture mycap match any
```

Specifying a Range of Packet Lengths for Packet Capture

```
Device(config)# monitor capture cap1 interface FortyGigabitEthernet 1/0/1 in match pktlen-range max 100 min 50
```

Specifying an ether type for packets

```
MAC ACL:
Device(config)#mac access-list extended mac1
```

```
Device(config-ext-macl)#permit any any 0x806 0x0
Device(config-ext-macl)exit
Device(config)#monitor capture mycap access-list mac1

IP ACL:
Device#ip access-list extended ip1
Device(config-ext-nacl)#permit 1 any any icmp-message-type
Device(config-ext-nacl)# exit
Device#monitor capture mycap access-list ip1
```

What to do next

If your capture point contains all the parameters you want, activate it.

Deleting Capture Point Parameters

Although listed in sequence, you can execute the steps to delete parameters in any order. You can also delete them in one, two, or several lines. Except for attachment points, which can be multiple, you can delete any parameter.

Follow these steps to delete the parameters of a capture point.

Before you begin

Define parameters of a capture point before you can use these instructions to delete it.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**no monitor capture** {*capture-name*} **match** **Example:** `Device# no monitor capture mycap match`	Deletes all filters defined on capture point (mycap).

Step 3	**no monitor capture** {*capture-name*} **limit** [**duration**] [**packet-length**] [**packets**]
	Example:
	`Device# no monitor capture mycap limit duration packet-len`
	`Device# no monitor capture mycap limit`
		Deletes the session time limit and the packet segment length retained by Wireshark. It leaves other specified limits in place.
		Deletes all limits on Wireshark.

Step 4	**no monitor capture** {*capture-name*} **file** [**location**] [**buffer-size**]
	Example:
	`Device# no monitor capture mycap file`
	`Device# no monitor capture mycap file location`
		Deletes the file association. The capture point will no longer capture packets. It only displays them.
		Deletes the file location association. The file location is no longer associated with the capture point. However, other defined file association is unaffected by this action.

	Command or Action	Purpose

Step 5	**show monitor capture** {*capture-name*} [**parameter**]
	Example:
	`Device# show monitor capture mycap parameter`
	`monitor capture mycap interface GigabitEthernet1/0/1 in`
		Displays the capture point parameters that remain defined after your parameter deletion operations. You can run this command at any point in the procedure to see what parameters are associated with a capture point.

Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

What to do next

If your capture point contains all the parameters you want, activate it.

Note If you delete the parameters when the capture point is active, the switch shows an error "*Capture is active*".

Deleting a Capture Point

Follow these steps to delete a capture point.

Before you begin

Define a capture point before you can use these instructions to delete it. Stop the capture point before you can delete it.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**no monitor capture** {*capture-name*} **Example:** `Device# no monitor capture mycap`	Deletes the specified capture point (mycap).
Step 3	**show monitor capture** {*capture-name*} [**parameter**] **Example:**	Displays a message indicating that the specified capture point doesn't exist because it was deleted.

	Command or Action	Purpose
	`Device# show monitor capture mycap parameter` `Capture mycap does not exist`	
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

You can define a new capture point with the same name as the one you deleted. You can perform these instructions when you want to start over with defining a capture point.

Activating and Deactivating a Capture Point

Follow these steps to activate or deactivate a capture point.

Before you begin

You can activate a capture point even if an attachment point and a core system filter are defined and the associated filename exists. In such an instance, the existing file is overwritten.

You can activate a capture point with no associated filename only to display. When the filename isn't specified, the packets are captured into the buffer. Live display (display during capture) is available in both file and buffer modes.

If no display filters are specified, packets aren't displayed live. All the packets captured by the core system filter are displayed. The default display mode is brief.

Procedure

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

	Command or Action	Purpose
Step 2	**monitor capture** {*capture-name*} **stop** **Example:** `Device# monitor capture name stop`	Deactivates a capture point.
Step 3	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 4	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 5	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

While activating and deactivating a capture point, you could encounter a few errors. Here are examples of some of the possible errors.

Missing attachment point on activation

```
Device# monitor capture mycap match any
Device# monitor capture mycap start
No Target is attached to capture failed to disable provision featurefailed to remove
policyfailed to disable provision featurefailed to remove policyfailed to disable provision
 featurefailed to remove policy
Capture statistics collected at software (Buffer):
 Capture duration - 0 seconds
 Packets received - 0
 Packets dropped - 0
 Packets oversized - 0

Unable to activate Capture.
Device# unable to get action unable to get action unable to get action
Device# monitor capture mycap interface g1/0/1 both
Device#monitor capture mycap start
Device#
*Nov 5 12:33:43.906: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
```

Missing filter on activation

```
Device# monitor capture mycap int g1/0/1 both
Device# monitor capture mycap start
Filter not attached to capture
Capture statistics collected at software (Buffer):
 Capture duration - 0 seconds
```

```
 Packets received - 0
 Packets dropped - 0
 Packets oversized - 0

Unable to activate Capture.
Device# monitor capture mycap match any
Device# monitor capture mycap start
Device#
*Nov 5 12:35:37.200: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
```

Attempting to activate a capture point while another one is already active

```
Device# monitor capture mycap start
PD start invoked while previous run is active Failed to start capture : Wireshark operation
 failure
Unable to activate Capture.
Device# show monitor capture

Status Information for Capture test
 Target Type:
 Interface: GigabitEthernet1/0/13, Direction: both
 Interface: GigabitEthernet1/0/14, Direction: both
 Status : Active
 Filter Details:
 Capture all packets
 Buffer Details:
 Buffer Type: LINEAR (default)
 Buffer Size (in MB): 10
 File Details:
 Associated file name: flash:cchh.pcap
 Limit Details:
 Number of Packets to capture: 0 (no limit)
 Packet Capture duration: 0 (no limit)
 Packet Size to capture: 0 (no limit)
 Maximum number of packets to capture per second: 1000
 Packet sampling rate: 0 (no sampling)

Status Information for Capture mycap
 Target Type:
 Interface: GigabitEthernet1/0/1, Direction: both
 Status : Inactive
 Filter Details:
 Capture all packets
 Buffer Details:
 Buffer Type: LINEAR (default)
 Buffer Size (in MB): 10
 File Details:
 File not associated
 Limit Details:
 Number of Packets to capture: 0 (no limit)
 Packet Capture duration: 0 (no limit)
 Packet Size to capture: 0 (no limit)
 Maximum number of packets to capture per second: 1000
 Packet sampling rate: 0 (no sampling)
Device# monitor capture test stop
Capture statistics collected at software (Buffer & Wireshark):
 Capture duration - 157 seconds
 Packets received - 0
 Packets dropped - 0
 Packets oversized - 0

Device#
*Nov 5 13:18:17.406: %BUFCAP-6-DISABLE: Capture Point test disabled.
Device# monitor capture mycap start
Device#
```

```
*Nov 5 13:18:22.664: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
Device#
```

Clearing the Capture Point Buffer

Follow these steps to clear the buffer contents or save them to an external file for storage.

Note If you have more than one capture that is storing packets in a buffer, clear the buffer before starting a new capture to avoid memory loss. Don't try to clear buffer on an active capture point.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**monitor capture** {*capture-name*} [**clear** | **export** *filename*] **Example:** `Device# monitor capture mycap clear`	Clear - Completely deletes the buffer. Export - Saves the captured packets in the buffer and deletes the buffer.
Step 3	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 4	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 5	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Examples: Capture Point Buffer Handling

Exporting Capture to a File

```
Device# monitor capture mycap export flash:mycap.pcap

Storage configured as File for this capture
```

Clearing Capture Point Buffer

```
Device# monitor capture mycap clear

 Capture configured with file options
```

What to do next

Note If you try to clear the capture point buffer on licenses other than DNA Advantage, the switch shows an error "*Failed to clear capture buffer: Capture Buffer BUSY*".

How to Implement Embedded Packet Capture

Managing Packet Data Capture

To manage Packet Data Capture in the buffer mode, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**monitor capture** *capture-name* **access-list** *access-list-name* **Example:** `Device# monitor capture mycap access-list v4acl`	Configures a monitor capture specifying an access list as the core filter for the packet capture.
Step 3	**monitor capture** *capture-name* **limit duration** *seconds* **Example:** `Device# monitor capture mycap limit duration 1000`	Configures monitor capture limits.
Step 4	**monitor capture** *capture-name* **interface** *interface-name* **both** **Example:** `Device# monitor capture mycap interface GigabitEthernet 0/0/1 both`	Configures monitor capture specifying an attachment point and the packet flow direction.
Step 5	**monitor capture** *capture-name* **buffer circular size** *bytes*	Configures a buffer to capture packet data.

	Command or Action	Purpose
	Example: `Device# monitor capture mycap buffer circular size 10`	
Step 6	**monitor capture** *capture-name* **start** **Example:** `Device# monitor capture mycap start`	Starts the capture of packet data at a traffic trace point into a buffer.
Step 7	**monitor capture** *capture-name* **stop** **Example:** `Device# monitor capture mycap stop`	Stops the capture of packet data at a traffic trace point.
Step 8	**monitor capture** *capture-name* **export** *file--location/file-name* **Example:** `Device# monitor capture mycap export tftp://10.1.88.9/mycap.pcap`	Exports captured data for analysis.
Step 9	**end** **Example:** `Device# end`	Returns to privileged EXEC mode.

Monitoring and Maintaining Captured Data

Perform this task to monitor and maintain the packet data captured. Capture buffer details and capture point details are displayed.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**show monitor capture** *capture-buffer-name* **buffer dump** **Example:** `Device# show monitor capture mycap buffer dump`	(Optional) Displays a hexadecimal dump of captured packet and its metadata.

	Command or Action	Purpose
Step 3	**show monitor capture** *capture-buffer-name* **parameter** **Example:** `Device# show monitor capture mycap parameter`	(Optional) Displays a list of commands that were used to specify the capture.
Step 4	**debug epc capture-point** **Example:** `Device# debug epc capture-point`	(Optional) Enables packet capture point debugging.
Step 5	**debug epc provision** **Example:** `Device# debug epc provision`	(Optional) Enables packet capture provisioning debugging.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuration Examples for Packet Capture

The following sections provide configuration examples for packet capture.

Configuration Examples for Wireshark

The following sections provide configuration examples for Wireshark.

Example: Displaying a Brief Output from a .pcap File

You can display the output from a .pcap file by entering:

```
Device# show monitor capture file flash:mycap.pcap brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=0/0, ttl=254
  2 0.000051000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=0/0, ttl=255 (request in 1)
  3 0.000908000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=1/256, ttl=254
  4 0.001782000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=1/256, ttl=255 (request in 3)
  5 0.002961000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=2/512, ttl=254
```

```
  6 0.003676000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=2/512, ttl=255 (request in 5)
  7 0.004835000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=3/768, ttl=254
  8 0.005579000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=3/768, ttl=255 (request in 7)
  9 0.006850000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=4/1024, ttl=254
 10 0.007586000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=4/1024, ttl=255 (request in 9)
 11 0.008768000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=5/1280, ttl=254
 12 0.009497000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=5/1280, ttl=255 (request in 11)
 13 0.010695000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=6/1536, ttl=254
 14 0.011427000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=6/1536, ttl=255 (request in 13)
 15 0.012728000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=7/1792, ttl=254
 16 0.013458000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=7/1792, ttl=255 (request in 15)
 17 0.014652000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=8/2048, ttl=254
 18 0.015394000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=8/2048, ttl=255 (request in 17)
 19 0.016682000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=9/2304, ttl=254
 20 0.017439000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=9/2304, ttl=255 (request in 19)
 21 0.018655000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=10/2560, ttl=254
 22 0.019385000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=10/2560, ttl=255 (request in 21)
 23 0.020575000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=11/2816, ttl=254
 --More‹
```

Example: Displaying Detailed Output from a .pcap File

You can display the detailed .pcap file output by entering:

```
Device# show monitor capture file flash:mycap.pcap detailed
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

Frame 1: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov  6, 2015 11:44:48.322497000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1446810288.322497000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 114 bytes (912 bits)
    Capture Length: 114 bytes (912 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:icmp:data]
Ethernet II, Src: Cisco_f3:63:46 (00:e1:6d:f3:63:46), Dst: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
```

```
    Destination: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
        Address: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: Cisco_f3:63:46 (00:e1:6d:f3:63:46)
        Address: Cisco_f3:63:46 (00:e1:6d:f3:63:46)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.2 (10.10.10.2), Dst: 10.10.10.1 (10.10.10.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not
ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport)
(0x00)
    Total Length: 100
    Identification: 0x04ba (1210)
    Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 254
    Protocol: ICMP (1)
    Header checksum: 0x8fc8 [validation disabled]
        [Good: False]
        [Bad: False]
    Source: 10.10.10.2 (10.10.10.2)
    Destination: 10.10.10.1 (10.10.10.1)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xe4db [correct]
    Identifier (BE): 46 (0x002e)
    Identifier (LE): 11776 (0x2e00)
    Sequence number (BE): 0 (0x0000)
    Sequence number (LE): 0 (0x0000)
    Data (72 bytes)

0000  00 00 00 00 09 c9 8f 77 ab cd ab cd ab cd ab cd   .......w........
0010  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0020  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0030  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0040  ab cd ab cd ab cd ab cd                           ........
        Data: 0000000009c98f77abcdabcdabcdabcdabcdabcdabcdabcd...
        [Length: 72]

Frame 2: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
    Interface id: 0
```

Example: Displaying a Packet Dump Output from a .pcap File.

You can display the packet dump output by entering:

```
Device# show monitor capture file flash:mycap.pcap dump
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

0000  00 e1 6d 31 f1 c6 00 e1 6d f3 63 46 08 00 45 00   ..m1....m.cF..E.
0010  00 64 04 ba 00 00 fe 01 8f c8 0a 0a 0a 02 0a 0a   .d..............
```

```
0020  0a 01 08 00 e4 db 00 2e 00 00 00 00 00 00 09 c9   ................
0030  8f 77 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   .w..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd                                            ..

0000  00 e1 6d 31 f1 80 00 e1 6d 31 f1 80 08 00 45 00   ..m1....m1....E.
0010  00 64 04 ba 00 00 ff 01 8e c8 0a 0a 0a 01 0a 0a   .d..............
0020  0a 02 00 00 ec db 00 2e 00 00 00 00 00 00 09 c9   ................
0030  8f 77 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   .w..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd                                            ..

0000  00 e1 6d 31 f1 c6 00 e1 6d f3 63 46 08 00 45 00   ..m1....m.cF..E.
0010  00 64 04 bb 00 00 fe 01 8f c7 0a 0a 0a 02 0a 0a   .d..............
0020  0a 01 08 00 e4 d7 00 2e 00 01 00 00 00 00 09 c9   ................
0030  8f 7a ab cd ab cd ab cd ab cd ab cd ab cd ab cd   .z..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
```

Example: Displaying Packets from a .pcap File using a Display Filter

You can display the .pcap file packets output by entering:

```
Device# show monitor capture file flash:mycap.pcap display-filter "ip.src == 10.10.10.2"
brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=0/0, ttl=254
  3 0.000908000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=1/256, ttl=254
  5 0.002961000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=2/512, ttl=254
  7 0.004835000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=3/768, ttl=254
  9 0.006850000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=4/1024, ttl=254
 11 0.008768000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=5/1280, ttl=254
 13 0.010695000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=6/1536, ttl=254
 15 0.012728000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=7/1792, ttl=254
 17 0.014652000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=8/2048, ttl=254
 19 0.016682000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=9/2304, ttl=254
 21 0.018655000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=10/2560, ttl=254
 23 0.020575000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=11/2816, ttl=254
```

Example: Displaying the Number of Packets Captured in a .pcap File

You can display the number of packets captured in a .pcap file by entering:

```
Device# show monitor capture file flash:mycap.pcap packet-count
File name:           /flash/mycap.pcap
Number of packets:   50
```

Example: Displaying a Single Packet Dump from a .pcap File

You can display a single packet dump from a .pcap file by entering:

```
Device#  show monitor capture file flash:mycap.pcap packet-number 10 dump
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

0000  00 e1 6d 31 f1 80 00 e1 6d 31 f1 80 08 00 45 00   ..m1....m1....E.
0010  00 64 04 be 00 00 ff 01 8e c4 0a 0a 0a 01 0a 0a   .d..............
0020  0a 02 00 00 ec ce 00 2e 00 04 00 00 00 00 09 c9   ................
0030  8f 80 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd
```

Example: Displaying Statistics of Packets Captured in a .pcap File

You can display the statistics of the packets captured in a .pcap file by entering:

```
Device# show monitor capture file flash:mycap.pcap statistics "h225,counter"
================== H225 Message and Reason Counter ==================
RAS-Messages:
Call Signalling:
=====================================================================
```

Example: Simple Capture and Display

This example shows how to monitor traffic in the Layer 3 interface Gigabit Ethernet 1/0/1:

Step 1: Define a capture point to match on the relevant traffic by entering:

```
Device# monitor capture mycap interface GigabitEthernet1/0/3 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 50
Device# monitor capture mycap buffer size 100
```

To avoid high CPU utilization, a low packet count and duration as limits are set.

Step 2: Confirm that the capture point has been correctly defined by entering:

```
Device# show monitor capture mycap parameter
      monitor capture mycap interface GigabitEthernet1/0/3 in
   monitor capture mycap match ipv4  any any
   monitor capture mycap buffer size 100
   monitor capture mycap limit packets 50 duration 60

Device# show monitor capture mycap
Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
```

```
   Buffer Type: LINEAR (default)
   Buffer Size (in MB): 100
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 50
   Packet Capture duration: 60
   Packet Size to capture: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

Step 3: Start the capture process and display the results.

```
Device# monitor capture mycap start display
 Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1   0.000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030, seq=0/0,
 ttl=254
  2   0.003682   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=1/256, ttl=254
  3   0.006586   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=2/512, ttl=254
  4   0.008941   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=3/768, ttl=254
  5   0.011138   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=4/1024, ttl=254
  6   0.014099   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=5/1280, ttl=254
  7   0.016868   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=6/1536, ttl=254
  8   0.019210   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=7/1792, ttl=254
  9   0.024785   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=8/2048, ttl=254
 --More--
```

Step 4: Delete the capture point by entering:

```
Device# no monitor capture mycap
```

Note A **stop** command isn't required in this particular case since we've set a limit and the capture stops once that limit is reached.

For more information on syntax to be used for pcap statistics, refer the "*Additional References*" section.

Example: Simple Capture and Store

This example shows how to capture packets to a filter:

Step 1: Define a capture point to match on the relevant traffic and associate it to a file by entering:

```
Device# monitor capture mycap interface GigabitEthernet1/0/3 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 50
Device# monitor capture mycap file location flash:mycap.pcap
```

Step 2: Confirm that the capture point has been correctly defined by entering:

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/3 in
   monitor capture mycap match ipv4  any any
   monitor capture mycap file location flash:mycap.pcap
   monitor capture mycap limit packets 50 duration 60

Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: LINEAR (default)
  File Details:
   Associated file name: flash:mycap.pcap
  Limit Details:
   Number of Packets to capture: 50
   Packet Capture duration: 60
   Packet Size to capture: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

Step 3: Launch packet capture by entering:

```
Device# monitor capture mycap start
```

Step 4: Display extended capture statistics during runtime by entering:

```
Device#  show monitor capture mycap capture-statistics
Capture statistics collected at software:
        Capture duration - 15 seconds
        Packets received - 40
        Packets dropped - 0
        Packets oversized - 0
        Packets errored - 0
        Packets sent - 40
        Bytes received - 7280
        Bytes dropped - 0
        Bytes oversized - 0
        Bytes errored - 0
        Bytes sent ¬ 4560
```

Step 5: After sufficient time has passed, stop the capture by entering:

```
# monitor capture mycap stop
Capture statistics collected at software (Buffer & Wireshark):
        Capture duration - 20 seconds
        Packets received - 50
        Packets dropped - 0
        Packets oversized - 0
```

Note Alternatively, you could allow the capture operation to stop automatically after the time has elapsed or the packet count are met.

The mycap.pcap file now contains the captured packets.

Step 6: Display extended capture statistics after stop by entering:

```
Device#  show monitor capture mycap capture-statistics
Capture statistics collected at software:
        Capture duration - 20 seconds
        Packets received - 50
        Packets dropped - 0
        Packets oversized - 0
        Packets errored - 0
        Packets sent - 50
        Bytes received - 8190
        Bytes dropped - 0
        Bytes oversized - 0
        Bytes errored - 0
        Bytes sent ¬ 5130
```

Step 7: Display the packets by entering:

```
Device# show monitor capture file flash:mycap.pcap
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=0/0, ttl=254
  2 0.002555000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=1/256, ttl=254
  3 0.006199000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=2/512, ttl=254
  4 0.009199000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=3/768, ttl=254
  5 0.011647000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=4/1024, ttl=254
  6 0.014168000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=5/1280, ttl=254
  7 0.016737000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=6/1536, ttl=254
  8 0.019403000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=7/1792, ttl=254
  9 0.022151000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=8/2048, ttl=254
 10 0.024722000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=9/2304, ttl=254
 11 0.026890000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=10/2560, ttl=254
 12 0.028862000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=11/2816, ttl=254
 --More--
```

For more information on syntax to be used for pcap statistics, refer the "*Additional References*" section.

Step 8: Delete the capture point by entering:

```
Device# no monitor capture mycap
```

Example: Using Buffer Capture

This example shows how to use buffer capture:

Step 1: Launch a capture session with the buffer capture option by entering:

```
Device# monitor capture mycap interface GigabitEthernet1/0/3 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap buffer circular size 1
Device# monitor capture mycap start
```

Step 2: Determine whether the capture is active by entering:

```
Device# show monitor capture mycap
Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Active
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: CIRCULAR
   Buffer Size (in MB): 1
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Maximum number of packets to capture per second: 1000
   Packet sampling rate: 0 (no sampling)
```

Step 3: Display extended capture statistics during runtime by entering:

```
Device# show monitor capture mycap capture-statistics
Capture statistics collected at software:
        Capture duration - 88 seconds
        Packets received - 1000
        Packets dropped - 0
        Packets oversized - 0
        Packets errored - 0
        Packets sent - 1000
        Bytes received - 182000
        Bytes dropped - 0
        Bytes oversized - 0
        Bytes errored - 0
        Bytes sent - 114000
```

Step 4: Stop the capture by entering:

```
Device# monitor capture mycap stop
Capture statistics collected at software (Buffer):
        Capture duration - 2185 seconds
        Packets received - 51500
        Packets dropped - 0
        Packets oversized - 0
```

Step 5: Display extended capture statistics after stop by entering:

```
Device# show monitor capture mycap capture-statistics
Capture statistics collected at software:
        Capture duration - 156 seconds
        Packets received - 2000
        Packets dropped - 0
        Packets oversized - 0
        Packets errored - 0
        Packets sent - 2000
        Bytes received - 364000
        Bytes dropped - 0
        Bytes oversized - 0
        Bytes errored - 0
        Bytes sent - 228000
```

Step 6: Determine whether the capture is active by entering:

```
Device# show monitor capture mycap
Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: CIRCULAR
   Buffer Size (in MB): 1
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Maximum number of packets to capture per second: 1000
   Packet sampling rate: 0 (no sampling)
```

Step 7: Display the packets in the buffer by entering:

```
Device# show monitor capture mycap buffer brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1   0.000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40057/31132, ttl=254
  2   0.000030   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40058/31388, ttl=254
  3   0.000052   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40059/31644, ttl=254
  4   0.000073   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40060/31900, ttl=254
  5   0.000094   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40061/32156, ttl=254
  6   0.000115   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40062/32412, ttl=254
  7   0.000137   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40063/32668, ttl=254
  8   0.000158   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40064/32924, ttl=254
  9   0.000179   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40065/33180, ttl=254
```

```
 10   0.000200   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40066/33436, ttl=254
 11   0.000221   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40067/33692, ttl=254
 12   0.000243   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40068/33948, ttl=254
 --More--
```

Notice that the packets are buffered.

Step 8: Display the packets in other display modes.

```
Device# show monitor capture mycap buffer detailed
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

Frame 1: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov  6, 2015 18:10:06.297972000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1446833406.297972000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 114 bytes (912 bits)
    Capture Length: 114 bytes (912 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:icmp:data]
Ethernet II, Src: Cisco_f3:63:46 (00:e1:6d:f3:63:46), Dst: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)

    Destination: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
        Address: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: Cisco_f3:63:46 (00:e1:6d:f3:63:46)
        Address: Cisco_f3:63:46 (00:e1:6d:f3:63:46)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.2 (10.10.10.2), Dst: 10.10.10.1 (10.10.10.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not
ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport)
(0x00)
    Total Length: 100
    Identification: 0xabdd (43997)
    Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 254
    Protocol: ICMP (1)
    Header checksum: 0xe8a4 [validation disabled]
        [Good: False]
        [Bad: False]
    Source: 10.10.10.2 (10.10.10.2)
    Destination: 10.10.10.1 (10.10.10.1)
Internet Control Message Protocol
```

```
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xa620 [correct]
    Identifier (BE): 56 (0x0038)
    Identifier (LE): 14336 (0x3800)
    Sequence number (BE): 40057 (0x9c79)
    Sequence number (LE): 31132 (0x799c)
    Data (72 bytes)

0000  00 00 00 00 0b 15 30 63 ab cd ab cd ab cd ab cd   ......0c........
0010  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0020  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0030  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0040  ab cd ab cd ab cd ab cd                           ........
       Data: 000000000b153063abcdabcdabcdabcdabcdabcdabcdabcd...
       [Length: 72]

Frame 2: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0


Device# show monitor capture mycap buffer dump
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

0000  00 e1 6d 31 f1 c6 00 e1 6d f3 63 46 08 00 45 00   ..m1....m.cF..E.
0010  00 64 ab dd 00 00 fe 01 e8 a4 0a 0a 0a 02 0a 0a   .d..............
0020  0a 01 08 00 a6 20 00 38 9c 79 00 00 00 00 0b 15   ..... .8.y......
0030  30 63 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   0c..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd                                            ..

0000  00 e1 6d 31 f1 c6 00 e1 6d f3 63 46 08 00 45 00   ..m1....m.cF..E.
0010  00 64 ab de 00 00 fe 01 e8 a3 0a 0a 0a 02 0a 0a   .d..............
0020  0a 01 08 00 a6 1d 00 38 9c 7a 00 00 00 00 0b 15   .......8.z......
0030  30 65 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   0e..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd
```

Step 9: Clear the buffer by entering:

```
Device# monitor capture mycap clear
```

Note NOTE - Clearing the buffer deletes the buffer along with the contents.

Note If you require the buffer to display its contents, run the clear commands after show commands.

Step 10: Restart the traffic, wait for 10 seconds, then display the buffer contents by entering:

Note We can't run show from buffer during an active capture. Stop capture before running show from buffer. We can however run a show on a pcap file during an active capture in both file and buffer mode. In file mode, we can display the packets in the pcap file of the current capture session as well when the capture is active.

```
Device# monitor capture mycap start
Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Active
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: CIRCULAR
   Buffer Size (in MB): 1
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Maximum number of packets to capture per second: 1000
   Packet sampling rate: 0 (no sampling)
```

Step 11: Stop the packet capture and display the buffer contents by entering:

```
Device# monitor capture mycap stop
Capture statistics collected at software (Buffer):
 Capture duration - 111 seconds
 Packets received - 5000
 Packets dropped - 0
 Packets oversized - 0
```

Step 12: Determine whether the capture is active by entering:

```
Device# show monitor capture mycap
Status Information for Capture mycap
 Target Type:
  Interface: GigabitEthernet1/0/3, Direction: in
 Status : Inactive
Filter Details:
 IPv4
  Source IP: any
  Destination IP: any
 Protocol: any
Buffer Details:
 Buffer Type: CIRCULAR
 Buffer Size (in MB): 1
File Details:
 File not associated
Limit Details:
 Number of Packets to capture: 0 (no limit)
 Packet Capture duration: 0 (no limit)
 Packet Size to capture: 0 (no limit)
```

```
 Maximum number of packets to capture per second: 1000
 Packet sampling rate: 0 (no sampling)
```

Step 13: Display the packets in the buffer by entering:

```
Device# show monitor capture mycap buffer brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=0/0, ttl=254
  2 0.000030000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=1/256, ttl=254
  3 0.000051000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=2/512, ttl=254
  4 0.000072000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=3/768, ttl=254
  5 0.000093000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=4/1024, ttl=254
  6 0.000114000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=5/1280, ttl=254
  7 0.000136000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=6/1536, ttl=254
  8 0.000157000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=7/1792, ttl=254
  9 0.000178000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=8/2048, ttl=254
 10 0.000199000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=9/2304, ttl=254
 11 0.000220000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=10/2560, ttl=254
 12 0.000241000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=11/2816, ttl=254
 --More‹
```

Step 14: Store the buffer contents to the mycap.pcap file in the internal flash: storage device by entering:

```
Device# monitor capture mycap export flash:mycap.pcap
Exported Successfully
```

Note The current implementation of export is such that when you run the command, export is "started" but not complete when it returns the prompt to you. So we have to wait for a message display on the console from Wireshark before it can run a display of packets in the file.

Step 15: Display capture packets from the file by entering:

```
Device#  show monitor capture file flash:mycap.pcap
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=0/0, ttl=254
  2 0.000030000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=1/256, ttl=254
  3 0.000051000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=2/512, ttl=254
  4 0.000072000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=3/768, ttl=254
  5 0.000093000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=4/1024, ttl=254
```

```
  6 0.000114000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=5/1280, ttl=254
  7 0.000136000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=6/1536, ttl=254
  8 0.000157000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=7/1792, ttl=254
  9 0.000178000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=8/2048, ttl=254
 10 0.000199000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=9/2304, ttl=254
 11 0.000220000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=10/2560, ttl=254
 12 0.000241000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=11/2816, ttl=254
 --More--
```

Step 16: Delete the capture point by entering:

```
Device# no monitor capture mycap
```

Example: Simple Capture and Store of Packets in Egress Direction

This example shows how to capture packets to a filter:

Step 1: Define a capture point to match on the relevant traffic and associate it to a file by entering:

```
Device# monitor capture mycap interface Gigabit 1/0/1 out match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 100
Device# monitor capture mycap file location flash:mycap.pcap buffer-size 90
```

Step 2: Confirm that the capture point has been correctly defined by entering:

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/1 out
   monitor capture mycap match ipv4  any any
   monitor capture mycap file location flash:mycap.pcap buffer-size 90
   monitor capture mycap limit packets 100 duration 60

Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/1, Direction: out
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: LINEAR (default)
  File Details:
   Associated file name: flash:mycap.pcap
   Size of buffer(in MB): 90
  Limit Details:
   Number of Packets to capture: 100
   Packet Capture duration: 60
   Packet Size to capture: 0 (no limit)
   Packets per second: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

Step 3: Launch packet capture by entering:

```
Device# monitor capture mycap start
A file by the same capture file name already exists, overwrite?[confirm]
Turning on lock-step mode

Device#
*Oct 14 09:35:32.661: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
```

Note Allow the capture operation to stop automatically after the time has elapsed or the packet count are met. When you see the following message in the output, you'll know that the capture operation has stopped:

```
*Oct 14 09:36:34.632: %BUFCAP-6-DISABLE_ASYNC: Capture Point mycap disabled. Rea
son : Wireshark Session Ended
```

The mycap.pcap file now contains the captured packets.

Step 4: Display the packets by entering:

```
Device# show monitor capture file flash:mycap.pcap
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  0.000000    10.1.1.30 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000    10.1.1.31 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000    10.1.1.32 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000    10.1.1.33 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000    10.1.1.34 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000    10.1.1.35 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000    10.1.1.36 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000    10.1.1.37 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000    10.1.1.38 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000    10.1.1.39 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 5: Delete the capture point by entering:

```
Device# no monitor capture mycap
```

Configuration Examples for Embedded Packet Capture

The following sections provide configuration examples for EPC.

Example: Managing Packet Data Capture

The following example shows how to manage packet data capture:

```
Device> enable
Device# monitor capture mycap access-list v4acl
Device# monitor capture mycap limit duration 1000
Device# monitor capture mycap interface GigabitEthernet 0/0/1 both
Device# monitor capture mycap buffer circular size 10
Device# monitor capture mycap start
Device# monitor capture mycap stop
Device# monitor capture mycap export tftp://10.1.88.9/mycap.pcap
Device# end
```

Example: Monitoring and Maintaining Captured Data

The following example shows how to dump packets in ASCII format:

```
Device# show monitor capture mycap buffer dump
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

0
0000: 01005E00 00020000 0C07AC1D 080045C0 ..^...........E.
0010: 00300000 00000111 CFDC091D 0002E000 .0..............
0020: 000207C1 07C1001C 802A0000 10030AFA .........*......
0030: 1D006369 73636F00 0000091D 0001 ..example.......
1
0000: 01005E00 0002001B 2BF69280 080046C0 ..^.....+.....F.
0010: 00200000 00000102 44170000 0000E000 . ......D.......
0020: 00019404 00001700 E8FF0000 0000 ..............
2
0000: 01005E00 0002001B 2BF68680 080045C0 ..^.....+.....E.
0010: 00300000 00000111 CFDB091D 0003E000 .0..............
0020: 000207C1 07C1001C 88B50000 08030A6E ..............n
0030: 1D006369 73636F00 0000091D 0001 ..example.......
3
0000: 01005E00 000A001C 0F2EDC00 080045C0 ..^...........E.
0010: 003C0000 00000258 CE7F091D 0004E000 .<.....X........
0020: 000A0205 F3000000 00000000 00000000 ................
0030: 00000000 00D10001 000C0100 01000000 ................
0040: 000F0004 00080501 0300
```

The following example shows how to display the list of commands used to configure the capture named mycap:

```
Device# show monitor capture mycap parameter
monitor capture mycap interface GigabitEthernet 1/0/1 both
monitor capture mycap match any
monitor capture mycap buffer size 10
monitor capture mycap limit pps 1000
```

The following example shows how to debug the capture point:

```
Device# debug epc capture-point
EPC capture point operations debugging is on

Device# monitor capture mycap start
*Jun 4 14:17:15.463: EPC CP: Starting the capture cap1
*Jun 4 14:17:15.463: EPC CP: (brief=3, detailed=4, dump=5) = 0
*Jun 4 14:17:15.463: EPC CP: final check before activation
*Jun 4 14:17:15.463: EPC CP: setting up c3pl infra
*Jun 4 14:17:15.463: EPC CP: Setup c3pl acl-class-policy
*Jun 4 14:17:15.463: EPC CP: Creating a class
*Jun 4 14:17:15.464: EPC CP: Creating a class : Successful
*Jun 4 14:17:15.464: EPC CP: class-map Created
*Jun 4 14:17:15.464: EPC CP: creating policy-name epc_policy_cap1
*Jun 4 14:17:15.464: EPC CP: Creating Policy epc_policy_cap1 of type 49 and client type 21
*Jun 4 14:17:15.464: EPC CP: Storing a Policy
*Jun 4 14:17:15.464: EPC CP: calling ppm_store_policy with epc_policy
*Jun 4 14:17:15.464: EPC CP: Creating Policy : Successful
*Jun 4 14:17:15.464: EPC CP: policy-map created
*Jun 4 14:17:15.464: EPC CP: creating filter for ANY
*Jun 4 14:17:15.464: EPC CP: Adding acl to class : Successful
*Jun 4 14:17:15.464: EPC CP: Setup c3pl class to policy
*Jun 4 14:17:15.464: EPC CP: Attaching Class to Policy
*Jun 4 14:17:15.464: EPC CP: Attaching epc_class_cap1 to epc_policy_cap1
*Jun 4 14:17:15.464: EPC CP: Attaching Class to Policy : Successful
*Jun 4 14:17:15.464: EPC CP: setting up c3pl qos
*Jun 4 14:17:15.464: EPC CP: DBG> Set packet rate limit to 1000
```

```
*Jun 4 14:17:15.464: EPC CP: creating action for policy_map epc_policy_cap1 class_map
epc_class_cap1
*Jun 4 14:17:15.464: EPC CP: DBG> Set packet rate limit to 1000
*Jun 4 14:17:15.464: EPC CP: Activating Interface GigabitEthernet1/0/1 direction both
*Jun 4 14:17:15.464: EPC CP: Id attached 0
*Jun 4 14:17:15.464: EPC CP: inserting into active lists
*Jun 4 14:17:15.464: EPC CP: Id attached 0
*Jun 4 14:17:15.465: EPC CP: inserting into active lists
*Jun 4 14:17:15.465: EPC CP: Activating Vlan
*Jun 4 14:17:15.465: EPC CP: Deleting all temp interfaces
*Jun 4 14:17:15.465: %BUFCAP-6-ENABLE: Capture Point cap1 enabled.
*Jun 4 14:17:15.465: EPC CP: Active Capture 1

Device# monitor capture mycap1 stop
*Jun 4 14:17:31.963: EPC CP: Stopping the capture cap1
*Jun 4 14:17:31.963: EPC CP: Warning: unable to unbind capture cap1
*Jun 4 14:17:31.963: EPC CP: Deactivating policy-map
*Jun 4 14:17:31.963: EPC CP: Policy epc_policy_cap1
*Jun 4 14:17:31.964: EPC CP: Deactivating policy-map Successful
*Jun 4 14:17:31.964: EPC CP: removing povision feature
*Jun 4 14:17:31.964: EPC CP: Found action for policy-map epc_policy_cap1 class-map
epc_class_cap1
*Jun 4 14:17:31.964: EPC CP: cleanning up c3pl infra
*Jun 4 14:17:31.964: EPC CP: Removing Class epc_class_cap1 from Policy
*Jun 4 14:17:31.964: EPC CP: Removing Class from epc_policy_cap1
*Jun 4 14:17:31.964: EPC CP: Successfully removed
*Jun 4 14:17:31.964: EPC CP: Removing acl mac from class
*Jun 4 14:17:31.964: EPC CP: Removing acl from class : Successful
*Jun 4 14:17:31.964: EPC CP: Removing all policies
*Jun 4 14:17:31.964: EPC CP: Removing Policy epc_policy_cap1
*Jun 4 14:17:31.964: EPC CP: Removing Policy : Successful
*Jun 4 14:17:31.964: EPC CP: Removing class epc_class_cap1
*Jun 4 14:17:31.965: EPC CP: Removing class : Successful
*Jun 4 14:17:31.965: %BUFCAP-6-DISABLE: Capture Point cap1 disabled.
*Jun 4 14:17:31.965: EPC CP: Active Capture 0
```

The following example shows how to debug the Embedded Packet Capture (EPC) provisioning:

```
Device# debug epc provision
EPC provisionioning debugging is on

Device# monitor capture mycap start
*Jun 4 14:17:54.991: EPC PROV: No action found for policy-map epc_policy_cap1 class-map
epc_class_cap1
*Jun 4 14:17:54.991: EPC PROV:
*Jun 4 14:17:54.991: Attempting to install service policy epc_policy_cap1
*Jun 4 14:17:54.992: EPC PROV: Attached service policy to epc idb subblock
*Jun 4 14:17:54.992: EPC PROV: Successful. Create feature object
*Jun 4 14:17:54.992: EPC PROV:
*Jun 4 14:17:54.992: Attempting to install service policy epc_policy_cap1
*Jun 4 14:17:54.992: EPC PROV: Successful. Create feature object
*Jun 4 14:17:54.992: %BUFCAP-6-ENABLE: Capture Point cap1 enabled.

Device# monitor capture mycap stop
*Jun 4 14:18:02.503: EPC PROV: Successful. Remove feature object
*Jun 4 14:18:02.504: EPC PROV: Successful. Remove feature object
*Jun 4 14:18:02.504: EPC PROV: Destroyed epc idb subblock
*Jun 4 14:18:02.504: EPC PROV: Found action for policy-map epc_policy_cap1 class-map
epc_class_cap1
*Jun 4 14:18:02.504: EPC PROV: Deleting EPC action
*Jun 4 14:18:02.504: EPC PROV: Successful. CLASS_REMOVE, policy-map epc_policy_cap1, class
 epc_class_cap1
*Jun 4 14:18:02.504: %BUFCAP-6-DISABLE: Capture Point cap1 disabled.
```

Additional References

Related Documents

Related Topic	Document Title
Display Filters	For syntax of Display Filters, refer to: Display Filter Reference
Pcap file statistics	For syntax used to display pcap file statistics, refer to "-z" option details at: Tshark Command Reference

Feature History for Packet Capture

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

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

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Packet Capture	The Packet Capture feature is an onboard packet capture facility that allows network administrators to capture packets flowing to, through, and from the device.
Cisco IOS XE Amsterdam 17.3.1	Enhancements to Embedded Packet Capture (EPC) Packet filters: Packet length and Ether type	In Embedded Packet Capture (EPC), packets can be capture based on packet length or ether type.
Cisco IOS XE Cupertino 17.7.1	Packet Capture	Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).
Cisco IOS XE Cupertino 17.8.1	Embedded Packet Capture	Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/.