



Configuring Packet Capture

- [Prerequisites for Configuring Packet Capture, on page 1](#)
- [Restrictions for Configuring Packet Capture, on page 2](#)
- [Information About Packet Capture, on page 4](#)
- [How to Configure Packet Capture, on page 14](#)
- [Configuration Examples for Packet Capture, on page 30](#)
- [Additional References, on page 48](#)
- [Feature History for Configuring Packet Capture, on page 48](#)

Prerequisites for Configuring Packet Capture

Packet capture is supported on Cisco Catalyst 9400 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 and 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. Some guidelines for using the system resources are provided in the table below.

Table 1: System Requirements for the EPC Subsystem

System Resources	Requirements
Hardware	CPU utilization requirements are platform dependent.

System Resources	Requirements
Memory	The packet buffer is stored in DRAM. The size of the packet buffer is user specified.
Diskspace	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

- Global packet capture on Wireshark is not supported.
- Limiting circular file storage by file size is not supported.
- File limit is limited to the size of the flash in DNA Advantage.
- Wireshark cannot 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, the capture must be restarted manually.
- The streaming capture mode supports approximately 1500 pps; lock-step mode supports approximately 45 Mbps (measured with 256-byte packets); lock-step mode with high speed capture supports roughly 750Mbps (measure with 256-byte packets). When the matching traffic rate exceeds this number, you may experience packet loss. Only one session can be started when using high-speed capture mode.
- You cannot make changes to a capture point when the capture is active.
- Wireshark does not capture packets dropped by floodblock.
- Only one ACL (IPv4, IPv6 or MAC) is allowed in a Wireshark class map.
- ACL logging and Wireshark are incompatible. Once Wireshark is activated, it takes priority. All traffic, including that being captured by ACL logging on any ports, will be 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 will occur 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 be 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 are not synchronized to the standby supervisor in NSF and SSO scenarios.

Embedded Wireshark is supported with the following limitations:

- Capture filters and display filters are not supported.
- Active capture decoding is not 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.

Restrictions for Configuring Embedded Packet Capture

- Layer 2 EtherChannels are not supported.
- Neither VRFs, management ports, nor private VLANs can be used as attachment points.
- Embedded Packet Capture (EPC) is not supported on logical ports, which includes port channels, switch virtual interfaces (SVIs), and subinterfaces. It is supported only on physical ports.
- EPC is not supported on a VLAN interface that is in shutdown state.
- If the user changes interface from switch port to routed port (Layer 2 to Layer 3) or vice versa, they must delete the capture point and create a new one, once the interface comes back up. Stop/start the capture point will not work.
- Packets captured in the output direction of an interface might not reflect the changes made by the device rewrite (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 is 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 does not capture the replicated packets on egress.
- The Rewrite information of both ingress and egress packets are not captured.
- CPU-injected packets are considered control plane packets. Therefore, these types of packets will not be captured on an interface egress capture.
- Control plane packets are not rate limited and performance impacting. Please use filters to limit control plane packet capture.
- Decoding of protocols such as Control and Provisioning of Wireless Access Points (CAPWAP) is supported in DNA Advantage.
- Up to 8 capture points can be defined, but only one can be active at a time. You need to stop one before you can start the other.
- MAC filter will not 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 will not be supported on a Layer 3 port or SVI.
- MAC filter cannot capture Layer 2 packets (ARP) on Layer 3 interfaces.
- IPv6-based ACLs are not supported in VACL.

- EPC cannot capture based on the underlying routing protocols in MPLS packets.

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 and to 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 of the characteristics associated with a given instance of Wireshark: 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 do not 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 and activated simultaneously.

In case of stacked systems, the capture point is activated on the active member. A switchover will terminate any active packet capture session and it will have to be restarted.

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, which is copied and passed to Wireshark. To be displayed by Wireshark, a packet must pass through an attachment point, as well as all of 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 hardware-forwarded traffic is copied to software for Wireshark purposes.
- **Capture filter**—The capture filter is applied by Wireshark. 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 still copied and 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 Cisco Catalyst 9400 Series Switches does not use the syntax of the capture filter.

- **Display filter**—The display filter is applied by Wireshark, and its match criteria are similar to those of the capture filter. Packets that fail the display filter are not 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.



Note When specifying CAPWAP as an attachment point, the core system filter is not used.

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. These implicitly constructed class maps are not reflected in the device running-config and are not NVGEN'd.

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

Capture Filter

The capture filter allows you to direct Wireshark to further filter incoming packets based on various conditions. Wireshark applies the capture filter immediately on receipt of the packet; packets that fail the capture filter are neither stored nor displayed.

A device receives this parameter and passes it unchanged to Wireshark. Because Wireshark parses the application filter definition, the defining syntax is the one provided by the Wireshark display filter. This syntax and that of standard Cisco IOS differ, which allows you to specify ACL match criteria that cannot be expressed with standard syntax.

Note The capture filter syntax matches that of the Wireshark display filter. The syntax for capture and display filters are identical in the Wireshark implementation on the Catalyst 4500 Series Switches.

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. Because the syntax of the display filter is identical to the capture filter, the display filter is superfluous if a capture filter is also defined.

Actions

Wireshark can be invoked 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 capture and display filters:

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

When invoked on a .pcap file only, only the decode and display action is applicable.

Storage of Captured Packets to Buffer in Memory

Packets can be stored in the capture buffer in memory for subsequent decoding, analysis, or storage to a .pcap file.

The capture buffer can be in linear or circular mode. In linear mode, new packets are discarded when the buffer is full. In circular mode, if the buffer is full, the oldest packets are discarded to accommodate the new packets. Although the buffer can also be cleared when needed, this mode is mainly used for debugging network traffic. However, it is not possible to only 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 Wireshark is used on switches in a stack, packet captures can be stored only on flash or USB flash devices connected to the active switch.

For example, if bootflash1 is connected to the active switch, and bootflash2 is connected to the secondary switch, only bootflash1 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 will probably result in errors.

Wireshark can store captured packets to a .pcap file. The capture file can be located on the following storage devices:

- Device on-board flash storage (bootflash:)
- external flash disk (slot:)
- USB drive (usb0:)



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

When configuring a Wireshark capture point, you can associate a filename. When the capture point is activated, Wireshark creates a file with the specified name and writes packets to it. If the file already exists at the time of creation of the capture point, Wireshark queries you as to whether the file can be overwritten. If the file already exists at the time of activating the capture point, Wireshark will overwrite the existing file. Only one capture point may be associated with a given filename.

If the destination of the Wireshark writing process is full, Wireshark fails with partial data in the file. You must ensure that there is 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 do not require details beyond the first 64 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. Memory buffer size can be specified 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 capture and 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. A capture filter is not applicable in this situation.

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 and capture filters are applicable here.

Wireshark Capture Point Activation and Deactivation

After a Wireshark capture point has been defined with its attachment points, filters, actions, and other options, it must be activated. Until the capture point is activated, it does not actually capture packets.

Before a capture point is activated, some functional checks are performed. A capture point cannot be activated if it has neither a core system filter nor attachment points defined. Attempting to activate a capture point that does not meet these requirements generates an error.

The capture and display filters are specified as needed.

After Wireshark capture points are activated, they can be deactivated in multiple ways. A capture point that is storing only packets to a .pcap file can be halted manually or configured with time or packet limits, after which the capture point halts automatically. Only packets that pass the Wireshark capture filter are counted against the packet limit threshold.

When a Wireshark capture point is activated, a fixed rate policer is applied automatically in the hardware so that the CPU is not flooded with Wireshark-directed packets. The disadvantage of the rate policer is that you cannot 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 3 attachment points, the rates of all 3 attachment points added together is policed to 1000 pps.



Note Policer is not 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:

- Layer 2 security features—Packets that are dropped by Layer 2 security features (such as port security, MAC address filtering, and spanning tree) are not captured by Wireshark. This differs from the behavior of SPAN.
- If port security and Wireshark are applied on an ingress capture, a packet that is dropped by port security will still be captured by Wireshark. If port security is applied on an ingress capture, and Wireshark is applied on an egress capture, a packet that is dropped by port security will not be captured by Wireshark.
- Packets dropped by Dynamic ARP Inspection (DAI) are not captured by Wireshark.
- If a port that is in STP blocked state is used as an attachment point and the core filter is matched, Wireshark will capture the packets that come into the port, even though the packets will be dropped by the switch.
- Classification-based security features—Packets that are dropped by input classification-based security features (such as ACLs and IPSG) are not caught by Wireshark capture points that are connected to attachment points at the same layer. In contrast, packets that are dropped by output classification-based security features are caught by Wireshark capture points that are connected to attachment points at the same layer. The logical model is that the Wireshark attachment point occurs after the security feature lookup on the input side, and symmetrically before the security feature lookup on the output side.

Wireshark capture policies connected to Layer 2 attachment points in the input direction capture packets dropped by Layer 3 classification-based security features. Symmetrically, Wireshark capture policies attached to Layer 3 attachment points in the output direction capture packets dropped by Layer 2 classification-based security features.

- Classification-based security features—Packets that are dropped by input classification-based security features (such as ACLs and IPSG) are not caught by Wireshark capture points that are connected to attachment points at the same layer. In contrast, packets that are dropped by output classification-based security features are caught by Wireshark capture points that are connected to attachment points at the same layer. The logical model is that the Wireshark attachment point occurs after the security feature lookup on the input side, and symmetrically before the security feature lookup on the output side.

On ingress, a packet goes through a Layer 2 port, a VLAN, and a Layer 3 port/SVI. On egress, the packet goes through a Layer 3 port/SVI, a VLAN, and a Layer 2 port. If the attachment point is before the point where the packet is dropped, Wireshark will capture the packet. Otherwise, Wireshark will not capture the packet. For example, Wireshark capture policies connected to Layer 2 attachment points in the input direction capture packets dropped by Layer 3 classification-based security features. Symmetrically, Wireshark capture policies attached to Layer 3 attachment points in the output direction capture packets dropped by Layer 2 classification-based security features.

- Routed ports and Layer 3 port channels—When a routed port or Layer 3 port channel is used as a Wireshark attachment point, the policy that is applied to capture the packets is treated as attached at Layer 3. Wireshark only captures packets that are being routed by the interface.
- Routed ports and switch virtual interfaces (SVIs)—Wireshark cannot capture the output of an SVI because the packets that go out of an SVI's output are generated by CPU. To capture these packets, include the control plane as an attachment point.
- VLANs—Starting with Cisco IOS Release 16.1, when a VLAN is used as a Wireshark attachment point, packet capture is supported on L2 and L3 in both input and output directions.

- **VLANs**—When a VLAN is used as a Wireshark attachment point, packets are captured in both input and output directions. A packet that is bridged in the VLAN generates two copies, one on input and one on output.
- **Private VLANs**—Secondary PVLANS are disallowed as Wireshark attachment points. Using a primary PVLAN as a Wireshark attachment point enables capture of packets in the primary PVLAN and all associated secondary PVLANS. The entire PV domain becomes the attachment point.
- **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 will not capture them.
- **SPAN**—Wireshark cannot 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.
- You can capture packets from a maximum of 1000 VLANs at a time, if no ACLs are applied. If ACLs are applied, the hardware will have less space for Wireshark to use. As a result, the maximum number of VLANs that can be used for packet capture at a time will be lower. Using more than 1000 VLANs tunnels at a time or extensive ACLs might have unpredictable results. For example, mobility may go down.



Note Capturing an excessive number of attachment points at the same time is strongly discouraged because it may cause excessive CPU utilization and unpredictable hardware behavior.

Guidelines for Configuring Wireshark

- During Wireshark packet capture, hardware forwarding happens concurrently.
- Because packet forwarding typically occurs in hardware, packets are not 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 of time, 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 very little traffic matches the core filter.
- Limit the number of Wireshark instances to two or less to avoid CPU or memory resource drain.

You can use up to eight Wireshark instances. An active **show** command that decodes and displays packets from a .pcap file or capture buffer counts as one instance.

- Whenever an ACL is installed or modified on a switch in the ingress direction, for the first 15 seconds, the software ignores packet classification details sent by the hardware. Instead, it uses software-based classification for the packets received by CPU. So, during this period, the system can only capture fewer packets (compared to the time after the first 15 seconds) and CPU usage will be high.
- Whenever an ACL that is associated with a running capture is modified, you must restart the capture for the ACL modifications to take effect. If you do not restart the capture, it will continue to use the original ACL as if it had not been modified.
- Writing to flash disk is a CPU-intensive operation, so 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 do not specify the display option) while capturing live packets rather than 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 use the default buffer size, packets may be dropped. Increase buffer size and avoid packet loss.
- If you want to decode and display live packets in the console window, ensure that the Wireshark session is bounded 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.



Note A core filter is required except when using a CAPWAP tunnel interface as a capture point attachment point.

- 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 is necessary to remove a parameter.
- Wireshark allows you to specify one or more attachment points. To add more than one attachment point, reenter the command with the new attachment point. To remove an attachment point, use the **no** form of the command. You can specify an interface range as an attachment point.

For example, enter **monitor capture mycap interface gi 3/1 in** where interface gi 3/1 is an attachment point. If you also need to attach interface gi 3/2, enter it as **monitor capture mycap interface gi 3/2 in**.

- 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.
- You cannot modify any parameters of a capture point while a session is active. To modify any parameter, stop the session, make the changes, and restart the session. Because an access list is generic to a switch and unrelated to the Wireshark process, it is alterable during a Wireshark session.
- If the file already exists at the time of creation of the capture point, Wireshark queries you as to whether the file can be overwritten. If the file already exists at the time of activating the capture point, Wireshark will overwrite the existing file.
- The Wireshark capture session operates normally in streaming mode where packets are both captured and processed. However, when you specify a buffer size of at least 32 MB but less than 80MB, the session automatically turns on lock-step mode in which a Wireshark capture session is split into two phases: capture and process. In the capture phase, the packets are stored in the temporary buffer. The duration parameter in lock-step mode serves as capture duration rather than session duration. When the buffer is full or the capture duration or packet limit has been attained, a session transitions to the process phase, wherein it stops accepting packets and starts processing packets in the buffer. You can also stop the capture manually. You will see a message in the output when the capture stops. With this second approach (lock-step mode), a higher capture throughput can be achieved. Last, when you specify a buffer size of at least 80MB, the session turns on lock-step mode with high-speed capture. This is similar to lock-step mode except that it captures the packets directly from the hardware queue and passes the packet to the Wireshark packet queue.



Note If you are capturing packets to a buffer, there is no file storage defined. Hence, you must export your capture from the buffer to a static storage file. Use the **monitor capture capture-name export file-location : file-name** command.

- 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 a stop condition such as duration or packet capture limit is met, or if an internal error occurs, or resource is full (specifically if disk is full in file mode).
- Dropped packets will not be shown at the end of the capture. However, only the count of dropped and oversized packets will be displayed.

Default Wireshark Configuration

The table below shows the default Wireshark configuration.

Feature	Default Setting
Duration	No limit
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. The packet capture rate can be throttled using further administrative controls. For example, options allow for filtering the packets to be captured using an Access Control List and, optionally, further defined by specifying a maximum packet capture rate or by specifying a sampling interval.

Prior to Cisco IOS XE Amsterdam 17.2.1, EPC is not 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 is 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 capture point's parameters.
3. Activate or deactivate a capture point.
4. Delete the capture point when you are no longer using it.

Defining a Capture Point

The example in this procedure defines a very 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 You must define an attachment point, direction of capture, and core filter to have a functional capture point.

An exception to needing to define a core filter is when you are defining a wireless capture point using a CAPWAP tunneling interface. In this case, you do not define your core filter. It cannot be used.

Follow these steps to define a capture point.

Procedure

	Command or Action	Purpose
Step 1	<p>enable</p> <p>Example:</p> <pre>Device> enable</pre>	<p>Enables privileged EXEC mode.</p> <p>Enter your password if prompted.</p>
Step 2	<p>monitor capture {<i>capture-name</i>} {interface <i>interface-type</i> <i>interface-id</i> control-plane} {in out both}</p> <p>Example:</p> <pre>Device# monitor capture mycap interface GigabitEthernet1/0/1 in</pre>	<p>Defines the capture point, specifies the attachment point with which the capture point is associated, and specifies the direction of the capture.</p> <p>The keywords have these meanings:</p> <ul style="list-style-type: none"> • <i>capture-name</i>—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 8 characters. Only alphanumeric characters and underscore (_) is permitted • (Optional) interface <i>interface-type</i> <i>interface-id</i>—Specifies the attachment point with which the capture point is associated (GigabitEthernet1/0/1 is used in the example). <p>Note Optionally, you can define multiple attachment points and all of 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.</p> <p>Use one of the following for <i>interface-type</i>:</p> <ul style="list-style-type: none"> • GigabitEthernet—Specifies the attachment point as GigabitEthernet. • vlan—Specifies the attachment point as a VLAN. <p>Note Only ingress capture (in) is allowed when using this interface as an attachment point.</p>

	Command or Action	Purpose
		<ul style="list-style-type: none"> • capwap—Specifies the attachment point as a CAPWAP tunnel. <p>Note When using this interface as an attachment point, a core filter cannot be used.</p> <ul style="list-style-type: none"> • (Optional) control-plane—Specifies the control plane as an attachment point. • in out both—Specifies the direction of capture.
Step 3	<p>monitor capture {<i>capture-name</i>} [match {any ipv4 any any ipv6 any any}]</p> <p>Example:</p> <pre>Device# monitor capture mycap interface GigabitEthernet1/0/1 in match any</pre>	<p>Defines the core system filter.</p> <p>Note When using the CAPWAP tunneling interface as an attachment point, do not perform this step because a core filter cannot be used.</p> <p>The keywords have these meanings:</p> <ul style="list-style-type: none"> • <i>capture-name</i>—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. <p>Note A capture point cannot be activated if it has neither a core system filter nor attachment points defined. Attempting to activate a capture point that does not meet these requirements generates an error.</p> <ul style="list-style-type: none"> • ipv4—Specifies an IP version 4 filter. • ipv6—Specifies an IP version 6 filter.
Step 4	<p>show monitor capture {<i>capture-name</i>} [parameter]</p> <p>Example:</p> <pre>Device# show monitor capture mycap parameter monitor capture mycap interface GigabitEthernet1/0/1 in monitor capture mycap match any</pre>	<p>Displays the capture point parameters that you defined in Step 2 and confirms that you defined a capture point.</p>

	Command or Action	Purpose
Step 5	show capwap summary Example: Device# show capwap summary	Displays the CAPWAP tunnels available as attachment points for a wireless capture. Note Use this command only if you are using a CAPWAP tunnel as an attachment point to perform a wireless capture. See the CAPWAP example in the examples section.
Step 6	show running-config Example: Device# show running-config	Verifies your entries.
Step 7	copy running-config startup-config Example: Device# copy running-config startup-config	(Optional) Saves your entries in the configuration file.

Example

To define a capture point with a CAPWAP attachment point:

```

Device# show capwap summary

CAPWAP Tunnels General Statistics:
  Number of Capwap Data Tunnels      = 1
  Number of Capwap Mobility Tunnels   = 0
  Number of Capwap Multicast Tunnels  = 0

Name  APName                               Type PhyPortIf Mode      McastIf
-----
Ca0   AP442b.03a9.6715                       data Gi3/0/6   unicast  -

Name  SrcIP          SrcPort DestIP          DstPort DtlsEn MTU    Xact
-----
Ca0   10.10.14.32    5247   10.10.14.2     38514   No     1449  0

Device# monitor capture mycap interface capwap 0 both
Device# monitor capture mycap file location flash:mycap.pcap
Device# monitor capture mycap file buffer-size 1
Device# monitor capture mycap start

*Aug 20 11:02:21.983: %BUFCAP-6-ENABLE: Capture Point mycap enabled.on

Device# show monitor capture mycap parameter
  monitor capture mycap interface capwap 0 in
  monitor capture mycap interface capwap 0 out
  monitor capture mycap file location flash:mycap.pcap buffer-size 1
    
```

```

Device#
Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
  Interface: CAPWAP,
  Ingress:
0
  Egress:
0
  Status : Active
  Filter Details:
  Capture all packets
  Buffer Details:
  Buffer Type: LINEAR (default)
  File Details:
  Associated file name: flash:mycap.pcap
  Size of buffer(in MB): 1
  Limit Details:
  Number of Packets to capture: 0 (no limit)
  Packet Capture duration: 0 (no limit)
  Packet Size to capture: 0 (no limit)
  Packets per second: 0 (no limit)
  Packet sampling rate: 0 (no sampling)
Device#
Device# show monitor capture file flash:mycap.pcap
  1  0.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
  2  0.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
  3  2.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
  4  2.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
  5  3.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
  6  4.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
  7  4.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
  8  5.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
  9  5.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
 10  6.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
 11  8.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
 12  9.225986 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 13  9.225986 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 14  9.225986 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 15  9.231998 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 16  9.231998 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 17  9.231998 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 18  9.236987 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 19 10.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
 20 10.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
 21 12.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
  Flags=.....
 22 12.239993 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 23 12.244997 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
 24 12.244997 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data

```

```

25 12.250994 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
26 12.256990 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
27 12.262987 10.10.14.2 -> 10.10.14.32 DTLSv1.0 Application Data
28 12.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
Flags=.....
29 12.802012 10.10.14.3 -> 10.10.14.255 NBNS Name query NB WPAD.<00>
30 13.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
Flags=.....
    
```

What to do next

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



Note You cannot change a capture point's parameters using the methods presented in this topic.

If the user enters an incorrect capture name, or an invalid/non existing attachment point, the switch will show errors like "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, the steps to specify values for the parameters can be executed 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 will need to confirm interactively when certain parameters already specified are being modified.

Follow these steps to modify a capture point's parameters.

Before you begin

A capture point must be defined 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.

	Command or Action	Purpose
Step 3	<p>monitor capture {<i>capture-name</i>} limit { [duration <i>seconds</i>] [packet-length <i>size</i>] [packets <i>num</i>] }</p> <p>Example:</p> <pre>Device# monitor capture mycap limit duration 60 packet-len 400</pre>	Specifies the session limit in seconds (60), packets captured, or the packet segment length to be retained by Wireshark (400).
Step 4	<p>monitor capture {<i>capture-name</i>} file {<i>location filename</i>}</p> <p>Example:</p> <pre>Device# monitor capture mycap file location flash:mycap.pcap</pre>	<p>Specifies the file association, if the capture point intends to capture packets rather than only display them.</p> <p>Note If the file already exists, you have to confirm if it can be overwritten.</p>
Step 5	<p>monitor capture {<i>capture-name</i>} file {buffer-size <i>size</i>}</p> <p>Example:</p> <pre>Device# monitor capture mycap file buffer-size 100</pre>	Specifies the size of the memory buffer used by Wireshark to handle traffic bursts.
Step 6	<p>show monitor capture {<i>capture-name</i>} [parameter]</p> <p>Example:</p> <pre>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</pre>	Displays the capture point parameters that you defined previously.
Step 7	<p>end</p> <p>Example:</p> <pre>Device(config)# end</pre>	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
```

What to do next

If your capture point contains all of 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 location 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.
Step 5	show monitor capture {capture-name} [parameter] Example:	Displays the capture point parameters that remain defined after your parameter deletion operations. You can run this command at any

	Command or Action	Purpose
	Device# show monitor capture mycap parameter monitor capture mycap interface GigabitEthernet1/0/1 in	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: Device# show monitor capture mycap parameter Capture mycap does not exist	Displays a message indicating that the specified capture point doesn't exist because it was deleted.
Step 4	end Example:	Returns to privileged EXEC mode.

	Command or Action	Purpose
	<code>Device(config)# end</code>	
Step 5	show running-config Example: <code>Device# show running-config</code>	Verifies your entries.
Step 6	copy running-config startup-config Example: <code>Device# copy running-config startup-config</code>	(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

A capture point can be activated even if an attachment point and a core system filter have been defined and the associated filename already exists. In such an instance, the existing file will be overwritten.

A capture point with no associated filename can only be activated to display. When the filename is not 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 are not displayed live, and all the packets captured by the core system filter are displayed. The default display mode is brief.



Note When using a CAPWAP tunneling interface as an attachment point, core filters are not used, so there is no requirement to define them in this case.

Procedure

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

	Command or Action	Purpose
Step 2	<pre>monitor capture {capture-name} start [capture-filter filter-string] [display [display-filter filter-string]] [brief detailed dump] Example: Device# monitor capture mycap start display display-filter "net 10.1.1.0 0.0.0.255 and port 80"</pre>	Activates a capture point.
Step 3	<pre>monitor capture {capture-name} start [display [display-filter filter-string]] [brief detailed dump] Example: Device# monitor capture mycap start display display-filter "stp"</pre>	Activates a capture point and filters the display, so only packets containing "stp" are displayed.
Step 4	<pre>monitor capture {capture-name} stop Example: Device# monitor capture name stop</pre>	Deactivates a capture point.
Step 5	<pre>end Example: Device(config)# end</pre>	Returns to privileged EXEC mode.
Step 6	<pre>show running-config Example: Device# show running-config</pre>	Verifies your entries.
Step 7	<pre>copy running-config startup-config Example: Device# copy running-config startup-config</pre>	(Optional) Saves your entries in the configuration file.

Activating Capture Point with Capture Filter

```
Device# monitor capture mycap start capture-filter "net 10.1.1.0 0.0.0.255 and port 80"
```

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 { <i>capture-name</i> } [clear export <i>filename</i>]	Clear - Completely deletes the buffer.

	Command or Action	Purpose
	Example: Device# <code>monitor capture mycap clear</code>	Note When you run the clear command, <ul style="list-style-type: none"> • On DNA Advantage license - the command clears the buffer contents without deleting the buffer. • On all other licenses - the command deletes the buffer itself. Export - Saves the captured packets in the buffer and deletes the buffer.
Step 3	end Example: Device(config)# <code>end</code>	Returns to privileged EXEC mode.
Step 4	show running-config Example: Device# <code>show running-config</code>	Verifies your entries.
Step 5	copy running-config startup-config Example: Device# <code>copy running-config startup-config</code>	(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



Note Export of an active capture point is only supported on DNA Advantage. For all other type of licenses you need to stop the capture first and only then export.

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 <i>capture-name</i> access-list <i>access-list-name</i> 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 <i>capture-name</i> limit duration <i>seconds</i> Example: Device# monitor capture mycap limit duration 1000	Configures monitor capture limits.
Step 4	monitor capture <i>capture-name</i> interface <i>interface-name</i> 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 <i>capture-name</i> buffer circular <i>size bytes</i>	Configures a buffer to capture packet data.

	Command or Action	Purpose
	Example: <pre>Device# monitor capture mycap buffer circular size 10</pre>	
Step 6	monitor capture <i>capture-name</i> start Example: <pre>Device# monitor capture mycap start</pre>	Starts the capture of packet data at a traffic trace point into a buffer.
Step 7	monitor capture <i>capture-name</i> stop Example: <pre>Device# monitor capture mycap stop</pre>	Stops the capture of packet data at a traffic trace point.
Step 8	monitor capture <i>capture-name</i> export <i>file--location/file-name</i> Example: <pre>Device# monitor capture mycap export tftp://10.1.88.9/mycap.pcap</pre>	Exports captured data for analysis.
Step 9	end Example: <pre>Device# end</pre>	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: <pre>Device> enable</pre>	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	show monitor capture <i>capture-buffer-name</i> buffer dump Example: <pre>Device# show monitor capture mycap buffer dump</pre>	(Optional) Displays a hexadecimal dump of captured packet and its metadata.

	Command or Action	Purpose
Step 3	show monitor capture <i>capture-buffer-name</i> 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)

```

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

```

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: 0000000009c98f77abcdabcdabcdabcdabcdabcdabcd...
      [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 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 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 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:

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

```
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 has been 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 is not required in this particular case since we have set a limit and the capture will automatically stop 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 stop automatically after the time has elapsed or the packet count has been 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 have been 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: 000000000b153063abcdabcdabcdabcdabcdabcdabcd...
      [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 contents to be displayed, 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 cannot run show from buffer during an active capture. Capture should be stopped 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 current capture session's pcap file 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 the command is run, export is "started" but not complete when it returns the prompt to the user. 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
```

Example: Simple Capture and Store of Packets in Egress Direction

```

 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 stop automatically after the time has elapsed or the packet count has been met. When you see the following message in the output, will know that the capture operation has stopped:

```
*Oct 14 09:36:34.632: %BUFCAP-6-DISABLE_ASYNC: Capture Point mycap disabled. Reason : 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 .....
0030: 1D006369 73636F00 0000091D 0001 ..example.....
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

Error Message Decoder

Description	Link
To help you research and resolve system error messages in this release, use the Error Message Decoder tool.	https://www.cisco.com/cgi-bin/Support/Errordecoder/index.cgi

Feature History for Configuring 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 noted otherwise.

Table 2: Feature History for Configuring Packet Capture

Release	Feature Name	Feature Information
Cisco IOS XE Everest 16.6.1	Configuring Packet Capture	This feature was introduced.
Cisco IOS XE Amsterdam 17.2.1	Configuring EPC on an interface either in down state or admin state.	Configuring EPC on an interface that is either in down state or admin down state does not affect packet capture once the interface changes to up state.